Oracle® Cloud

Blockchain App Builder for Oracle Blockchain Platform

G36555-02
September 2025

Oracle Cloud Blockchain App Builder for Oracle Blockchain Platform,

G36555-02

Copyright © 2025, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

About This Content

1 Build Chaincodes with Low-Code Blockchain App Builder

2 Using the Blockchain App Builder Command-Line Interface

Install and Configure Blockchain App Builder CLI 3
Upgrade Blockchain App Builder CLI 9
Create a Chaincode Project with the Blockchain App Builder CLI 10
- Input Specification File 11
- Scaffolded TypeScript Chaincode Project 21
- Scaffolded Go Chaincode Project 40

Deploy Your Chaincode Using the CLI 59
- Deploy Your Chaincode to a Local Hyperledger Fabric Network 60
- Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network 65
- Package Your Chaincode Project for Manual Deployment to Oracle Blockchain Platform 66

Test Your Chaincode Using the CLI 67
- Test Your Chaincode on a Local Hyperledger Fabric Network 67
- Test Your Chaincode on a Remote Oracle Blockchain Platform Network 72
- Execute Berkeley DB SQL Rich Queries 73

Upgrading Chaincode Projects in the CLI 74
Synchronize Specification File Changes With Generated Source Code 75
Apply a Patch to the Blockchain App Builder CLI 76
Writing Unit Test Cases and Coverage Reports for the Chaincode Project 77
Generate a Postman Collection Using the CLI 77
Troubleshoot Blockchain App Builder CLI 82

3 Using the Blockchain App Builder Extension for Visual Studio Code

Install and Configure the Blockchain App Builder Extension for Visual Studio Code 2
Upgrade the Blockchain App Builder Extension for Visual Studio Code 8
Create a Chaincode Project with the Blockchain App Builder Visual Studio Code Extension 8
- Input Specification File 11

Copyright © 2025, Oracle and/or its affiliates.

Scaffolded TypeScript Chaincode Project 20
Scaffolded Go Chaincode Project 39
Deploy Your Chaincode Using Visual Studio Code 58
Deploy the Chaincode to a Local Hyperledger Fabric Network 59
Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network 62
Package Your Chaincode Project for Manual Deployment to Oracle Blockchain Platform 63
Test Your Chaincode Using Visual Studio Code 64
Test Your Chaincode on a Local Hyperledger Fabric Network 64
Testing Lifecycle Operations on a Remote Oracle Blockchain Platform Network 65
Execute Berkeley DB SQL Rich Queries 66
Generate CLI Commands from Queries 67
Upgrading Chaincode Projects in Visual Studio Code 67
Synchronize Specification File Changes With Generated Source Code 68
Debugging from Visual Studio Code 69
Generate a Postman Collection Using Visual Studio Code 70
Troubleshoot Blockchain App Builder Visual Studio Code Extension 74

4 Tokenization Support Using Blockchain App Builder

Tokenization 1
Tokens and the Account/Balance Model 2
Token Standards 3
Tokenization Flow 3
Access Control 5
MVCC Optimization 6
Token Taxonomy Framework 7
Input Specification File for Token Taxonomy Framework 8
Scaffolded TypeScript Project for Token Taxonomy Framework 12
TypeScript Methods for Token Conversion 156
TypeScript Methods for Token Account Status 179
Scaffolded Go Project for Token Taxonomy Framework 192
Go Methods for Token Conversion 348
Go Methods for Token Account Status 373
ERC-721 389
Input Specification File for ERC-721 389
Scaffolded TypeScript NFT Project for ERC-721 394
TypeScript Methods for ERC-721 NFT Locking 466
TypeScript Methods for ERC-721 Token Account Status 475
Scaffolded Go NFT Project for ERC-721 482
Go Methods for ERC-721 NFT Locking 553
Go Methods for ERC-721 Token Account Status 564
ERC-1155 573

Copyright © 2025, Oracle and/or its affiliates.

Input Specification File for ERC-1155 573
ERC-1155 Tokenization Flow 580
Scaffolded TypeScript Token Project for ERC-1155 584
TypeScript Methods for ERC-1155 NFT Locking 683
TypeScript Methods for ERC-1155 Token Account Status 692
Scaffolded Go Token Project for ERC-1155 700
Go Methods for ERC-1155 NFT Locking 808
Go Methods for ERC-1155 Token Account Status 818
Deploying and Testing Token Chaincode 828
Working With the Sample Token Specification Files 828
Disaster Recovery Support for Tokenization 829
Digital Assets Edition Enhancements 830
Wrapper APIs 832
Generate Wrapper APIs Using the CLI 833
Generate Wrapper APIs Using Visual Studio Code 836
Wrapper API Package Components 838
Deploy Wrapper APIs 841
Chaincode Events 844
Token Taxonomy Framework Enhancements 851
Confidential Payments Model 852
Application Models 969
Enhancements to Postman Collections 970
Auditor Roles 974
Method Level Access Controls 974

Copyright © 2025, Oracle and/or its affiliates.

About This Content

Blockchain App Builder is a low-code development, debugging, and deployment tool that works with Oracle Blockchain Platform networks.

Audience

This guide is intended for anyone who uses the Blockchain App Builder tool with Oracle Blockchain Platform.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Copyright © 2025, Oracle and/or its affiliates.

1

Build Chaincodes with Low-Code Blockchain App Builder

Blockchain App Builder for Oracle Blockchain Platform is a tool set that assists with rapid development, testing, debugging, and deployment of chaincode on Oracle Blockchain Platform networks, comprising cloud BaaS nodes on Oracle Cloud Infrastructure or on-premises nodes using Enterprise Edition.

For more information on Oracle Blockchain Platform beyond Blockchain App Builder, see Using Oracle Blockchain Platform.

A smart contract (also known as a chaincode) defines the different states of a business object between two or more parties and business logic that validates and implements changes as the object moves among these different states. At the heart of every blockchain application is one or more chaincodes. A chaincode must be bug-free and tested before it is deployed.

Blockchain App Builder enables you to generate complex chaincodes in TypeScript (for Node.js chaincode) and Go from a simple specification file. You can specify multiple asset definitions and behaviors in the specification file. You can then generate and test your chaincodes either on your local system by using a preconfigured instance of Hyperledger Fabric inside Blockchain App Builder, or by connecting to your Oracle Blockchain Platform network.

> **Note**
>
> Although JavaScript isn't supported by Blockchain App Builder, because TypeScript projects are compiled to JavaScript, you can add basic JavaScript to a TypeScript project if needed.

Blockchain App Builder supports the full development life cycle either from a command-line interface or as an extension for Visual Studio Code.

To get the Blockchain App Builder tools and samples, in the console open the **Developer Tools** tab and then select the **Blockchain App Builder** pane. From here you can download the command-line interface tools or the Visual Studio Code extension. Additionally, there are samples such as Fabcar, Marbles, Fiat Money Token, Loyalty Token, NFT Art Collection Marketplace, and Fractional NFT in Real Estate, which you can use to see how the tools work or as a template for your own chaincode projects.

Oracle Blockchain Platform Digital Assets Edition includes an enhanced version of Blockchain App Builder and other functionality including sample applications to support wholesale central bank digital currency (CBDC), bond marketplaces, and generic token frameworks. For more information, see Oracle Blockchain Platform Digital Assets Edition.

Copyright © 2025, Oracle and/or its affiliates.

2

Using the Blockchain App Builder Command-Line Interface

The Blockchain App Builder command-line interface helps you build and scaffold a fully-functional chaincode project from a specification file.

After the project is built, you can run and test it on a local Hyperledger Fabric network, or your provisioned Oracle Blockchain Platform network. You can then run SQL rich queries, debug the chaincode, or write and run unit tests using the generated code.

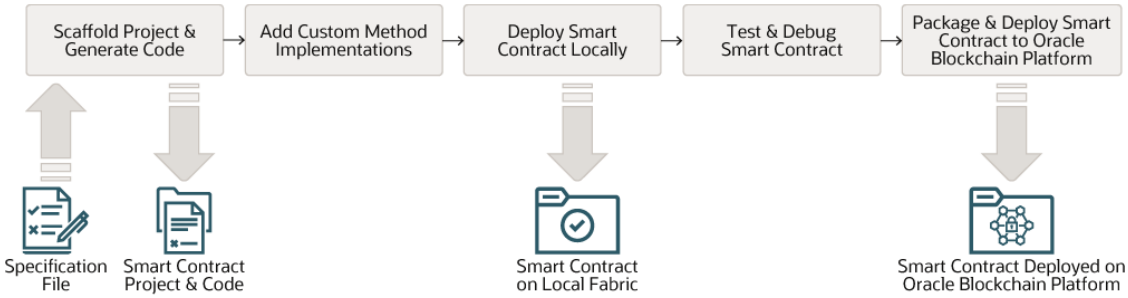


Table 2-1 Workflow When Using the CLI

Task	Description	Related Topics
Install and configure	Download the Blockchain App Builder CLI from your Oracle Blockchain Platform console and install it and any prerequisite software.	• Install and Configure Blockchain App Builder CLI
Create the chaincode project	Create a specification file for the chaincode project.	• Create a Chaincode Project with the Blockchain App Builder CLI

Copyright © 2025, Oracle and/or its affiliates.

Table 2-1 (Cont.) Workflow When Using the CLI

Task	Description	Related Topics
Generate the chaincode	Edit the specification file to define the assets and chaincodes to generate, and then run the CLI initialization process to generate your chaincode from the specification file.	Detailed reference information about the structure and contents of the specification file and the generated chaincode project: • Input Specification File • Scaffolded TypeScript Chaincode Project • Scaffolded Go Chaincode Project Detailed information about tokenization support: • Tokenization Support Using Blockchain App Builder • Scaffolded TypeScript Token Project for ERC-1155 • Scaffolded Go Token Project for ERC-1155 • Scaffolded TypeScript NFT Project for ERC-721 • Scaffolded Go NFT Project for ERC-721 • Scaffolded TypeScript Project for Token Taxonomy Framework • Scaffolded Go Project for Token Taxonomy Framework
Deploy the chaincode	After your chaincode project is created, you can deploy it locally to the included preconfigured Hyperledger Fabric network, or remotely to your Oracle Blockchain Platform Cloud or Enterprise Edition. You can also package the chaincode project for manual deployment to Oracle Blockchain Platform.	• Deploy Your Chaincode to a Local Hyperledger Fabric Network • Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network • Package Your Chaincode Project for Manual Deployment to Oracle Blockchain Platform
Test the chaincode	After your chaincode is running on a network, you can test any of the generated methods. Additionally, if you chose to create the `executeQuery` method during your chaincode development, you can run SQL rich queries if your chaincode is deployed to an Oracle Blockchain Platform network.	• Test Your Chaincode on a Local Hyperledger Fabric Network • Test Your Chaincode on a Remote Oracle Blockchain Platform Network • Execute Berkeley DB SQL Rich Queries
Synchronize your updates	When you update your specification file, you can synchronize the changes with the generated chaincode files.	• Synchronize Specification File Changes With Generated Source Code
Apply patches to the Blockchain App Builder CLI	You can use the `patch` command to apply a patch to the Blockchain App Builder CLI.	• Apply a Patch to the Blockchain App Builder CLI
Run unit tests	A basic unit test case is included in the project. You can add and run additional tests.	• Writing Unit Test Cases and Coverage Reports for the Chaincode Project

Copyright © 2025, Oracle and/or its affiliates.

Install and Configure Blockchain App Builder CLI

The following platforms are supported:

- macOS
- Oracle Linux 8.0 or 9.0
- Microsoft Windows 10 or 11

After you've completed the installation process:

- Verify your installation.
- If you're using Go chaincode projects, complete the additional configuration steps.

Prerequisites

Before you install Blockchain App Builder CLI on your local system, you must install the prerequisites.

> **Note**
>
> Blockchain App Builder coordinates with Oracle Blockchain Platform and its compilers. If you use any versions of the prerequisites other than the ones mentioned in the following section, deploying your chaincode to a remote Oracle Blockchain Platform network might fail.

When you install Blockchain App Builder, a prerequisites check runs first. If the prerequisites check fails, the installation process stops.

- macOS
- Linux
- Windows

macOS

Prerequisites

- Rancher Desktop (tested with 1.19.3). Blockchain App Builder can also work with Docker, but it has been tested and verified with Rancher Desktop. If you plan to use Rancher Desktop, uninstall Docker completely before installing Rancher Desktop. After you install Rancher Desktop, ensure that the container runtime is set to `dockerd (moby)`. To verify the container runtime in Rancher Desktop 1.19.3, click **Kubernetes Settings** > **Container Runtime**.
- The latest release of Node.js version 18 (tested with 18.15.0 and 18.16.0). Do not use versions of Node.js earlier or later than version 18.
- npm v8.x or v9.x (tested with 9.5.0 and 9.5.1)
- Go v1.23.2 or later, but not Go v1.24 or later. After installing Blockchain App Builder, see Additional Setup for Go Chaincode Projects.

Copyright © 2025, Oracle and/or its affiliates.

- If you want to use the synchronization feature of Blockchain App Builder, install Git and then configure your user name and email as shown in the following commands. Specify your user name and email address in the place of `<your_name>` and `<email>`.

  ```
  git config --global user.name "<your_name>"


  git config --global user.email "<email>"
  ```

Install Node.js and npm by Using nvm

Using nvm to install Node.js and npm gives you the ability to run more commands without `sudo`.

1. Enter the following command to install nvm:

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh |
   bash
   ```

2. Add the following code snippet to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

   ```
   export NVM_DIR="$HOME/.nvm"
   [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
   [ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This
   loads nvm bash_completion
   ```

3. Log out and then log back in to your operating system.
4. Enter the following command to verify the nvm installation:

   ```
   nvm version
   ```

5. Enter the following command to install Node.js and npm:

   ```
   nvm install 18.16.0
   ```

6. Enter the following command to set Node.js 18.16.0 as the default in nvm:

   ```
   nvm alias default 18.16.0
   ```

Linux

Prerequisites

- Docker v20.10.0 or later
- Docker Compose v1.23.0 or later
- The latest release of Node.js version 18 (tested with 18.15.0 and 18.16.0). Do not use versions of Node.js earlier or later than version 18.
- npm v8.x or v9.x (tested with 9.5.0 and 9.5.1)
- Go v1.23.2 or later, but not Go v1.24 or later. After installing Blockchain App Builder, see Additional Setup for Go Chaincode Projects.

Copyright © 2025, Oracle and/or its affiliates.

- If you want to use the synchronization feature of Blockchain App Builder, install Git and then configure your user name and email as shown in the following commands. Specify your user name and email address in the place of `<your_name>` and `<email>`.

  ```
  git config --global user.name "<your_name>"


  git config --global user.email "<email>"
  ```

Install Node.js and npm by Using nvm

Using nvm to install Node.js and npm gives you the ability to run more commands without `sudo`.

1. Enter the following command to install nvm:

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh |
   bash
   ```

2. Add the following code snippet to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

   ```
   export NVM_DIR="$HOME/.nvm"
   [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
   [ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This
   loads nvm bash_completion
   ```

3. Log out and then log back in to your operating system.
4. Enter the following command to verify the nvm installation:

   ```
   nvm version
   ```

5. Enter the following command to install Node.js and npm:

   ```
   nvm install 18.16.0
   ```

6. Enter the following command to set Node.js 18.16.0 as the default in nvm:

   ```
   nvm alias default 18.16.0
   ```

Install Docker

Ensure that `dnf` is updated and pointing to the current repository based on your kernel.

1. Enter the following command to add Docker to the repository list:

   ```
   sudo dnf config-manager --add-repo=https://download.docker.com/linux/
   centos/docker-ce.repo
   ```

2. Enter the following command to install Docker:

   ```
   dnf install docker-ce -y --allowerasing
   ```


Copyright © 2025, Oracle and/or its affiliates.

3. Enter the following command to start Docker as a service:

```
sudo systemctl enable --now docker
```

4. Enter the following commands to ensure that the current user has access to Docker:

```
sudo groupadd docker
sudo usermod -aG docker $USER
```

5. Enter the following command to restart the system:

```
sudo reboot
```

Install Docker Compose

1. Enter the following `curl` command to get Docker Compose:

```
sudo curl -L https://github.com/docker/compose/releases/download/v2.5.0/
docker-compose-linux-x86_64 -o /usr/local/bin/docker-compos
```

2. Enter the following command to add executable permissions to Docker Compose:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Windows

Prerequisites

- Rancher Desktop (tested with 1.19.3). Blockchain App Builder can also work with Docker, but it has been tested and verified with Rancher Desktop.
- The latest release of Node.js version 18 (tested with 18.15.0 and 18.16.0). Do not use versions of Node.js earlier or later than version 18.
- npm v8.x or v9.x (tested with 9.5.0 and 9.5.1)
- Go v1.23.2 or later, but not Go v1.24 or later. After installing Blockchain App Builder, see Additional Setup for Go Chaincode Projects.
- If you want to use the synchronization feature of Blockchain App Builder, install Git and configure your user name and email as shown in the following commands. Specify your user name and email address in the place of `<your_name>` and `<email>`.

```
git config --global user.name "<your_name>"


git config --global user.email "<email>"
```

Install Rancher Desktop

Complete the following steps to install Rancher Desktop on Microsoft Windows.

1. If Docker is installed on your local computer, uninstall it completely.
2. Download and install Rancher Desktop.

Copyright © 2025, Oracle and/or its affiliates.

3. After the installation wizard completes, before you open Rancher Desktop, run the following commands:

```
wsl --install
wsl --set-default-version 2
wsl --setdefault rancher-desktop
```

4. Open Rancher Desktop to complete the setup process.
5. After you install Rancher Desktop, ensure that the container runtime is set to `dockerd (moby)`. To verify the container runtime in Rancher Desktop 1.19.3, click **Kubernetes Settings** > **Container Runtime**.

Install Blockchain App Builder

Download the Blockchain App Builder CLI package (`oracle-ochain-cli-x.x.x.tgz`) from the **Developer Tools** tab on the **Blockchain App Builder** pane of the Oracle Blockchain Platform console.

- macOS
- Linux
- Windows

macOS

1. Enter the following command to install Xcode or the XCode command line tools (xcode-select).

```
sudo xcode-select –install
```

2. Enter the following command to install Blockchain App Builder (adjust the name of the `.tgz` file for the version that you are installing):

```
npm install -g oracle-ochain-cli-x.x.x.tgz
```

Mac OS Catalina can have issues with xcode-select. If you encounter these issues, use the following command to reset and restart xcode-select:

```
xcode-select –-reset
```

Linux

1. Enter the following command to install Blockchain App Builder (adjust the name of the `.tgz` file for the version that you are installing):

```
npm install -g oracle-ochain-cli-x.x.x.tgz
```

2. Log out as the current user and then log in again so that group membership takes effect.

Copyright © 2025, Oracle and/or its affiliates.

Windows

After you've installed all the prerequisite software, enter the following command to install Blockchain App Builder (adjust the name of the `.tgz` file for the version that you are installing):

```
npm install -g oracle-ochain-cli-x.x.x.tgz
```

Verify the Installation

In your terminal, type `ochain -v`. The output shows the Blockchain App Builder CLI usage, options, and commands.

Additional Setup for Go Chaincode Projects

To develop a Go project, you must set the `GOPATH` environment variable. This allows Go to locate your workspace and run your code.

- macOS
- Linux
- Windows

macOS

Before setting the `GOPATH` environment variable, make sure that a `go/` folder exists in your `$HOME` directory. If not, enter the following command to create a `go/` directory in your home directory:

```
mkdir $HOME/go
```

Set your `GOPATH` environment variable by adding the following variables to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

```
export PATH=$PATH:/usr/local/go/bin
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
```

After editing the file, run the following command to make your changes take effect immediately:

```
source ~/.bash_profile
```

Alternately, you can apply the change system-wide by adding the previous variables to the `/etc/bashrc` file.

Copyright © 2025, Oracle and/or its affiliates.

Linux

Before setting the `GOPATH` environment variable, make sure that a `go/` folder exists in your `$HOME` directory. If not, enter the following command to create a `go/` directory in your home directory:

```
mkdir $HOME/go
```

Set your `GOPATH` environment variable by adding the following variables to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

```
export PATH=$PATH:/usr/local/go/bin
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
```

After editing the file, run the following command to make your changes take effect immediately:

```
source ~/.bash_profile
```

Alternately, you can apply the change system-wide by adding the previous variables to the `/etc/bashrc` file.

Windows

Create a `go/` directory in your home directory: `C:\Users\<username>\go`.

Upgrade Blockchain App Builder CLI

To upgrade Blockchain App Builder, you must uninstall the previous version and then install the newer version.

1. Run the following command to uninstall Blockchain App Builder.

   ```
   npm uninstall -g @oracle/ochain-cli
   ```

2. Verify that Blockchain App Builder is no longer installed by running the following command.

   ```
   ochain -v
   ```

   If Blockchain App Builder is no longer installed, an error message similar to the following text is displayed.

   ```
   -bash: ochain: command not found
   ```

3. Download the latest version of the Blockchain App Builder CLI package (`oracle-ochain-cli-x.x.x.tgz`) from the **Developer Tools** tab on the Blockchain App Builder pane of the Oracle Blockchain Platform console, and then install Blockchain App Builder. For more information, see Install and Configure Blockchain App Builder CLI.

Copyright © 2025, Oracle and/or its affiliates.

Create a Chaincode Project with the Blockchain App Builder CLI

To create a chaincode project with Blockchain App Builder, you scaffold a chaincode project from a detailed specification file.

Background

Blockchain App Builder's `init` command initializes and scaffolds a ready-to-use chaincode project. Based on simple input, the `init` command can generate complex chaincode projects that include the following features:

- Multiple assets (models) and their behaviors (controllers)
- Auto-generated CRUD (Create/Read/Update/Delete) and non-CRUD methods
- Automatic validation of arguments
- Marshalling/unmarshalling of arguments
- Transparent persistence capability (ORM)
- Rich query support

The generated project follows the model/controller and decorator pattern, which allows an asset's properties that are maintained on the ledger to be specified as typed fields and extended with specific behaviors and validation rules. This pattern reduces the number of lines of code, which can help with readability and scalability.

Prerequisites

Before you scaffold a project, you must create an input specification file. For more information, see Input Specification File.

Scaffolding the Chaincode Project with the `init` Command

Entering `ochain init -h` lists the command usage and options. The `init` command has the following options:

- **--cc/-c:**
  The name of the chaincode project. The default value is `MyChaincode`.
- **--lang/-l**
  The language of the scaffolded chaincode. Blockchain App Builder supports TypeScript (`ts`) and Go (`go`). If no option is specified, the language defaults to `ts`.
- **--conf/-f** or **--spec**
  The path to an input specification file. Blockchain App Builder reads the input specification file and generates the scaffolded project with many helper tools, which help in reducing the overall development effort. Taking full advantage of the input specification file can significantly reduce the development time.

  The specification file can be in `yaml` or `json` format. If the path is not specified, it defaults to the current directory. See Input Specification File.
- **--out/-o**
  The output directory of the scaffolded chaincode project. If not specified, the scaffolded project is generated in the current directory.

  The output is a fully contained, lightweight, and scalable TypeScript or Go chaincode project.
- **--root/-r**

Copyright © 2025, Oracle and/or its affiliates.

Valid and required only for Go. The root directory in the *GOHOME* variable for your Go chaincodes. The default value is `example.com`.

Example

```
ochain init --cc MyNewTsProject --lang ts --conf spec.yml
```

Defaults

If no options are specified in the `ochain init` command, the name of the scaffolded project is `MyChaincode` and the language is TypeScript.

The `MyChaincode.model.ts` file contains only one asset, called `MyAsset`, with one property named `value`. The `MyChaincode.controller.ts` file contains one controller with the corresponding CRUD methods for the `MyAsset` model.

Output

When the process is complete, you'll have a fully functional chaincode project that you can deploy either locally or to a remote Oracle Blockchain Platform instance. For a detailed overview of the files that are created, see:

- Scaffolded TypeScript Chaincode Project
- Scaffolded Go Chaincode Project

For a detailed overview of a token-based project, see also:

- Scaffolded TypeScript Token Project for ERC-1155
- Scaffolded Go Token Project for ERC-1155
- Scaffolded TypeScript NFT Project for ERC-721
- Scaffolded Go NFT Project for ERC-721
- Scaffolded TypeScript Project for Token Taxonomy Framework
- Scaffolded Go Project for Token Taxonomy Framework

Input Specification File

The Blockchain App Builder initialization command reads the input specification file and generates the scaffolded project, which includes tools to assist in the chaincode development process.

In the specification file you can specify multiple asset definitions and behavior, CRUD and non-CRUD method declaration, custom methods, validation of arguments, auto marshalling/unmarshalling, transparent persistence capability, and the ability to complete rich data queries using SQL SELECT statements or the CouchDB query language. These features are generated for you.

For information on specifying token assets see the following topics:

- Input Specification File for Token Taxonomy Framework
- Input Specification File for ERC-721
- Input Specification File for ERC-1155

The specification file can be written in either YAML or JSON format. You can see sample specification files in both formats in the Blockchain App Builder package download:

- `Fabcar-Typescript.yml`

Copyright © 2025, Oracle and/or its affiliates.

- `Marbles-Go.yml`

> **Note**
>
> In accordance with Go conventions, exported names begin with a capital letter. Therefore all the asset properties and methods must have names starting with capital letters in the specification file.

Structure of the Specification File

Typically, you structure a specification file in the following way:

```
assets:
    name:
    type:
    properties:
        name:
        type:
        id:
        derived:
           strategy:
           algorithm:
           format:
        mandatory:
        default:
        validate:
    methods:
        crud:
        others:
customMethods:
```

Blockchain App Builder supports two special asset types, embedded assets and token assets, in addition to generic assets with no specified type. The special assets are defined as `type: embedded` or `type: token` under the `assets:` section of the specification file.

Table 2-2 Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`assets:`	This property takes the definition and behavior of the asset. You can define multiple assets.	

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`name:`	The name of the asset. The following names are reserved. Do not use these names for assets. • `account` • `role` • `hold` • `token` • `authorization` • `tokenAdmin` • `Account` • `Role` • `Hold` • `Token` • `Authorization` • `TokenAdmin`	`name: owner # Information about the owner`
`type:`	**Asset types** The following special asset types are supported: • `embedded` • `token` If you do not specify a `type` parameter in the `assets` section, the asset is of the generic type.	

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples

`type:`
type: embedded
	If this property is set to `embedded` the asset is defined as an embedded asset. Embedded assets do not have CRUD methods and must be part of another asset to be stored in the ledger.
	In the example, the property `address` is embedded, and is defined in another asset.
	Embedded assets do not support circular references. For instance, in the previous example the `address` asset cannot contain a reference to the `employee` asset.
		Asset: `employee`
		
		`name: employee`
		`properties:`
		`name: employeeId`
		`type: string`
		`mandatory: true`
		`id: true`
		
		`name: firstName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		
		`name: lastName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		
		`name: age`
		`type: number`
		`validate: positive(),min(18)`
		
		`name: address`
		`type: address`
		
		Asset: `address`
		
		`name: address`
		
		`type: embedded`
		
		`properties:`
		`name: street`
		`type: string`
		
		`name: city`
		`type: string`
		
		`name: state`
		`type: string`
		
		`name: country`
		`type: string`

`properties:`	Describes all of the properties of an asset.	

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`name:`	The name of the property.	`name: ownerId # Unique ID for each owner`

`id:`	• true
	This specifies the identifier of this asset. This property is mandatory.
		`name: owner # Information about the owner`
		`properties:`
		`  name: ownerId # Unique ID for each owner`
		`  type: string`
		`  mandatory: true`
		`  **id: true**`
		`  name: name # Name of the owner`
		`  type: string`
		`  mandatory: true`

`type:`	**Property types**
	The following basic property types are supported:
	• `number`
	• `float`
	• `string`
	• `boolean`
	• `date`
	• `array`
	For Go chaincodes, `number` is mapped to `int` and `float` is mapped to `float64`. Other types are not currently supported, including the following types:
	• `complex`
	• `unsigned/ signed int`
	• `8/16/32/64 bits`
		`name: year # Model year`
		`  **type: number**`
		`  mandatory: true`
		`  validate: min(1910),max(2020)`
		`  name: color # Color - no validation as color names are innumerable`
		`  **type: string**`
		`  mandatory: true`

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples

`derived:`	This property specifies that the id property is derived from other keys. Dependent properties must be of the `string` data type and not an embedded asset.
	This property has two mandatory parameters:
	• `strategy`: takes values of `concat` or `hash`.
	• `format`: takes an array of specification strings and values to be used by the strategy.
	Example 1:
	• The property `employeeID` is dependent on the `firstName` and `lastName` properties.
	• This property is a concatenation of the values listed in the `format` array.
	• `IND%1#%2%tIND` is the 0th index in the array and describes the final format.
	• %n is a position specifier that takes its values from the other indexes in the array.
	• `%t` indicates that the value is the `stub.timestamp` value from the channel header.
	• If you need to use the percent sign (%) in the `format` string, it must be escaped with another percent sign (%).
	• The final format in this example would
		Example 1
		`name: employee`
		`properties:`
		`name: employeeId`
		`type: string`
		`mandatory: true`
		`id: true`
		`derived:`
		`strategy: concat`
		`format:`
		`["IND%1#%2%tIND","firstName","lastName"]`
		`name: firstName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		`name: lastName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		`name: age`
		`type: number`
		`validate: positive(),min(18)`
		Example 2
		`name: account`
		`properties:`
		`name: accountId`
		`type: string`
		`mandatory: true`
		`id: true`
		`derived:`
		`strategy: hash`
		`algorithm: 'sha256'`
		`format: ["IND%1#%2%t","bankName","ifsccode"]`
		`name: bankName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		`name: ifsccode`
		`type: string`
		`mandatory: true`

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
	be: `INDfirstName#lastName1606885454916IND` Example 2: • When using `hash`, you must also use the `algorithm` parameter. The default is `sha256`; `md5` is also supported. • `IND%1#%2%t` is the 0th index in the array and describes the final format. • `%n` is a position specifier that takes its values from the other indexes in the array. • `%t` indicates that the value is the `stub.timestamp` value from the channel header. • If you need to use the percent sign (%) in the `format` string, it must be escaped with another percent sign (%).	

`mandatory:`	• true
	• false
	The corresponding property is mandatory and cannot be skipped while creating an asset.
		`name: phone # Phone number - validate as (ddd)-ddd-dddd where dashes could also be periods or spaces`
		`type: string`
		`mandatory: true`
		`validate: /^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$/`
		`name: cars # The list of car VINs owned by this owner`
		`type: string[]`
		`mandatory: false`

`default:`	The default value of this property.	

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`validate:`	The given property is validated against some of the ready-to-use validations provided by Blockchain App Builder. You can chain validations if you ensure that the chain is valid. If the `validate` property is not provided, then the validation is done against only the property `type`.	

`validate:`
type: number
	• positive()
	• negative()
	• min()
	• max()
	These validations can be chained together, separated by commas.
		`name: offerApplied`
		`type: number`
		`validate: negative(),min(-4)`
		
		`name: year  # Model year`
		`type: number`
		`mandatory: true`
		`validate: min(1910),max(2020)`

`validate:`
type: string
	• `min()`
	• `max()`
	• `email()`
	• `url()`
	• `/regex/` supports PHP regex
	For Go chaincodes, regular expressions which contain certain reserved characters or whitespace characters must be properly escaped.
		`name: website`
		`type: string`
		`mandatory: false`
		`validate: url()`
		
		`name: phone  # Phone number - validate as (ddd)-ddd-dddd where dashes could also be periods or spaces`
		`type: string`
		`mandatory: true`
		`validate: /^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$/`
		
		`name: Color #Color can be red, blue, or green`
		`type: string`
		`mandatory: true`
		`validate: /^\\s*(red|blue|green)\\s*$/`

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`validate:` *type: boolean*	• true • false In the example, the validation of property `active` is by the type itself (`boolean`)	`name: active` **`type: boolean`**
`validate:` *type: array*	By type itself, in the form of `type: number[]`, this conveys that the array is of type `number`. You can enter limits to the array in the format `number[1:5]`, which means that the minimum length is 1, and the maximum is 5. You can omit either one.	`name: items` **`type: number[:5]`**

`validate:`
type: date
	• `min()`
	• `max()`
	The date must be one of these formats:
	• `YYYY-MM-DD`
	• `YYYY-MM-DDTHH:MM:SSZ`, where T separates the date from the time, and the Z indicates UTC. Timezone offsets can replace the Z, as in -05:00 for Central Daylight Savings Time.
		`name: expiryDate`
		`type: date`
		`validate: max('2020-06-26')`
		
		`name: completionDate`
		`type: date`
		`validate: min('2020-06-26T02:30:55Z')`

`methods:`	Use this to state which of the CRUD (Create/Read/Update/Delete) or additional methods to generate.
	By default all CRUD and other methods are generated.
		`methods:`
		`  crud: [create, getById, update, delete]`
		`  others: [getHistoryById, getByRange]`

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
crud:	• create • getByID (read) • update • delete If this array is left empty, no CRUD methods will be created. If the crud parameter is not specified, all four methods will be created by default. The crud parameter is not applicable to token and embedded assets.	methods: **crud: [create, getById, delete]** others: [] # no other methods will be created
others:	• getHistoryById • getByRange getHistoryById returns the history of the asset in a list. getByRange returns all of the assets in a given range. For more information, see getByRange (TypeScript) and GetByRange (Go). If this array is left empty, no other methods will be created. If the others parameter is not used at all, both methods will be created by default. The others parameter is not applicable to token and embedded assets.	methods: crud: [create, delete] **others: [] # no other methods will be created** methods: crud: [create, getById, update, delete] **others: [getHistoryById, getByRange]**

Copyright © 2025, Oracle and/or its affiliates.

Table 2-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples

`customMethods:`	This property creates invokable custom method templates in the main controller file. It takes the method signature and creates the function declaration in the controller file.
	You can provide language-specific function declarations here.
	A custom method named `executeQuery` is provided. If you add this method to the specification file, it details how Berkeley DB SQL and CouchDB rich queries can be executed. This method can be calld only when you are connected to Oracle Blockchain Platform Cloud or Enterprise Edition.
		TypeScript
		`customMethods:`
		`- executeQuery`
		`- "buyCar(vin: string, buyerId: string, sellerId: string, price: number, date: Date)"`
		`- "addCar(vin: string, dealerId: string, price: number, date: Date)"`
		Go
		`customMethods:`
		`- executeQuery`
		`- "BuyCar(vin string, buyerId string, sellerId string, price int)"`
		`- "AddCar(vin string, dealerId string, price int)"`

Scaffolded TypeScript Chaincode Project

Blockchain App Builder takes the input from your specification file and generates a fully-functional scaffolded chaincode project. The project contains automatically generated classes and functions, CRUD methods, SDK methods, automatic validation of arguments, marshalling/un-marshalling and transparent persistence capability (ORM).

If the chaincode project uses the TypeScript language, the scaffolded project contains three main files:

- `main.ts`
- `<chaincodeName>.model.ts`
- `<chaincodeName>.controller.ts`

All of the necessary libraries are installed and packaged. The `tsconfig.json` file contains the necessary configuration to compile and build the TypeScript project.

The `<chaincodeName>.model.ts` file in the `model` subdirectory contains multiple asset definitions and the `<chaincodeName>.controller.ts` file in the `controller` subdirectory contains the assets behavior and CRUD methods.

The decorators in `model.ts` and `controller.ts` provide support for features such as automatic validation of arguments, marshalling/unmarshalling of arguments, transparent persistence capability (ORM), and calling rich queries.

Reference:

Copyright © 2025, Oracle and/or its affiliates.

- Models
- Decorators
- ORM
- SDK Methods
- Controller
- Automatically Generated Methods
- Controller Method Details
- Custom Methods
- Init Method

Models

Every model class extends the `OchainModel` class, which has an additional read-only property called `assetType`. This property can be used to fetch only assets of this type. Any changes to this property are ignored during the creation and updating of the asset. The property value by default is `<modelName>`.

The `OchainModel` class enforces decorator behaviors on properties of the class.

```
@Id('supplierId')
export class Supplier extends OchainModel<Supplier> {
    public readonly assetType = 'supplier';
    @Mandatory()
    @Validate(yup.string())
    public supplierId: string;
```

Decorators

Class decorators

`@Id(identifier)`

This decorator identifies the property which uniquely defines the underlying asset. This property is used as a key of the record, which represents this asset in the chaincode's state. This decorator is automatically applied when a new TypeScript project is scaffolded. The `identifier` argument of the decorator takes the value from specification file.

```
@Id('supplierId')
export class Supplier extends OchainModel{
...
}
```

Property decorators

Multiple property decorators can be used. The decorators are resolved in top to bottom order.

```
@Mandatory()
```


Copyright © 2025, Oracle and/or its affiliates.

This decorator marks the following property as mandatory, so that it cannot be skipped while saving to the ledger. If skipped, an error is thrown.

```
@Mandatory()
public supplierID: string;
```

```
@Default(param)
```

This decorator indicates that the following property has a default value. The default value in the argument (`param`) is used when the property is skipped while saving to the ledger.

```
@Default('open for business')
@Validate(yup.string())
public remarks: string;
```

```
@Validate(param)
```

The property following this decorator is validated against the schema specified by the parameter. The argument `param` takes a yup schema, and multiple schema methods can be chained together to add complex validations. For more information, see https://www.npmjs.com/package/yup.

```
@Validate(yup.number().min(3))
public productsShipped: number;
```

```
@ReadOnly(param)
```

This property decorator marks the underlying property as having a read-only value. The value in the argument, for example *param*, is used when the property is saved in the ledger. After the value is set it cannot be edited or removed.

```
@ReadOnly('digicur')
public token_name: string;
```

```
@Embedded(PropertyClass)
```

This property decorator marks the underlying property as an embeddable asset. It takes the embeddable class as a parameter. This class must extend the `EmbeddedModel` class. This requirement is validated by the decorator.

In the following example, the `Employee` class has a property called `address` of type `Address`, which will be embedded in the `Employee` asset. This is denoted by the `@Embedded()` decorator.

```
export class Employee extends OchainModel<Employee> {

   public readonly assetType = 'employee';

   @Mandatory()
   @Validate(yup.string())
   public emplyeeID: string;
```


Copyright © 2025, Oracle and/or its affiliates.

```
   @Mandatory()
   @Validate(yup.string().max(30))
   public firstName: string;

   @Mandatory()
   @Validate(yup.string().max(30))
   public lastName: string;

   @Validate(yup.number().positive().min(18))
   public age: number;

   @Embedded(Address)
   public address: Address;
}


export class Address extends EmbeddedModel<Address> {

   @Validate(yup.string())
   public street: string;

   @Validate(yup.string())
   public city: string;

   @Validate(yup.string())
   public state: string;

   @Validate(yup.string())
   public country: string;
}
```

When a new instance of the `Address` class is created, all the properties of the `Address` class are automatically validated by the `@Validate()` decorator. Note that the `Address` class does not have the `assetType` property or `@Id()` class decorator. This asset and its properties are not saved in the ledger separately but are saved along with the `Employee` asset. Embedded assets are user-defined classes that function as value types. The instance of this class can only be stored in the ledger as a part of the containing object (`OchainModel` assets). All of the above decorators are applied automatically based on the input file while scaffolding the project.

```
@Derived(STRATEGY, ALGORITHM, FORMAT)
```

This decorator is used for defining the attribute derived from other properties. This decorator has two mandatory parameters:

- `STRATEGY`: Takes values of `CONCAT` or `HASH`. If `HASH` is specified, the additional parameter `ALGORITHM` is required. The default algorithm is `sha256`; `md5` is also supported.
- `FORMAT`: Takes an array of specification strings and values to be used by the strategy.

```
@Id('supplierID')
export class Supplier extends OchainModel<Supplier> {

   public readonly assetType = 'supplier';
```


Copyright © 2025, Oracle and/or its affiliates.

```
    @Mandatory()
    @Derived(STRATEGY.HASH.'sha256',['IND%1IND%2','license','name'])
    @Validate(yup.string())
    public supplierID: string;

    @Validate(yup.string().min(2).max(4))
    public license: string;

    @Validate(yup.string().min(2).max(4))
    public name: string;
```

Method decorators

`@Validator(…params)`

This decorator is applied on methods of the main controller class. This decorator is important for parsing the arguments, validating against all the property decorators, and returning a model/type object. Controller methods must have this decorator to be invokable. It takes multiple user-created models or yup schemas as parameters.

The order of the parameters must be exactly the same as the order of the arguments in the method.

In the following example, the `Supplier` model reference is passed in the parameter that corresponds to the `asset` type in the method argument. At run time, the decorator parses and converts the method argument to a JSON object, validates against the `Supplier` validators, and after successful validation converts the JSON object to a `Supplier` object and assigns it to the `asset` variable. Then the underlying method is finally called.

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
    return await this.Ctx.Model.save(asset);
}
```

In the following example, multiple asset references are passed; they correspond to the object types of the method arguments. Note the order of the parameters.

```
@Validator(Supplier, Manufacturer)
public async createProducts(supplier: Supplier, manufacturer: Manufacturer) {
}
```

Apart from asset references, yup schema objects can also be passed if the arguments are of basic types. In the following example, `supplierId` and `rawMaterialSupply` are of type `string` and `number` respectively, so the yup schema of similar type and correct order is passed to the decorator. Note the chaining of yup schema methods.

```
@Validator(yup.string(), yup.number().positive())
public async fetchRawMaterial(supplierID: string, rawMaterialSupply: number) {
    const supplier = await this.Ctx.Model.get(supplierID, Supplier);
    supplier.rawMaterialAvailable = supplier.rawMaterialAvailable +
rawMaterialSupply;
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Model.update(supplier);
}
```

ORM

Transparent Persistence Capability or simplified ORM is captured in the `Model` class of the Context (`Ctx`) object. If your model calls any of the following SDK methods, access them by using `this.Ctx.Model`.

The following SDK methods implement ORM:

- `save` calls the Hyperledger Fabric `putState` method.
- `get` calls the Hyperledger Fabric `getState` method.
- `update` calls the Hyperledger Fabric `putState` method.
- `delete` calls the Hyperledger Fabric `deleteState` method.
- `history` calls the Hyperledger Fabric `getHistoryForKey` method.
- `getByRange` calls the Hyperledger Fabric `getStateByRange` method.
- `getByRangeWithPagination` calls the Hyperledger Fabric `getStateByRangeWithPagination` method.

For more information, see: SDK Methods.

SDK Methods

> **Note**
>
> Beginning with version 21.3.2, the way to access the ORM methods has changed. Run the `ochain --version` command to determine the version of Blockchain App Builder.

In previous releases, the ORM methods were inherited from the `OchainModel` class. In version 21.3.2 and later, the methods are defined on the `Model` class of Context (`Ctx`) object. To call these methods, access them by using `this.Ctx.Model.<method_name>`.

The following example shows a method call in previous releases:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier){
    return await asset.save();
}
```

The following example shows a method call from the version 21.3.2 and later:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
      return await this.Ctx.Model.save(asset);
}
```

After you upgrade to version 21.3.2, make this change in all chaincode projects that you created with an earlier version of Blockchain App Builder. If you use the `sync` command to synchronize changes between the specification file and your source code, the changes are

Copyright © 2025, Oracle and/or its affiliates.

automatically brought to your controller for the ready-to-use methods. You still need to manually resolve any conflicts.

save
The save method adds the caller asset details to the ledger.

This method calls the Hyperledger Fabric putState internally. All marshalling/unmarshalling is handled internally. The save method is part of the Model class, which you access by using the Ctx object.

```
Ctx.Model.save(asset: <Instance of Asset Class> , extraMetadata?: any) :
Promise <any>
```

Parameters:

- extraMetadata : any (Optional) – Used to save metadata apart from the asset in the ledger.

Returns:

- Promise<any> - Returns a promise on completion.

Example:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
    return await this.Ctx.Model.save(asset);
}
```

get
The get method is a method of OchainModel class, which is inherited by the concrete model classes of {chaincodeName}.model.ts. The get method is part of the Model class, which you access by using the Ctx object.

To return any asset from a given id, use the generic controller method getAssetById.

```
Ctx.Model.get(id: string, modelName: <Model Asset Class Name>) :
Promise<asset>
```

Parameters:

- id : string – Key used to save data into the ledger.
- modelName: <Model Asset Class Name> – (Optional) Model asset class to return.

Returns:

- Promise: <Asset> - If the modelName parameter is not provided and data exists in ledger, then Promise<object> is returned. If the id parameter does not exist in ledger, an error message is returned. If the modelName parameter is provided, then an object of type <Asset> is returned. Even though any asset with given id is returned from the ledger, this method will handle casting into the caller Asset type. If the asset returned from the ledger is not of the Asset type, then the method throws an error. This check is done by the read-only assetType property in the Model class.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
@Validator(yup.string())
public async getSupplierById(id: string) {
    const asset = await this.Ctx.Model.get(id, Supplier);
    return asset;
}
```

In the example, `asset` is of the type `Supplier`.

`update`
The `update` method updates the caller `asset` details in the ledger. This method returns a promise.

This method calls the Hyperledger Fabric `putState` internally. All the marshalling/ unmarshalling is handled internally. The `update` method is part of the `Model` class, which you can access by using the `Ctx` object.

```
Ctx.Model.update(asset: <Instance of Asset Class> , extraMetadata?: any) :
Promise <any>
```

Parameters:

- `extraMetadata : any` (Optional) – Used to save metadata apart from the asset in the ledger.

Returns:

- `Promise<any>` - Returns a promise on completion.

Example:

```
@Validator(Supplier)
public async updateSupplier(asset: Supplier) {
    return await this.Ctx.Model.update(asset);
}
```

`delete`
This deletes the asset from the ledger given by `id` if it exists. This method calls the Hyperledger Fabric `deleteState` method internally. The `delete` method is part of the `Model` class, which you can access by using the `Ctx` object.

```
Ctx.Model.delete(id: string): Promise <any>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise <any>` - Returns a promise on completion.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
@Validator(yup.string())
public async deleteSupplier(id: string) {
    const result = await this.Ctx.Model.delete(id);
    return result;
}
```

`history`
The `history` method is part of the `Model` class, which you can access by using the `Ctx` object. This method returns the asset history given by `id` from the ledger, if it exists.

This method calls the Hyperledger Fabric `getHistoryForKey` method internally.

```
Ctx.Model.history(id: string): Promise <any>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise <any[]>` - Returns any [] on completion.

Example

```
@Validator(yup.string())
public async getSupplierHistoryById(id: string) {
    const result = await this.Ctx.Model.history(id);
    return result;
}
```

Example of the returned asset history for `getSupplierHistoryById`:

```
[
    {
        "trxId":
"8ef4eae6389e9d592a475c47d7d9fe6253618ca3ae0bcf77b5de57be6d6c3829",
        "timeStamp": 1602568005,
        "isDelete": false,
        "value": {
            "assetType": "supplier",
            "supplierId": "s01",
            "rawMaterialAvailable": 10,
            "license": "abcdabcdabcd",
            "expiryDate": "2020-05-28T18:30:00.000Z",
            "active": true
        }
    },
    {
        "trxId":
"92c772ce41ab75aec2c05d17d7ca9238ce85c33795308296eabfd41ad34e1499",
        "timeStamp": 1602568147,
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "isDelete": false,
        "value": {
            "assetType": "supplier",
            "supplierId": "s01",
            "rawMaterialAvailable": 15,
            "license": "valid license",
            "expiryDate": "2020-05-28T18:30:00.000Z",
            "active": true
        }
    }
]
```

getByRange
The `getByRange` method is a static method of `OchainModel` class which is inherited by the concrete `Model` classes of `{chaincodeName}.model.ts`.

This method returns a list of assets between the range `startId` and `endId`. This method calls the Hyperledger Fabric `getStateByRange` method internally.

If the `modelName` parameter is not provided, the method returns `Promise<Object [ ] >`. If the `modelName` parameter is provided, then the method handles casting into the caller `Model` type. In the following example, the result array is of the type `Supplier`. If the asset returned from the ledger is not of the `Model` type, then it will not be included in the list. This check is done by the read-only `assetType` property in the `Model` class.

To return all the assets between the range `startId` and `endId`, use the generic controller method `getAssetsByRange`.

```
Ctx.Model.getByRange(startId: string, endId: string, modelName: <Asset Model
Class Name> ): Promise <any>
```

Parameters:

- `startId : string` – Starting key of the range, which is included in the range.
- `endId : string` – Ending key of the range, which is excluded from the range.
- `modelName: <Model Asset Class Name>` – (Optional) Model asset class to return.

Returns:

- `Promise< Asset[ ] >` - Returns array of `<Asset>` on completion.

Example:

```
@Validator(yup.string(), yup.string())
public async getSupplierByRange(startId: string, endId: string) {
    const result = await this.Ctx.Model.getByRange(startId, endId, Supplier);
    return result;
}
```

getByRangeWithPagination
The `getByRangeWithPagination` method is a static method of `OchainModel` class which is inherited by the concrete `Model` classes of `{chaincodeName}.model.ts`.

Copyright © 2025, Oracle and/or its affiliates.

This method returns a list of asset between the range `startId` and `endId`. This method calls the Hyperledger Fabric `getStateByRangeWithPagination` method internally.

If the `modelName` parameter is not provided, the method returns `Promise<Object [ ] >`. If the `modelName` parameter is provided, then the method handles casting into the caller `Model` type. In the following example, the result array is of the type `Supplier`. If the asset returned from the ledger is not of the `Model` type, then it will not be included in the list. This check is done by the read-only `assetType` property in the `Model` class.

To return all the assets between the range `startId` and `endId`, filtered by page size and bookmarks, use the generic controller method `getAssetsByRange`.

```
public async getByRangeWithPagination<T extends OchainModel<T>>(startId:
string, endId: string, pageSize: number, bookmark?: string, instance?: new
(data: any, skipMandatoryCheck: boolean, skipReadOnlyCheck: boolean) => T):
Promise<T[]>
```

Parameters:

- `startId : string` – Starting key of the range, which is included in the range.
- `endId : string` – Ending key of the range, which is excluded from the range.
- `pageSize : number` - The page size of the query.
- `bookmark : string` - The bookmark of the query. Output starts from this bookmark.
- `modelName: <Model Asset Class Name>` – (Optional) Model asset class to return.

Returns:

- `Promise< Asset[ ] >` - Returns array of `<Asset>` on completion.

`getId`
When the asset has a derived key as `Id`, you can use this method to get a derived ID. This method returns an error if the derived key contains `%t` (time stamp).

Parameters:

- `object` – The object must contain all of the properties on which the derived key is dependent.

Returns:

- Returns the derived key as a string.

Example:

```
@Validator(yup.string(), yup.string())

public async customGetterForSupplier(license: string, name: string){
    let object = {
      license : license,
      name: name
    }
    const id = await this.Ctx.Model.getID(object);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return this.Ctx.Model.get(id);
}
```

For token SDK methods, see the topics under Tokenization Support Using Blockchain App Builder.

Controller

The main controller class extends the `OchainController` class. There is only one main controller.

```
export class TSProjectController extends OchainController{
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable from outside; the rest of them are hidden.

Automatically Generated Methods

As described in Input Specification File, you can specify which CRUD methods to generate in the specification file. For example, if you specified to generate all methods, the result would be similar to the following code:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
    return await this.Ctx.Model.save(asset);
}

@Validator(yup.string())
public async getSupplierById(id: string) {
    const asset = await this.Ctx.Model.get(id, Supplier);
    return asset;
}

@Validator(Supplier)
public async updateSupplier(asset: Supplier) {
    return await this.Ctx.Model.update(asset);
}

@Validator(yup.string())
public async deleteSupplier(id: string) {
    const result = await this.Ctx.Model.delete(id);
    return result;
}

@Validator(yup.string())
public async getSupplierHistoryById(id: string) {
    const result = await this.Ctx.Model.history(id);
    return result;
}

@Validator(yup.string(), yup.string())
public async getSupplierByRange(startId: string, endId: string) {
    const result = await this.Ctx.Model.getByRange(startId, endId, Supplier);
    return result;
}
```


Copyright © 2025, Oracle and/or its affiliates.

Controller Method Details

Apart from the previous model CRUD and non-CRUD methods, Blockchain App Builder provides ready-to-use support from our controller for the following Hyperledger Fabric methods:

- `getAssetById`
- `getAssetsByRange`
- `getAssetHistoryById`
- `query`
- `queryWithPagination`
- `generateCompositeKey`
- `getByCompositeKey`
- `getTransactionId`
- `getTransactionTimestamp`
- `getChannelID`
- `getCreator`
- `getSignedProposal`
- `getArgs`
- `getStringArgs`
- `getMspID`
- `getNetworkStub`

> **Note**
>
> These methods are available with the `this` context in any class that extends the `OChainController` class.

For example:

```
public async getModelById(id: string) {
    const asset = await this.getAssetById(id);
    return asset;
}
@Validator(yup.string(), yup.string())
public async getModelsByRange(startId: string, endId: string) {
    const asset = await this.getAssetsByRange(startId, endId);
    return asset;
}
public async getModelHistoryById(id: string) {
    const result = await this.getAssetHistoryById(id);
    return result;
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getAssetById`
The `getAssetById` method returns an asset based on a specified ID. This is a generic method and can be used to get asset of any type.

```
this.getAssetById(id: string): Promise<byte[]>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise <byte [ ]>` - Returns promise on completion. You have to convert `byte[]` into an object.

`getAssetsByRange`
The `getAssetsByRange` method returns all assets present from `startId` (inclusive) to `endId` (exclusive) irrespective of asset types. This is a generic method and can be used to get assets of any type.

```
this.getAssetsByRange(startId: string, endId: string):
Promise<shim.Iterators.StateQueryIterator>
```

Parameters:

- `startId : string` – Starting key of the range, which is included in the range.
- `endId : string` – Ending key of the range, which is excluded from the range.

Returns:

- `Promise< shim.Iterators.StateQueryIterator>` - Returns an iterator on completion. You have to iterate over it.

`getAssetHistoryById`
The `getAssetHistoryById` method returns history iterator of an asset for a specified ID.

```
this.getAssetHistoryById(id: string):
Promise<shim.Iterators.HistoryQueryIterator>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise<shim.Iterators.HistoryQueryIterator>` - Returns a history query iterator. You have to iterate over it.

Copyright © 2025, Oracle and/or its affiliates.

`query`
The `query` method runs a rich SQL/Couch DB query over the ledger. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
this.query(queryStr: string):
Promise<shim.Iterators.StateQueryIterator>
```

Parameters:

- `queryStr : string` - Rich SQL/Couch DB query.

Returns:

- `Promise<shim.Iterators.StateQueryIterator>` - Returns a state query iterator. You have to iterate over it.

`queryWithPagination`
This method runs a rich SQL/Couch DB query over the ledger, filtered by page size and bookmarks. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
public async queryWithPagination(query: string, pageSize: number, bookmark?:
string)
```

Parameters:

- `query : string` - Rich SQL/Couch DB query.
- `pageSize : number` - The page size of the query.
- `bookmark : string` - The bookmark of the query. Output starts from this bookmark.

Returns:

- `Promise<shim.Iterators.StateQueryIterator>` - Returns a state query iterator. You have to iterate over it.

`generateCompositeKey`
This method generates and returns the composite key based on the `indexName` and the attributes given in the arguments.

```
this.generateCompositeKey(indexName: string, attributes:
string[]): string
```

Parameters:

- `indexName : string` - Object type of the key used to save data to the ledger.
- `attributes: string[ ]` - Attributes based on which a composite key will be formed.

Returns:

- `string` - Returns a composite key.

Copyright © 2025, Oracle and/or its affiliates.

`getByCompositeKey`
This method returns the asset that matches the key and the column given in the attribute parameter while creating a composite key. The `indexOfId` parameter indicates the index of the key returned in the array of the stub method `SplitCompositeKey`. Internally this method calls Hyperledger Fabric's `getStateByPartialCompositeKey`, `splitCompositeKey` and `getState` methods.

```
this.getByCompositeKey(key: string, columns: string[],
indexOfId: number): Promise<any []>
```

Parameters:

- `key: string` – Key used to save data to ledger.
- `columns: string[ ]` - Attributes based on which a key is generated.
- `indexOfId: number` - Index of attribute to be retrieved from key.

Returns:

- `Promise< any [ ]` - Returns any `[]` on completion.

`getTransactionId`
Returns the transaction ID for the current chaincode invocation request. The transaction ID uniquely identifies the transaction in the scope of the channel.

```
this.getTransactionId(): string
```

Parameters:

- none

Returns:

- `string` - Returns the transaction ID for the current chaincode invocation request.

`getTransactionTimestamp`
Returns the time stamp when the transaction was created. This is taken from the transaction `ChannelHeader`, therefore it will indicate the client's time stamp, and will have the same value across all endorsers.

```
this.getTransactionTimestamp(): Timestamp
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Timestamp` - Returns the time stamp when the transaction was created.

`getChannelID`
Returns the channel ID for the proposal for chaincode to process.

```
this.getChannelID(): string
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- none

Returns:

- `string` - Returns the channel ID.

`getCreator`
Returns the identity object of the chaincode invocation's submitter.

```
this.getCreator(): shim.SerializedIdentity
```

Parameters:

- none

Returns:

- `shim.SerializedIdentity` - Returns identity object.

`getSignedProposal`
Returns a fully decoded object of the signed transaction proposal.

```
this.getSignedProposal():
shim.ChaincodeProposal.SignedProposal
```

Parameters:

- none

Returns:

- `shim.ChaincodeProposal.SignedProposal` - Returns decoded object of the signed transaction proposal.

`getArgs`
Returns the arguments as array of strings from the chaincode invocation request.

```
this.getArgs(): string[]
```

Parameters:

- none

Returns:

- `string [ ]` - Returns arguments as array of strings from the chaincode invocation.

`getStringArgs`
Returns the arguments as array of strings from the chaincode invocation request.

```
this.getStringArgs(): string[]
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- none

Returns:

- `string [ ]` - Returns arguments as array of strings from the chaincode invocation.

`getMspID`
Returns the MSP ID of the invoking identity.

```
this.getMspID(): string
```

Parameters:

- none

Returns:

- `string` - Returns the MSP ID of the invoking identity.

`getNetworkStub`
You can access the shim stub by calling this method. This can help you write your own implementation of working directly with the assets.

```
this.getNetworkStub(): shim.ChaincodeStub
```

Parameters:

- none

Returns:

- `shim.ChaincodeStub` - Returns chaincode network stub.

`invokeCrossChaincode`
You can use this method in a chaincode to call a function in another chaincode. Both chaincodes must be installed on the same peer.

```
this.invokeCrossChaincode(chaincodeName: string, methodName: string, args:
string[], channelName: string): Promise<any>
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `Promise<any>` - Returns a JSON object that contains three fields:
  - `isValid` - `true` if the call is valid.

Copyright © 2025, Oracle and/or its affiliates.

- `payload` - The output returned by the cross-chaincode call, as a JSON object.
- `message` - The message returned by the cross-chaincode call, in UTF-8 format.

`invokeChaincode`
This method calls a function in another chaincode. Both chaincodes must be installed on the same peer.

```
this.invokeChaincode(chaincodeName: string, methodName: string, args:
string[], channelName: string): Promise<any>
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `Promise<any>` - Returns a JSON object that contains three fields:
  - `isValid` - `true` if the call is valid.
  - `payload` - The output returned by the cross-chaincode call, as a JSON object.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

Custom Methods

The following custom methods were generated from our example specification file.

The `executeQuery` method shows how SQL rich queries can be called. The validators against the arguments are added automatically by Blockchain App Builder based on the type of the argument specified in the specification file.

```
/**
*
*    BDB sql rich queries can be executed in OBP CS/EE.
*    This method can be invoked only when connected to remote OBP CS/EE
network.
*
*/
@Validator(yup.string()}
public async executeQuery(query: string) {
    const result = await OchainController.query(query);
    return result;
}
@Validator(yup.string(), yup.number()}
public async fetchRawMaterial(supplierId: string, rawMaterialSupply: number) {
}
@Validator(yup.string(), yup.string(), yup.number())
public async getRawMaterialFromSupplier(manufacturerId: string, supplierId:
string, rawMaterialSupply: number) {
}
@Validator(yup.string(), yup.number(), yup.number())
```


Copyright © 2025, Oracle and/or its affiliates.

```
public async createProducts(manufacturerId: string, rawMaterialConsumed:
number, productsCreated: number) {
}
public async sendProductsToDistribution() {
}
```

Init Method

A custom `init` method is provided in the controller with an empty definition. If you use Blockchain App Builder to deploy or upgrade, the `init` method is called automatically. If you deploy or upgrade from the Oracle Blockchain Platform console, you must call the `init` method manually. You can use a third-party tool such as Postman to call the `init` method manually.

```
export class TestTsProjectController extends OchainController {
    public async init(params: any) {
        return;
}
```

You can then use this method to initialize any application state.

Scaffolded Go Chaincode Project

Blockchain App Builder takes the input from your specification file and generates a fully-functional scaffolded chaincode project. The project contains automatically generated classes and functions, CRUD methods, SDK methods, automatic validation of arguments, marshalling/un-marshalling and transparent persistence capability (ORM).

If the chaincode project uses the Go language, the scaffolded project contains three main files:

- `main.go`
- `<chaincodeName>.model.go`
- `<chaincodeName>.controller.go`

All of the necessary libraries are installed and packaged.

The `<chaincodeName>.model.go` file in the `model` subdirectory contains multiple asset definitions and the `<chaincodeName>.controller.go` file in the `controller` subdirectory contains the asset's behavior and CRUD methods. The various Go struct tags and packages in `model.go` and `controller.go` provide support for features such as automatic validation of arguments, marshalling/unmarshalling of arguments, transparent persistence capability (ORM), and calling rich queries.

The scaffolded project is located in the `$GOPATH/src/example.com/<chaincodeName>` directory.

Reference:

- Model
- Validators
- ORM
- SDK Methods
- Composite Key Methods
- Stub Method

Copyright © 2025, Oracle and/or its affiliates.

- Other Methods
- Utility Package
- Controller
- Automatically Generated Methods
- Custom Methods
- Init Method

Model

Asset Type Property

By default every struct has an additional property called `AssetType`. You can use this property to fetch only assets of this type. Any changes to this property are ignored when assets are created or updated. The property value by default is `<modelName>`.

```
type Supplier struct {
AssetType string 'json:"AssetType" default:"TestGoProject.Supplier"'

SupplierId            string      'json:"SupplierId"
validate:"string,mandatory" id:"true'
RawMaterialAvailable  int         'json:"RawMaterialAvailable"
validate:"int,min=0"'
License               string      'json:"License" validate:"string,min=10"'
ExpiryDate            date.Date   'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
Active                bool      'json:"Active" validate:"bool" default:"true"'
Metadata              interface{} 'json:"Metadata,omitempty"'
}
```

Validators

Id
`id:"true"`

This validator identifies the property that uniquely defines the underlying asset. The asset is saved by the value in this key. This validator is automatically applied when a new Go project is scaffolded.

In the following example, `SupplierId` is the key for the supplier asset and has a tag property `id:"true"` for the `SupplierId` property.

```
type Supplier struct {
    Supplierld              string        'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable    int           'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License                 string        'json:"License"
validate:"string,min=10"'
    ExpiryDate              date.Date     'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                  bool          'json:"Active" validate:"bool"
default :"true"'
```


Copyright © 2025, Oracle and/or its affiliates.

```
    Metadata                interface{}   'json:"Metadata,omitempty"'
}
```

Derived

`derived:"strategy,algorithm,format"`

This decorator is used for defining the attribute derived from other properties. This decorator has two mandatory parameters:

- `strategy`: Takes values of `concat` or `hash`. If `hash` is specified, the additional parameter `algorithm` is required. The default algorithm is `sha256`; `md5` is also supported.
- `format`: Takes an array of specification strings and values to be used by the strategy.

```
type Supplier struct{
   AssetType string 'json:"AssetType" final:"chaincode1.Supplier"'
   SupplierId   string  'json:"SupplierId" validate:"string" id:"true"
mandatory:"true"
derived:"strategy=hash,algorith=sha256,format=IND%1%2,License,Name"'
   Name         string  'json:"Name" validate:"string,min=2,max=4"'
   License      string  'json:"License" validate:"string,min=2,max=4"'
}
```

Mandatory

`validate:"mandatory"`

This decorator marks the following property as mandatory, so that it cannot be skipped while saving to the ledger. If skipped, an error is thrown. In the following example, the `SupplierId` property has a `validate:"mandatory"` tag.

```
Type Supplier struct {
    Supplierld              string      'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable    int         'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License                 string      'json:"License"
validate:"string,min=10"'
    ExpiryDate              date.Date   'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                  bool        'json:"Active" validate:"bool"
default :"true"'
    Metadata                interface{} 'json:"Metadata,omitempty"'
}
```

Default

`default:"<param>"`

This decorator indicates that the following property has a default value. The value in the default tag is used when the property is skipped while saving to the ledger. In the following example, `Active` has a default value of `true`, specified as the tag `default:"true"`.

```
Type Supplier struct {
    Supplierld            string       'json:"Supplierld"
```


Copyright © 2025, Oracle and/or its affiliates.

```
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License               string       'json:"License"
validate:"string,min=10"'
    ExpiryDate            date.Date    'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                bool         'json:"Active" validate:"bool"
default :"true"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

Validate types
Basic Go types are validated for a property by defining a validate tag. The following validate tags are based on the basic types:

- string: `validate: "string"`
- date: `validate: "date"`
- number: `validate: "int"`
- boolean: `validate: "bool"`

Min validator
`validate:"min=<param>"`

You can use the `min` validator to set the minimum value for a property of type number or string.

For type int: In the following example, the `RawMaterialAvailable` property has a minimum value of 0. If a value less than 0 is applied to the `RawMaterialAvailable` property, an error will be returned.

For type string: The minimum validator checks the length of the string against the specified value. In the following example, the `License` property must be at least ten characters long.

```
Type Supplier struct {
    Supplierld            string       'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License               string       'json:"License"
validate:"string,min=10"'
    ExpiryDate            date.Date    'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                bool         'json:"Active" validate:"bool"
default :"true"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

Max validator
`validate:"max=<param>"`

You can use the max validator to set the maximum value for a property of type number and string.

Copyright © 2025, Oracle and/or its affiliates.

For type int: Similar to the min validator, if a value specified for a struct field is greater than the value provided in the validator, an error is returned.

For type string: Similar to the min validator, the max validator checks the length of the string against the given value. In the following example, the `Domain` property has a maximum value of 50, so if the `Domain` property has a string length of more than 50 characters, an error message will be returned.

```
type Retailer struct {
    Retailerld         string        'json:"Retailerld"
validate:"string,mandatory" id:"true"'
    ProductsOrdered    int           'json:"ProductsOrdered"
validate:"int,mandatory"'
    ProductsAvailable  int           'json:"ProductsAvailable" validate:"int"
default:"1"'
    ProductsSold       int           'json:"ProductsSold" validate:"int"'
    Remarks            string        'json:"Remarks" validate:"string"
default :"open for business"'
    Items              []int         'json:"Items"
validate:"array=int,range=l-5"'
    Domain             string        'json:"Domain"
validate:"url,min=30,max=50"'
    Metadata           interface{}   'json:"Metadata,omitempty"'
}
```

Date validators
Before validator:

```
validate:"before=<param>"
```

The before validator validates a property of type `date` to have a value less than the specified parameter.

In the following example, the `ExpiryDate` property must be before `"2020-06-26"` and if it is not, an error is returned.

```
Type Supplier struct {
    Supplierld            string       'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License               string       'json:"License"
validate:"string,min=10"'
    ExpiryDate            date.Date    'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                bool         'json:"Active" validate:"bool"
default :"true"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

After validator:

```
validate:"after=<param>"
```


Copyright © 2025, Oracle and/or its affiliates.

The after validator validates a property of type `date` to have a value greater than the specified parameter.

In the following example, the `CompletionDate` property must be after `"2020-06-26"`, and if not it will return an error.

```
Type Supplier struct {
    Manufacturerld        string       'json:"Manufacturerld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,max=8"'
    ProductsAvailable     int          'json:"ProductsAvailable"
validate:"int"'
    CompletionDate        date.Date    'json:"CompletionDate"
validate:"date,after=2020-06-26"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

URL validator

`validate:"url"`

The URL validator validates a property for URL strings.

In the following example, the `Domain` property must be a valid URL.

```
type Retailer struct {
    Retailerld         string        'json:"Retailerld"
validate:"string,mandatory" id:"true"'
    ProductsOrdered    int           'json:"ProductsOrdered"
validate:"int,mandatory"'
    ProductsAvailable  int           'json:"ProductsAvailable" validate:"int"
default:"1"'
    ProductsSold       int           'json:"ProductsSold" validate:"int"'
    Remarks            string        'json:"Remarks" validate:"string"
default :"open for business"'
    Items              []int         'json:"Items"
validate:"array=int,range=l-5"'
    Domain             string        'json:"Domain"
validate:"string,url,min=30,max=50"'
    Metadata           interface{}   'json:"Metadata,omitempty"'
}
```

Regexp validator

`validate:"regexp=<param>"`

The regexp validator validates a property with the specified regular expression.

In the following example, the `PhoneNumber` property will be validated for a mobile number in accordance with the regular expression.

```
type Customer struct {
Customerld      string       'json:"Customerld" validate:"string,mandatory"
id:"true"'
```


Copyright © 2025, Oracle and/or its affiliates.

```
Name            string       'json:"Name" validate:"string,mandatory"'
ProductsBought  int          'json:"ProductsBought" validate:"int"'
OfferApplied    int          'json:"OfferApplied" validate :"int,nax=0"'
PhoneNumber     string       'json:"PhoneNumber" validate:"string,regexp=A\(?
([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$"'
Received        bool         'json:"Received" validate:"bool"'
Metadata        interface{}  'json:"Metadata,omitempty"'
}
```

Multiple validators
Multiple validators can be applied a property.

In the following example, the `Domain` property has validation for a string, a URL, and minimum and maximum string length.

```
type Retailer struct {
    Retailerld         string        'json:"Retailerld"
validate:"string,mandatory" id:"true"'
    ProductsOrdered    int           'json:"ProductsOrdered"
validate:"int,mandatory"'
    ProductsAvailable  int           'json:"ProductsAvailable" validate:"int"
default:"1"'
    ProductsSold       int           'json:"ProductsSold" validate:"int"'
    Remarks            string        'json:"Remarks" validate:"string"
default :"open for business"'
    Items              []int         'json:"Items"
validate:"array=int,range=l-5"'
    Domain             string        'json:"Domain"
validate:"string,url,min=30,max=50"'
    Metadata           interface{}   'json:"Metadata,omitempty"'
}
```

ORM

Transparent Persistence Capability or simplified ORM is captured in the `Model` class of the Context (`Ctx`) object. If your model calls any of the following SDK methods, access them by using `t.Ctx.Model`.

The following SDK methods implement ORM:

- `Save` calls the Hyperledger Fabric `PutState` method.
- `Get` calls the Hyperledger Fabric `GetState` method.
- `Update` calls the Hyperledger Fabric `PutState` method.
- `Delete` calls the Hyperledger Fabric `DeleteState` method.
- `History` calls the Hyperledger Fabric `GetHistoryForKey` method.
- `GetByRange` calls the Hyperledger Fabric `GetStateByRange` method.
- `GetByRangeWithPagination` calls the Hyperledger Fabric `GetStateByRangeWithPagination` method.

SDK Methods

Go chaincodes implement Transparent Persistence Capability (ORM) with the model package.

Copyright © 2025, Oracle and/or its affiliates.

> **Note**
>
> Beginning with version 21.2.3, the way to access the ORM methods has changed. Run the `ochain --version` command to determine the version of Blockchain App Builder.

In previous releases, the ORM methods were exposed as static methods in the model package. The methods are now defined on the model receiver, which holds the transaction stub. To call these methods, you use the model receiver held by the transaction context in the controller. You call these methods as `t.Ctx.Model.<method_name>` instead of `model.<method_name>`.

The following example shows `Save` and `Get` method calls in previous releases:

```
func (t *Controller) CreateSupplier(asset Supplier) (interface{}, error) {
  return model.Save(&asset)
}

func (t *Controller) GetSupplierById(id string) (Supplier, error) {
  var asset Supplier
  _, err := model.Get(id, &asset)
  return asset, err
}
```

The following example shows `Save` and `Get` method calls from the version 21.2.3 and later:

```
func (t *Controller) CreateSupplier(asset Supplier) (interface{}, error) {
  return t.Ctx.Model.Save(&asset)
}

func (t *Controller) GetSupplierById(id string) (Supplier, error) {
  var asset Supplier
  _, err := t.Ctx.Model.Get(id, &asset)
  return asset, err
}
```

After you upgrade to version 21.2.3, make this change in all chaincode projects that you created with an earlier version of Blockchain App Builder. If you use the `sync` command to synchronize changes between the specification file and your source code, the changes are automatically brought to your controller for the ready-to-use methods. You must still manually resolve any conflicts.

The following ORM methods are exposed by the model package:

`Get`
Queries the ledger for the stored asset based on the given ID.

```
func Get(Id string, result ...interface{}) (interface{}, error)
```

Parameters:

- `Id` - The ID of the asset to retrieve from the ledger.

Copyright © 2025, Oracle and/or its affiliates.

- `result (interface{})` - An empty asset object of a particular type, which is passed by reference. This object will contain the result from this method. Use this parameter only if a type-specific result is required.
- `asset (interface)` - An empty asset object, which is passed by reference. This object will contain the result from this method. Use this parameter only if a type-specific result is required.

Returns:

- `interface {}` - The interface contains the asset in `map[string]interface{}` form. Before operating on this map, you must assert the returned interface with type `map[string]interface{}`. To convert this map into an asset object, you can use the utility API `util.ConvertMaptoStruct` (see: Utility Package).
- `error` - Contains an error if returned, or is nil.

`Update`
Updates the specified asset in the ledger with new values.

```
func Update(args ...interface{}) (interface{}, error)
```

Parameters:

- `obj (interface)` - The object to update in the ledger is passed by reference into this API along with the new values. The input asset is validated and verified according to the struct tags in the model specification and then stored into the ledger.

Returns:

- `interface{}` - The saved asset is returned as an interface.
- `error` - Contains an error if returned, or is nil.

`Save`
Saves the asset to the ledger after validating on all the struct tags.

```
func Save(args ...interface{}) (interface{}, error)
```

Parameters:

- `obj/args[0] (interface{})` - The object to store in the ledger is passed by reference in this utility method.
- `metadata/args[1] (interface{})` - (Optional) You can use this parameter to store metadata to the ledger along with the asset at runtime.

Returns:

- `interface {}` - The asset is returned as an interface.
- `error` - Contains an error if returned, or is nil.

`Delete`
Deletes the asset from the ledger.

```
func Delete(Id string) (interface{}, error)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `id (string)` - The ID of the asset to delete from the ledger.

Returns:

- `interface {}` - Contains the asset being deleted in `map[string]interface{}` form.

`GetByRange`
Returns a list of assets specified by a range of IDs.

```
func GetByRange(startKey string, endKey string, asset ...interface{})
([]map[string]interface{}, error)
```

Parameters:

- `startkey (string)` - Starting ID for the range of objects to retrieve.
- `endkey (string)` - End ID for the range of objects to retrieve.
- `asset interface` - (Optional) An empty array of assets, passed by reference. This array will contain the method result. Use this parameter if a type-specific result is required.

Returns:

- `[]map[string]interface{}` - This array contains the list of assets obtained from the ledger. You can access the objects by iterating over this array and asserting the objects as `map[string]interface{}` and using a utility to convert to an asset object.
- `error` - Contains an error if returned, or is nil.

`GetByRangeWithPagination`
The `GetByRangeWithPagination` method is a static method of `OchainModel` class which is inherited by the concrete `Model` classes of `{chaincodeName}.model.ts`.

This method returns a list of assets between the range `startId` and `endId`, filtered by page size and bookmark. This method calls the Hyperledger Fabric `GetStateByRangeWithPagination` method internally.

If the `modelName` parameter is not provided, the method returns `Promise<Object [ ] >`. If the `modelName` parameter is provided, then the method handles casting into the caller `Model` type. In the following example, the result array is of the type `Supplier`. If the asset returned from the ledger is not of the `Model` type, then it will not be included in the list. This check is done by the read-only `assetType` property in the `Model` class.

To return all the assets between the range `startId` and `endId`, filtered by page size and bookmarks, use the generic controller method `getAssetsByRange`.

```
func (m *Model) GetByRangeWithPagination(startKey string, endKey string,
pageSize int32, bookmark string, asset ...interface{})
([]map[string]interface{}, error)
```

Parameters:

- `startkey : string` – Starting key of the range, which is included in the range.
- `endkey : string` – Ending key of the range, which is excluded from the range.

Copyright © 2025, Oracle and/or its affiliates.

- `pageSize : number` – The page size of the query.
- `Bookmark : string` – The bookmark of the query. Output starts from this bookmark.
- `asset interface` – (Optional) An empty array of assets, passed by reference. This array will contain the result from this method. Use this parameter to get type-specific results.

Returns:

- `[]map[string]interface{}` – An array that contains the list of assets retrieved from the ledger. You can access the objects by iterating over this array and asserting the objects as `map[string]interface{}` and using a utility for conversion to an asset object.
- `error` – Contains an error if an error is returned, otherwise nil.

`GetHistoryById`
Returns the history of the asset with the given ID.

```
func GetHistoryByID(Id string) ([]interface{}, error)
```

Parameters:

- `Id (string)` - ID of the asset.

Returns:

- `[]interface{}` - This slice contains the history of the asset obtained from the ledger in form of slice of `map[string]interface{}`. You can access each history element by iterating over this slice and asserting the objects as `map[string]interface{}` and using a utility to convert to asset object.
- `error` - Contains the error if an error is returned.

`Query`
This method runs a SQL/Couch DB query over the ledger. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
func Query(queryString string) ([]interface{}, error)
```

Parameters:

- `queryString (string)` - The query string.

Returns:

- `[]interface{}` - The query output in the form of a slice of interfaces. Iterate over the slice and use the elements by converting them to their proper types.
- `error` - Contains the error if an error is returned.

Copyright © 2025, Oracle and/or its affiliates.

`QueryWithPagination`
This method runs a SQL/Couch DB query over the ledger, filtered by page size and bookmark. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
func (m *Model) QueryWithPagination(queryString string, pageSize int32,
bookmark string) ([]interface{}, error)
```

Parameters:

- `queryString (string)` - Rich SQL/Couch DB query.
- `pageSize : number` - The page size of the query.
- `bookmark : string` - The bookmark of the query. Output starts from this bookmark.

Returns:

- `[]interface{}` - The query output in the form of a slice of interfaces. Iterate over the slice and use the elements by converting them to their proper types.
- `error` - Contains the error if an error is returned.

`InvokeCrossChaincode`
You can use this method in a chaincode to call a function in another chaincode. Both chaincodes must be installed on the same peer.

```
func InvokeCrossChaincode(chaincodeName string, method string, args []string,
channelName string) (interface{}, error)
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `interface{}` - Returns a `map[string]interface{}` object that contains three keys:
  - `isValid` - `true` if the call is valid.
  - `payload` - The output returned by the cross-chaincode call, as a JSON object.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

Return Value Example:

```
{
      "isValid": true,
      "message": "Successfully invoked method [CreateAccount] on sub-
chaincode [erc721_go_453]",
      "payload": {
                  "AccountId":
"oaccount~6b83b8ab931f99442897dd04cd7a2a55f808686f49052a40334afe3753fda4c4",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "AssetType": "oaccount",
                "BapAccountVersion": 0,
                "NoOfNfts": 0,
                "OrgId": "appdev",
                "TokenType": "nonfungible",
                "UserId": "user2"
        }
}
```

`InvokeChaincode`
You can use this method in a chaincode to call a function in another chaincode. Both chaincodes must be installed on the same peer.

```
func InvokeChaincode(chaincodeName string, method string, args []string,
channelName string) (interface{}, error)
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `interface{}` - Returns a `map[string]interface{}` object that contains three keys:
  - `isValid` - `true` if the call is valid.
  - `payload` - The output returned by the cross-chaincode call, in UTF-8 format.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

Return Value Example:

```
{
    "isValid": true,
    "message": "Successfully invoked method [CreateAccount] on sub-chaincode
[erc721_go_453]",
    "payload":
"{\"AssetType\":\"oaccount\",\"AccountId\":\"oaccount~c6bd7f8dcc339bf7144ea2e1
cf953f8c1df2f28482b87ad7895ac29e7613a58f\",\"UserId\":\"user1\",\"OrgId\":\"ap
pdev\",\"TokenType\":\"nonfungible\",\"NoOfNfts\":0,\"BapAccountVersion\":0}"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Composite Key Methods

`GenerateCompositeKey`
This method generates and returns the composite key based on the `indexName` and the attributes given in the arguments.

```
func GenerateCompositeKey(indexName string, attributes []string)
            (string, error)
```

Parameters:

- `indexName (string)` - Object type of the composite key.
- `attrbutes ([]string)` - Attributes of the asset based on which the composite key will be formed.

Returns:

- `string` - The composite key result.
- `error` - Contains the error if an error is returned.

`GetByCompositeKey`
This method returns the asset that matches the specified key and column. The `index` parameter indicates the index of the key returned in the array of stub method `SplitCompositeKey`.

Internally this method calls Hyperledger Fabric's `getStateByPartialCompositeKey`, `splitCompositeKey` and `getState` methods.

```
func GetByCompositeKey(key string, columns []string, index int)
                (interface{}, error)
```

Parameters:

- `key (string)` - Object type provided while creating composite key.
- `column ([]string)` - The slice of attributes on which the ledger will be queried using the composite key.
- `index(int)` - Index of the attribute.

Returns:

- `Interface{}` - The list of matching assets.
- `error` - Contains the error if an error is returned.

Stub Method

`GetNetworkStub`
This method returns the Hyperledger Fabric `chaincodeStub` value.

Copyright © 2025, Oracle and/or its affiliates.

You can access the shim stub by calling the `GetNetworkStub` method. This can assist you in writing your own implementation that works directly with assets.

```
func GetNetworkStub() shim.ChaincodeStubInterface
```

Parameters:

- none

Returns:

- `shim.ChaincodeStubInterface` - The Hyperledger Fabric chaincode stub.

Other Methods

- `GetTransactionId()`
- `GetTransactionTimestamp()`
- `GetChannelID()`
- `GetCreator()`
- `GetSignedProposal()`
- `GetArgs()`
- `GetStringArgs()`
- `GetCreatorMspId()`
- `GetId`

`GetTransactionId`
This method returns the transaction ID for the current chaincode invocation request. The transaction ID uniquely identifies the transaction in the scope of the channel.

```
func GetTransactionId() string
```

Parameters:

- none

Returns:

- `string` - The transaction ID.

`GetTransactionTimestamp`
Returns the time stamp when the transaction was created. Because the value is taken from the transaction `ChannelHeader`, it indicates the client's time stamp and has the same value across all endorsers.

```
func GetTransactionTimestamp() (*timestamp.Timestamp, error)
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- `timestamp.Timestamp` - The time stamp.
- `error` - Contains the error if an error is returned.

`GetChannelID`
Returns the channel ID for the proposal for the chaincode to process.

```
func GetChannelID() string
```

Parameters:

- none

Returns:

- `string` - The requested channel ID in string format.

`GetCreator`
Returns the identity object of the chaincode invocation's submitter.

```
func GetCreator() ([]byte, error)
```

Parameters:

- none

Returns:

- `[]byte` - The required identity object in serialized form.
- `error` - Contains the error if an error is returned.

`GetSignedProposal`
Returns a fully decoded object of the signed transaction proposal.

```
func GetSignedProposal() (*peer.SignedProposal, error)
```

Parameters:

- none

Returns:

- `*peer.SignedProposal` - The signed proposal object.
- `error` - Contains the error if an error is returned.

`GetArgs`
Returns the arguments as an array of strings from the chaincode invocation request.

```
func GetArgs() [][]byte
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- `[][]byte` - The passed arguments.

`GetStringArgs`
Returns the arguments intended for the chaincode `Init` and `Invoke` methods as a string array.

```
func GetStringArgs() []string
```

Parameters:

- none

Returns:

- `[]string` - The arguments as a string array.

`GetCreatorMspId`
Returns the MSP ID of the invoking identity.

```
func GetCreatorMspId() string
```

Parameters:

- none

Returns:

- `string` - The MSP ID of the invoking identity.

`GetId`
When the asset has a derived key as `Id`, you can use this method to get a derived ID. This method returns an error if the derived key contains `%t` (time stamp).

Parameters:

- `object` - The object must contain all of the properties on which the derived key is dependent.

Returns:

- Returns the derived key as a string.

Example:

```
func (t *Controller) CustomGetterForSupplier(License string, Name string)
(interface{}, error){
   var asset Supplier
   asset.License = License
   asset.Name = Name
   id,err := t.Ctx.Model.GetId(&asset)

   if err !=nil {
      return nil, fmt.Errorf("error in getting ID %v", err.Error())
   }
   return t.GetSupplierById(id)
}
```


Copyright © 2025, Oracle and/or its affiliates.

Utility Package

The following methods are provided in the utility package.

`Util.CreateModel`
Parses the specified JSON string and creates an asset object of the specified type.

```
func CreateModel(obj interface{}, inputString string) error
```

Parameters:

- `inputString (string)` - The input JSON string to use to create the object.
- `obj (interface{})` - The reference of the object to create from the JSON string. This object will store the created model, which is also validated in accordance with validator tags.

Returns:

- `error` - Contains the error if an error is returned while creating or validating the asset.

`util.ConvertMapToStruct`
Convert the specified map into an object of the specified type.

```
func ConvertMapToStruct(inputMap map[string](interface{}), resultStruct
interface{}) error
```

Parameters:

- `inputMap (map[string](interface{}))` - The map to convert into an asset object.
- `resultStruct (interface{})` - The reference of the asset object to create from the map.

Returns:

- `error` - Contains the error if an error is returned while creating or validating the asset.

For token SDK methods, see the topics under Tokenization Support Using Blockchain App Builder.

Controller

The `Controller.go` file implements the CRUD and custom methods for the assets.

You can create any number of classes, functions, or files, but only those methods that are defined on chaincode struct are invokable from outside; the rest of them are hidden.

Automatically Generated Methods

As described in Input Specification File, you can specify which CRUD methods you want generated in the specification file. For example, if you chose to generate all methods, the result would look similar to the following code:

```
//
//Supplier
//
func (t *ChainCode) CreateSupplier(inputString string) (interface{}, error) {
    var obj Supplier
```


Copyright © 2025, Oracle and/or its affiliates.

```
    err := util.CreateModel(&obj, inputString)
    if err != nil {
        return nil, err
    }
    return model.Save(&obj)
}

func (t *ChainCode) GetSupplierById(id string) (interface{}, error) {
    asset, err := model.Get(id)
    return asset, err
}

func (t *ChainCode) UpdateSupplier(inputString string) (interface{}, error) {
    var obj Supplier
    err := util.CreateModel(&obj, inputstring)
    if err != nil {
        return nil, err
    }
return model.Update(&obj)
}

func (t *ChainCode) DeleteSupplier(id string) (interface{}, error) {
    return model.Delete(id)
}

func (t *ChainCode) GetSupplierHistoryById(id string) (interface{}, error) {
    historyArray, err := model.GetHistoryByld(id)
    return historyArray, err
}

func (t *ChainCode) GetSupplierByRange(startkey string, endKey string)
(interface{}, error) {
    assetArray, err := model.GetByRange(startkey, endKey)
    return assetArray, err
}
```

Custom Methods

The following custom methods were generated from the example specification file.

The `executeQuery` function shows how SQL rich queries can be called. The validators against the arguments are added automatically by Blockchain App Builder based on the type of the argument specified in the specification file.

You can implement functionality according to the necessary business logic. If you add custom methods, add them to the controller file. If you add custom methods to the library instead of the controller file, your changes will be lost when the library folder contents are updated during the synchronization or chaincode upgrade processes.

```
//
//Custom Methods
//
/*
*    BDB sql rich queries can be executed in OBP CS/EE.
*    This method can be invoked only when connected to remote OBP CS/EE
network.
```


Copyright © 2025, Oracle and/or its affiliates.

```
*/
func (t *ChainCode) ExecuteQuery(inputQuery string) (interface{}, error) {
    resultArray, err := model.Query(inputQuery)
    return resultArray, err
}

func (t *ChainCode) FetchRawMaterial(supplierId string, rawMaterialSupply
int) (interface{}, error) {
    return nil, nil
}

func (t *ChainCode) GetRawMaterialFromSupplier(manufacturerId string,
supplierId string, rawMaterialSupply int) (interface{} error) {
    return nil, nil
}

Func (t *ChainCode) CreateProducts(manufacturerId string, rawMaterialConsumed
int, productsCreated int) (interface{}, error) {
    return nil, nil
}

func (t *ChainCode) SendProductsToDistribution() (interface{}, error) {
    return nil, nil
    }
```

For Go chaincodes, every custom method must return two values: an *empty interface* and an *error*, as shown in the following example:

```
func (t *Controller) FetchRawMaterial(supplierId string, rawMaterialSupply
int) (interface{}, error) {
    return nil, nil
}
```

Init Method

A custom `Init` method is provided in the controller with an empty definition. If you use Blockchain App Builder to deploy or upgrade, the `Init` method is called automatically. If you deploy or upgrade from the Oracle Blockchain Platform console, you must call the `Init` method manually. You can use a third-party tool such as Postman to call the `Init` method manually.

```
type Controller struct {
}
func (t *Controller) Init(args string) (interface{}, error)
    { return nil, nil
}
```

You can then use this method to initialize any application state.

Deploy Your Chaincode Using the CLI

After you create a chaincode project, you can deploy it locally to the automatically generated Hyperledger Fabric network or remotely to your Oracle Blockchain Platform Cloud or

Copyright © 2025, Oracle and/or its affiliates.

Enterprise Edition. You can also package the chaincode project for manual deployment to Oracle Blockchain Platform.

Deploy Your Chaincode to a Local Hyperledger Fabric Network

After you create your chaincode project, you can deploy it to a local Hyperledger Fabric network. This single-channel test network is created for you when you install Blockchain App Builder.

The Blockchain App Builder `ochain run` command starts the Hyperledger Fabric network, other services, and installs and deploys the chaincode for you.

```
ochain run -h
Usage: run [options] [...args]
Run chaincode project locally in debug mode.

Arguments :
[...args] (optional) Chaincode instantiate arguments. Arguments should be
space separated.

Options:
-h, --help        output command usage information
-D, --debug       enable debug logging
-P, --debug-port  (optional) specify debug process port
-b, --build       (optional) rebuild runtime if already exists
-p, --project     (optional) Path to Chaincode project to run. If not
specified, it defaults to current directory.

Examples :
$> ochain run
```

The test network that is created when you install Blockchain App Builder runs Hyperledger Fabric v2.5.7 and Hyperledger Fabric CA v1.5.9.

Token Projects

For token chaincode projects, you must specify a list of token administrators with the `ochain run` command:

```
ochain run <adminList array>
```

The `adminList` array is an array of information that specifies the token administrators. The `adminList` array is a mandatory parameter when you deploy a token chaincode project for the first time. If you deploy the project again, you can pass an empty array for the `adminList` parameter or you can use the `adminList` parameter to add token administrators. Other deployers who are not the first-time deployer must supply an empty array for the `adminList` parameter. The parameter information is different for fungible tokens and non-fungible tokens:

- For fungible tokens that use the Token Taxonomy Framework standard, the parameters are `org_id` and `user_id`.
- For any tokens that use the ERC-1155 standard, the parameters are `orgId` and `userId`.
- For non-fungible tokens that use the ERC-721 standard and a TypeScript project, the parameters are `orgId` and `userId`.

Copyright © 2025, Oracle and/or its affiliates.

- For non-fungible tokens that use the ERC-721 standard and a Go project, the parameters are `OrgId` and `UserId`.

The following examples are for non-fungible tokens.

Example `adminList` array for TypeScript on macOS and Linux:

```
'[{"userId":"userid", "orgId":"OrgMSPId"}]'
```

Example `adminList` array for Go on macOS and Linux:

```
'[{"UserId":"userid", "OrgId":"OrgMSPId"}]'
```

Example `adminList` array for TypeScript on Microsoft Windows:

```
"[{\"userId\":\"userid\", \"orgId\":\"OrgMSPId\"}]"
```

Example `adminList` array for Go on Microsoft Windows:

```
"[{\"UserId\":\"userid\", \"OrgId\":\"OrgMSPId\"}]"
```

For the local Hyperledger Fabric network, the `OrgMSPId` field is fixed to the value `Org1MSP`.

To see the debug logs, pass the `--debug` option to the command. On Windows, use Command Prompt instead of PowerShell if you specify the `--debug` option. You can run the basic network and deploy the chaincode on a different port from the default by passing the `--port` option to the command.

Verifying

The following logs show that the chaincode has been installed and deployed successfully.

```
my-mac:TSProject myname$ ochain run
Recreating orderer.example.com ... done
Recreating ca.example.com ... done
Recreating peer0.org1.example.com ... done
[2020-09-23T18: 04:09.132] [INFO] default -
============ Started Install Chaincode ============
[2020-09-23T18:04:09.193] [INFO] default Chaincode TSProject:l not installed.
[2020-09-23T18:04:09.317] [INFO] default - Successfully sent install Proposal
and received ProposalResponse
[2020-09-23T18:04:09.317] [INFO] default - Successfully installed chaincode
TSProject
[2020-09-23T18:04:09.317] [INFO] default -
============ Finished Install Chaincode ============
[2020-09-23T18:04:09.317] [INFO] default - Successfully installed chaincode
TSProject
[2020-09-23T18:04:09.318] [INFO] default -
============ started instantiate Chaincode ============
[2020-09-23T18:04:09.366] [INFO] default - Successfully sent Proposal and
received ProposalResponse
[2020-09-23T18:04:11.434] [INFO] default - The chaincode instantiate
transaction has been committed on peer localhost:7051
[2020-09-23T18:04:11.434] [INFO] default - The chaincode instantiate
```


Copyright © 2025, Oracle and/or its affiliates.

```
transaction was valid.
[2020-09-23T18:04:11.435] [INFO] default - Successfully sent transaction to
the orderer.
[2020-09-23T18:04:11.435] [INFO] default - Successfully instantiated
chaincode TSProject on channel mychannel
[2020-09-23T18:04:11.435] [INFO] default -
============ Finished instantiate Chaincode ============
[2020-09-23T18:04:11.4351 INFO] default - Successfully instantiated chaincode
TSProject on channel mychannel
INFO (Runtime): Chaincode TSProject installed and ready:
INFO (RunCommand): Chaincode TSProject deployed
```

Troubleshooting

You might encounter the following issues when running your chaincode project on a local network.

Missing Go permissions

While installing Go chaincode project in local network, you might see an error similar to the following text:

```
Starting ca.example.com    ... done
Starting orderer.example.com ... done
Starting peer0.orgl.example.com ... done
INFO (Runtime): 2020/06/22 22:57:09 build started

INFO (Runtime): Building ....

INFO (Runtime): go build runtime/cgo: copying /Users/myname/Library/Caches/go-
build/f8/.….….d: open /usr/local/go/pkg/darwin_amd64/runtine/
cgo.a: permission denied

ERROR (Runtime): go build runtine/cgo: copying /Users/myname/Library/
Caches/go-build/f8/.….….d: open /usr/local/go/pkg/darwin_amd64/runtime/
cgo.a: permission denied

INFO (Runtime): An error occurred while building: exit status 1

Stopping peer0.orgl.exmple.com ... done
Stopping ca.example.com    ... done
Stopping orderer.example.con ... done
```

This is due to missing permissions for Go. This error has been seen only in Mac OS. This is a known issue.

- https://github.com/golang/go/issues/37962
- https://github.com/golang/go/issues/24674
- https://github.com/udhos/update-golang/issues/15

Copyright © 2025, Oracle and/or its affiliates.

Solution: Use the following command to change the permissions of your `$GOROOT` directory, and then retry the `ochain run` command:

```
sudo chmod -R 777 /usr/local/go
```

Deployment failure
Due to deployment failure, corrupt deployment, a full Docker peer container, or a Docker peer that was stopped in the local network, you might see an error similar to the following text:

```
============ Started instantiate Chaincode ============
[2028-19-01T19:25:lO.372] [ERROR] default - Error instantiating Chaincode
GollGl on channel mychannel, detailed
error: Error: error starting container: error starting container: Failed to
generate platform-specific docker
build: Failed to pull hyperledger/fabric-ccenv:latest : API error (404):
manifest for hyperledger/
fabric-ccenv:latest not found: manifest unknown: manifest unknown
[2020-19-01T19:25:10.372] (INFO) default -
============ Finished instantiate Chaincode ============
[2020-19-01119:25:10.372] [ERROR] default - Error: Error instantiating
Chaincode Goll01 on channel mychannel,
detailed error: Error: error starting container: error starting container:
Failed to generate platfom-specific
docker build: Failed to pull hyperledger/fabric-ccenv: latest : API error
(404): manifest for hyperledger/
fabric-ccenv:lalest not found: manifest unknown: manifest unknown exited:
signal: terminated
INFO: exited: signal: terminated

ERROR: Error in Chaincode deployment
```

This is due to a peer container that is not able to start up properly again.

Solution: Try the `ochain run` command again, but with the `-b` option. This option rebuilds the runtime for you.

```
ochain run -b
```

Environment Rebuild Required
Rebuild your environment if you see a `channel not found` error or an error similar to the following text:

```
Starting ca.example.com ...
Starting orderer.example.com ...
Starting orderer.example.com ... error
ERROR: for orderer.example.com
Cannot start service orderer.example.com:
error while creating mount source path '/host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0/node_modules/@oracle/ochain-cli/
runtime/network/basic-network/config': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
Starting ca.example.com... error
```


Copyright © 2025, Oracle and/or its affiliates.

```
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
ERROR: for orderer.example.com
Cannot start service orderer.example.com: error while creating mount source
path '/host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/config': mkdir /
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0:
operation not permitted
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
Encountered errors while bringing up the project.
ERROR: Starting ca.example.com ...
Starting orderer.example.com ...
Starting orderer.example.com ... error
ERROR: for orderer.example.com
Cannot start service orderer.example.com: error while creating mount source
path '/host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/config': mkdir /
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0:
operation not permitted
Starting ca.example.com ... error
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
ERROR: for orderer.example.com
Cannot start service orderer.example.com: error while creating mount source
path '/host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/config': mkdir /
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0:
operation not permitted
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
Encountered errors while bringing up the project.
ERROR: Error in Chaincode deployment
```

To rebuild your local environment, run the following command:

```
ochain run -b
```


Copyright © 2025, Oracle and/or its affiliates.

Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network

After you've deployed and tested your chaincode project on a local network to ensure it's working as designed, you can deploy it to Oracle Blockchain Platform.

Deploy Your Chaincode

Usage: `ochain deploy [options] [...args]`

The following arguments and options can be used with the `ochain deploy` command:

```
ochain deploy -h
Usage: deploy [options] [...args]
Deploy chaincode project to Oracle Blockchain Platform

Arguments:
     [...args]    (optional) Chaincode instantiate arguments. Arguments
should be space separated.

Options :
    -h, --help                      output command usage information
    -D, --debug                     enable debug logging
    -P, --project <project>         (optional) Path to Chaincode project to
deploy. If not specified, it defaults to current directory.
    -c, --channel <channel>         (optional) Blockchain Channel to deploy
chaincode to. If not specified, it defaults to the 'default' channel.
    -u, --username <username>       (required) A user name that has install
chaincode privileges. Contact your administrator for more details.
    -v, --userversion <userversion> (optional) A user-specified chaincode
version.
                            If a version isn't specified, for a new
chaincode it will start at v1 and then increment to v2, v3, and so on.
                            For an existing chaincode, v1.a will increment
to v1.a1, v1 will increment to v2, and v1.0 will increment to v1.1.
    -s --sign <password>            (required) Password to authenticate user.
    -r, --url <url>                 (required) Restproxy Url of the remote
OBP instance.
    For example,
    OBP Cloud Services: https://test-xyz-
abc.blockchain.ocp.oraclecloud.com:7443/restproxy
    OBP Enterprise Edition: https://restproxy.test-xyz-
abc.blockchain.platform.com:443/restproxy

Examples:
$> ochain deploy -u <username> -s <password> -r <REST proxy URL of the remote
instance> -c <name of the channel>
```

Enter the Oracle Identity Cloud Service user name and password for an Oracle Blockchain Platform user with the `ADMIN` or `REST_CLIENT` roles. For more information about users and roles, see *Set Up Users and Application Roles* in *Administering Oracle Blockchain Platform*.

To call the chaincode, only the `REST_CLIENT` role is necessary. To deploy or upgrade the chaincode, the Identity Cloud Service user must also be assigned the `ADMIN` role.

Copyright © 2025, Oracle and/or its affiliates.

After the chaincode has been successfully deployed to the remote Oracle Blockchain Platform, the log will show that the following events occurred:

- Oracle Blockchain Platform details were successfully fetched.
- The list of peers was successfully fetched.
- The chaincode project was successfully installed.
- The chaincode project was successfully approved and committed.
- The chaincode was successfully deployed on each peer and the channel.

In an environment with multiple organizations, to deploy the chaincode again on the same channel through a participant instance, use the console to deploy the chaincode.

Upgrading the Chaincode Project

Upgrading the chaincode is handled automatically by Blockchain App Builder. After you make changes to your chaincode, call the `ochain deploy` command again, which will automatically upgrade the project for you. When you run the `ochain deploy` command again, specify an empty array for the `adminList` parameter or use the `adminList` parameter to add token administrators. If you are not the first-time deployer, you must supply an empty array for the `adminList` parameter.

If your upgrade is successful, the log will show that the following events occurred:

- Oracle Blockchain Platform details were successfully fetched.
- The list of peers was successfully fetched.
- A check was made to determine if the chaincode project is already installed, and if so, the version was fetched.
- The chaincode version was successfully upgraded (for example, from version 1.0 to 2.0).

In an environment with multiple organizations, to upgrade the chaincode, use the console and manually approve the chaincode from the participants.

Package Your Chaincode Project for Manual Deployment to Oracle Blockchain Platform

You can package your chaincode projects for manual deployment to Oracle Blockchain Platform Cloud or Enterprise Edition.

Usage: `ochain package`

The `package` command creates an archive file that contains only the build and distribution files. The `binary`, `libs`, `node_modules`, and `test` folders from your chaincode project are not included. This archive file can be manually uploaded to Oracle Blockchain Platform for deployment.

Because of changes to software prerequisites, when you run the `ochain package` command for TypeScript chaincode, you are prompted for the provisioning date of the Oracle Blockchain Platform instance that you want to create the package for. The TypeScript chaincode created in Blockchain App Builder is not compatible with previous versions of Oracle Blockchain Platform without changes to the underlying infrastructure. Blockchain App Builder packages the chaincode infrastructure accordingly based on your selection.

```
ochain package -h
Usage: package [options]
```


Copyright © 2025, Oracle and/or its affiliates.

```
Package and archive an Ochain chaincode project
Options :
    -h, --help             output command usage information
    -D, --debug            enable debug logging
    -p, --project <path>   Path to the Ochain chaincode project to be
packaged. If not specified, it defaults to current directory.
    -o, --out <path>       Path to the generated chaincode archive file. If
not specified, it defaults to current directory.
About:
This CLI command packages and archives an existing chaincode project
Examples:
$> ochain package --project <Path to the Ochain chaicode project> –out <Path
to the generated chaincode archive file>
```

When the command completes successfully, the location of the package is returned.

This command takes two optional arguments:

- `--project`
  This option defines the location of the Blockchain App Builder chaincode project to package. If not specified, the location defaults to the current directory.
- `--out`
  This option can be used to give the output path of the generated archive file. If not specified, it defaults to the current directory.

Example:

```
ochain package -p /Blockchain/DevTools/bp1/CC -o /Blockchain/DevTools/bp1/
output

"Your chaincode project has been packaged at /Blockchain/DevTools/bp1/output/
CC.zip"
```

Test Your Chaincode Using the CLI

If your chaincode is running on a network, you can test any of the generated methods. Additionally, if you chose to create the `executeQuery` method during your chaincode development, you can run SQL rich queries if your chaincode is deployed to an Oracle Blockchain Platform network.

Test Your Chaincode on a Local Hyperledger Fabric Network

After your chaincode project is running on a local network, you can test it.

Open a new shell and navigate to the project directory to interact with your chaincodes. After a chaincode is installed and deployed, you can submit transactions to the functions inside your chaincode by using the `ochain invoke` and `ochain query` commands.

`ochain invoke`

Usage: `ochain invoke <methodName> <methodArguments>`

Copyright © 2025, Oracle and/or its affiliates.

The following arguments and options are available for the `ochain invoke` command:

```
ochain invoke -h
Usage: invoke [options] <methodName> [...args]
Invoke a Chaincode transaction

Arguments :
    <methodName>  (required) Name of chaincode method to invoke.
    [...args]     (optional) Chaincode method input parameters if any.
Parameters should be space separated strings/JSON strings for objects.

Options:
    -h, --help                output command usage information
    -D, --debug               enable debug logging
    -P, --project <path>      (optional) Path to Chaincode project to deploy.
If not specified, it defaults to current directory.
    -c, --channel <channel>   (optional) Blockchain Channel to deploy
chaincode to. If not specified, it defaults to the 'default' channel.
    -u, --username <username> (optional, if -r option is applied) A user name
that has install chaincode privileges. Contact your administrator for more
details.
    -s, --sign <password>     (optional) Password to authenticate user.
    -r, --url <url>           (required) Restproxy Url of the remote OBP
instance.
    For example,
    OBP Cloud Services: https://test-xyz-
abc.blockchain.ocp.oraclecloud.com:7443/restproxy
    OBP Enterprise Edition: https://restproxy.test-xyz-
abc.blockchain.platform.com:443/restproxy

Examples:
$> ochain invoke <method>
(without chaincode initial arguments)
$> ochain invoke <method>
'{"manufacturerId":"m01","rawMaterialAvailable":9,"productsAvailable":4,"compl
etionDate":"05-26-2020"}'
(for a single parameter)
$> ochain invoke <method> 's01' 's10'
$> ochain invoke <method> 's01'
'{"manufacturerId":"m01","rawMaterialAvailable":9,"productsAvailable":4,"compl
etionDate":"05-26-2020"}'
(for multiple parameters)
$> ochain invoke <method> 's01' 's10' -r <REST proxy URL of the remote
instance> -u <username> -s <password>
(for remote invocation)
```

Linux and macOS

Copyright © 2025, Oracle and/or its affiliates.

If a method takes one argument, enter it as a string, as shown in the following examples:

```
ochain invoke createSupplier
'{"supplierId":"s01","rawMaterialAvailable":5,"license":"valid
supplier","expiryDate":"2020-05-30","active":true}'


ochain invoke getSupplierDetails 's01'
'{"supplierId":"s01","rawMaterialAvailable":5,"license":"valid
supplier","expiryDate":"2020-05-30","active":true}'
```

If a method takes more than one argument, the arguments must be separated by a space. For example:

```
ochain invoke getSupplierByRange 's01' 's03'
```

You must use a different format to specify embedded assets in your chaincode, such as an employee asset which uses an embedded address asset, as shown in the following examples:

```
name: employee
  properties:
     name: employeeId
     type: string
     mandatory: true
     id: true

     name: firstName
     type: string
     validate: max(30)
     mandatory: true

     name: lastName
     type: string
     validate: max(30)
     mandatory: true

     name: age
     type: number
     validate: positive(),min(18)

     name: address
     type: address


name: address

type: embedded

properties:
   name: street
   type: string

   name: city
   type: string
```


Copyright © 2025, Oracle and/or its affiliates.

```
   name: state
   type: string

   name: country
   type: string
```

For these embedded assets, you can use a format similar to the following to call the chaincode:

```
ochain invoke createEmployee '{"employeeID":"e01", "firstName":"John",
"lastName":"Doe",
"age":35, "address":{"street":"Elm Ave", "city":"LA", "state":"California",
"country":"US"}}'
```

Microsoft Windows

The Windows command prompt doesn't accept single quotation marks ('), so all arguments must be specified in double quotation marks ("). Any argument that contains a double quotation mark must be escaped, as shown in the following example.

```
ochain invoke createSupplier
"{\"supplierId\":\"s01\",\"rawMaterialAvailable\":5,\"license\":\"valid
supplier\",\"expiryDate\":\"2020-05-30\",\"active\":true}"
```

If the method takes more than one argument, they must be separated by a space. For example:

```
ochain invoke getSupplierByRange "s01" "s03"
```

If you have embedded assets in your chaincode such as an employee asset which uses an embedded address asset as shown previously, you can use a format similar to the following to call the chaincode:

```
ochain invoke createEmployee "{\"employeeID\":\"e01\",
\"firstName\":\"John\",
\"lastName\":\"Doe\", \"age\":35, \"address\":{\"street\":\"Elm Ave\",
\"city\":\"LA\",
\"state\":\"California\", \"country\":\"US\"}}"
```

Validations

The method arguments are validated against the validations specified in the specification file. If any validation fails, errors will be listed in the output.

When the chaincode is called successfully, it will display a log similar to the following text:

```
========== Started Invoke Chaincode ==========
[2020-06-23T18:37:54.563] [INFO] default - Successfully sent Proposal and
received ProposalResponse
[2020-06-23T18:37:56.619] [INFO default - The chaincode invoke transaction
has been committed on peer localhost:7051
[2020-06-23T18:37:56.619] [INFO] default - The chaincode invoke transaction
was valid.
[2020-06-23T18:37:56.620] [INFO default - Successfully sent transaction to
```


Copyright © 2025, Oracle and/or its affiliates.

```
the orderer.
[2020-06-23T18:37:56.620] [INFO] default - Successfully invoked method
"createSupplier" on chaincode "TSProject" on channel "mychannel"
[2020-06-23T18:37:56.620] [INFO] default -
========== Finished Invoke Chaincode ==========
```

The `ochain query` Command

Usage: `ochain query <methodName> <methodArguments>`

The following arguments and options are available for the `ochain query` command:

```
ochain query -h
Usage: query [options] <methodName> [...args]
Invoke a Chaincode Query.

Arguments :
    <methodName>     (required) Name of chaincode method to invoke.
    [...args]        (optional) Chaincode method input parameters if any.
Parameters should be space separated strings/JSON strings for objects.

Options:
    -h, --help               output command usage information
    -D, --debug              enable debug logging
    -P, --project <path>     (optional) Path to Chaincode project to deploy.
If not specified, it defaults to current directory.
    -c, --channel <channel>  (optional) Blockchain Channel to deploy
chaincode to. If not specified, it defaults to the 'default' channel.
    -u, --username <username> (optional, if -r option is applied) A user name
that has install chaincode privileges. Contact your administrator for more
details.
    -s, --sign <password>    (optional) Password to authenticate user.
    -r, --url <url>          (required) Restproxy Url of the remote OBP
instance.
    For example,
    OBP Cloud Services: https://test-xyz-
abc.blockchain.ocp.oraclecloud.com:7443/restproxy
    OBP Enterprise Edition: https://restproxy.test-xyz-
abc.blockchain.platform.com:443/restproxy

Examples:
$> ochain query <method>
(without chaincode initial arguments)
$> ochain query <method> s01
(for a single parameter)
$> ochain query <method> 's01' 's10'
$> ochain query <method> 's01' '{"manufacturerId":"m01"}'
(for multiple parameters)
$> ochain query <method> 's01' 's10' -r <REST proxy URL of the remote
instance> -u <username> -s <password>
(for remote query)
```

The `ochain query` command follows the same rules of passing `<methodName>` and `<methodArguments>` as the `ochain invoke` command.

Copyright © 2025, Oracle and/or its affiliates.

- On macOS and Linux, single quotation marks can be used and there's no need to escape quotation marks within arguments.
- On Microsoft Windows, all arguments must surrounded by double quotation marks and any quotation mark within an argument must be escaped.

Testing Multiple Token Users Locally

To test a token project with multiple users locally, you can use the `tokenUser` property to change the caller of each transaction. Every scaffolded chaincode project includes a `.ochain.json` file, which stores metadata of the chaincode. You change the caller by updating the value of `tokenUser` field in the `.ochain.json` file.

```
{
  "name": "digiCurrCC",
  "description": "Chaincode package for digiCurrCC",
  "chaincodeName": "digiCurrCC",
  "chaincodeType": "node",
  "configFileLocation": "/Users/user1/token.yml",
  "appBuilderVersion": "21.2.3",
  "nodeVersion": "v12.18.1",
  "tokenUser": "admin"
}
```

When a project is scaffolded, the `tokenUser` property is set to the default `admin` user of the local network. To change the caller of a transaction, change the `tokenUser` property to match the `user_id` property that was set when the account was created when the `createAccount` (TypeScript) or `CreateAccount` (Go) method was called.

Automatic Installation and Deployment After Update

Whenever you update your chaincode, the changes will be compiled, installed and deployed automatically when it's deployed to a local network. There is no need to take down or bring up the local network again. All projects will be automatically compiled and deployed on every change.

Test Your Chaincode on a Remote Oracle Blockchain Platform Network

After your chaincode project has successfully deployed to your remote Oracle Blockchain Platform network, you can test it.

You test chaincode running on a remote network similar to testing it on a local network, as described in Test Your Chaincode on a Local Hyperledger Fabric Network.

If you deployed your chaincode manually, instead of using Blockchain App Builder, you must call the `init` function manually before you test your chaincode.

You can use the same `ochain invoke` and `ochain query` commands to run all method transactions on a remote Oracle Blockchain Platform Cloud or Enterprise Edition network; everything supported on the local network is also supported on the remote network. Pass the REST proxy URL of the remote Oracle Blockchain Platform instance (`-r`), user name (`-u`) and password (`-s`) options to the command. Make sure to specify the correct REST proxy port, which is typically either 7443 or 443.

Copyright © 2025, Oracle and/or its affiliates.

Example

```
ochain invoke createSupplier
'{"supplierId":"s01","rawMaterialAvailable":5,"license":"valid
supplier","expiryDate":"2020-05-30","active":true}' -r
[https://%3cblockchain-restproxy-url%3e:7443/]https://<blockchain-proxy-
url>:7443/
-u idcqa -s password
```

Testing Token Projects on a Remote Oracle Blockchain Platform Network

You can test chaincode projects that work with tokens by using Blockchain App Builder, the Oracle Blockchain Platform REST proxy, or the Hyperledger Fabric SDK.

Blockchain App Builder

You can use the Blockchain App Builder CLI to complete transactions with multiple users to test token chaincodes.

To test with multiple users, change the authorization parameters (user name and password options) in the invoke and query commands.

Oracle Blockchain Platform REST Proxy

You can use the REST proxy in Oracle Blockchain Platform to run your token chaincode on a remote Oracle Blockchain Platform network. Use any REST proxy client, such as Postman REST Client, to test your chaincode project.

To test multiple users, change the authorization parameters (user name and password) in your REST client, or connect to a different instance of Oracle Blockchain Platform.

Execute Berkeley DB SQL Rich Queries

If you chose to create the `executeQuery` method during your chaincode development, you can run SQL rich queries if your chaincode is deployed to an Oracle Blockchain Platform network.

If you specified `executeQuery` in the `customMethods` section of the specification file, a corresponding `executeQuery` method will be created in the controller.

Specification file:

```
customMethods:
    - executeQuery
    - "fetchRawMaterial(supplierid: string, rawMaterialSupply: number)"
    - "getRawMaterialFromSupplier(manufacturerId: string, supplierld: string,
rawMaterialSupply: number)"
    - "createProducts(manufacturerId: string, rawMaterialConsumed: number,
productsCreated: number)"
    - "sendProductsToDistribution()"
```

Controller file:

```
**
*
* BDB sql rich queries can be executed in OBP CS/EE.
* This method can be invoked only when connected to remote OBP CS/EE network.
*
```


Copyright © 2025, Oracle and/or its affiliates.

```
*/
@Validator(yup.string())
public async executeQuery(query: string) {
    const result = await this.query(query);
    return result;
    }
```

You can call this method to run Berkeley DB SQL rich queries on Oracle Blockchain Platform network by using the `-r`, `-u` and `-s` options to specify the remote Oracle Blockchain Platform REST proxy URL, user name and password respectively.

Example query for TypeScript:

```
ochain query  executeQuery "SELECT key, valueJson FROM <STATE> WHERE
json_extract(valueJson, '$.rawMaterialAvailable') = 4" -r
[https://%3cblockchain-restproxy-url%3e:7443/]https://<blockchain-restproxy-
url>:7443/
-u idcqa -s password
```

Example query for Go:

```
ochain query  executeQuery "SELECT key, valueJson FROM <STATE> WHERE
json_extract(valueJson, \"$.rawMaterialAvailable\") = 4" -r
[https://%3cblockchain-restproxy-url%3e:7443/]https://<blockchain-restproxy-
url>:7443/
-u idcqa -s password
```

The entire SQL query is taken in the argument, so you can make changes to your query on the fly. Make sure to specify the correct port for the REST proxy, which is typically either 7443 or 443. The syntax is different for TypeScript and Go chaincode. As shown in the previous example, the Go query uses double quotation marks for the query parameters instead of single quotation marks. The double quotation marks must be preceded by backslash characters.

Upgrading Chaincode Projects in the CLI

You can use the upgrade command to upgrade existing chaincode projects to use new features when Blockchain App Builder is updated. This command works with both TypeScript and Go projects.

For Go projects, upgrade to a supported version of Go before you run the command to upgrade your chaincode project. Blockchain App Builder supports Go v1.23.2 or later, but not Go v1.24 or later.

Usage:

```
upgrade [options]
```

```
ochain upgrade -h
Usage: upgrade [options]
Upgrade App Builder chaincode project
Options :
    -h, --help              output command usage information
    -D, --debug             enable debug logging
    -p, --project <path>    (optional) Path to Chaincode project to upgrade.
If not specified, it defaults to current directory.
```


Copyright © 2025, Oracle and/or its affiliates.

```
    -cc, --chaincode        (optional) To upgrade chaincode project
Examples :
$> ochain upgrade --project <path of chaincode dir>
```

To upgrade a chaincode project, navigate to the directory containing the project and then enter the following command:

```
ochain upgrade
```

If you encounter problems while upgrading a chaincode project, you can use the `--debug` option to generate debug logs. On Microsoft Windows, use Command Prompt instead of PowerShell if you specify the `--debug` option.

After you upgrade a chaincode project, synchronize the specification file with the generated source code. For more information, see Synchronize Specification File Changes With Generated Source Code.

Synchronize Specification File Changes With Generated Source Code

You can use the `ochain sync` command to bring new changes from the specification file to the current source files (model and controller). The command works with both TypeScript and Go projects.

Note:

- Synchronization is unidirectional: you can bring changes from your specification file into your chaincode project, but not the other way around. Changes made in your chaincode project remain as-is after the synchronizing process.
- The `ochain sync` command works only if the chaincode project was scaffolded by using a specification file. Do not delete, rename or move the specification file if you plan to synchronize any changes from the specification file to the source code in the future.
- If you used a single specification file to generate more than one chaincode project, you can synchronize only one project at a time by using the `ochain sync` command.

Usage:

```
sync [options] [...args]

ochain sync -h
Usage: sync [options] [...args]
Synchronize Changes from spec file to the required chaincode.
Arguments:
    [...args] (optional) Sync Arguments.
Options :
    -h, --help              output command usage information
    -D, --debug             enable debug logging
    -p, --project <path>    (optional) Path to Chaincode project to sync. If
not specified, it defaults to current directory
    -c, --confirm <bool>    (optional) Parameter to ensure if you have
resolved all the conflicts, and commit changes
Examples :
```


Copyright © 2025, Oracle and/or its affiliates.

```
$> ochain sync
without chaincode initial arguments
```

The `ochain sync` command has two optional arguments:

- `-p / --project`
  This option takes the chaincode project directory to synchronize. If not specified, it defaults to the current directory.
- `-c / --confirm`
  This option takes Boolean (true/false) values. If there are any conflicts during the merging process, you must resolve those conflicts manually and set this option to true in the next synchronization cycle. Don't use this option if you're not sure that the conflicts have been merged.

If the `ochain sync` command fails and you have installed and configured all prerequisites including Git and the Git user name and password, complete the following steps to reinitialize the synchronization operation:

1. Check the specification file for errors and fix any errors that you find.
2. In the chaincode project folder, remove the following folders if they are present: `.sync_temp`, `.sync_backup`, and `.sync_repo`.
3. In the `src` folder, remove the `.git` folder if it is present.
4. In the project root folder, edit the `ochain.json` file and set the `syncEnabled` property to `false`. Save and close the file.
5. Retry the `ochain sync` command.

Apply a Patch to the Blockchain App Builder CLI

You can use the `patch` command to apply a patch to the Blockchain App Builder command-line interface (CLI).

Usage:

```
patch [options]

ochain patch -h
Usage: patch [options]

Apply PatchFix to ochain

Options :
    -h, --help              output command usage information
    -D, --debug             enable debug logging
    -p, --path <path>       Path to Patch Zip file
```

To apply a patch to a project, navigate to the directory that contains the project and then enter the following command. You must pass the path to the archive file that contains the patch as an option to the `patch` command.

```
ochain patch --path path_to_archive_file
```


Copyright © 2025, Oracle and/or its affiliates.

Writing Unit Test Cases and Coverage Reports for the Chaincode Project

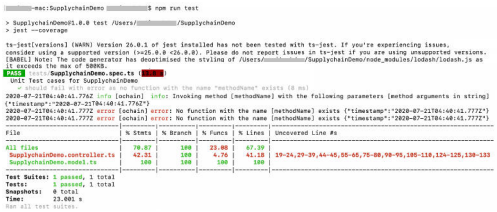
Blockchain App Builder includes support for writing unit test cases and coverage reports for the generated chaincode projects.

> **Note**
>
> If you're using Visual Studio Code, you can run unit test cases in the Terminal window.

TypeScript

To write unit test cases for a TypeScript chaincode, refer to the file `<chaincodeName>.spec.ts` in the `tests` folder inside the generated chaincode project. This file provides the complete unit testing setup for TypeScript chaincodes and an example unit test case in the comments section for reference. You can follow the example to write unit test cases for your chaincode methods.

To run the unit test cases, enter the command `npm run test` from the chaincode project folder. This command also generates coverage reports.



Go

To write unit test cases for a Go chaincode, refer to the file `src/src_test.go` inside the generated chaincode project. This file provides the complete unit testing setup for Go chaincodes and an example unit test case in the comments section for reference. You can following the example to write unit test cases for your chaincode methods.

To run the unit test cases, enter the command `go test` from the chaincode project folder. For coverage reports, add the flag `--cover`.

```
go test --cover
```

Generate a Postman Collection Using the CLI

You can use the `generate` command to create a Postman collection that includes example payloads for all of your chaincode controller APIs.

Postman is a tool that you can use to work with and test REST APIs. The generate command creates a Postman collection that is based on the chaincode that was automatically generated from a declarative specification file. The Postman collection contains the payloads for all of the

Copyright © 2025, Oracle and/or its affiliates.

methods that are specified in the chaincode controller file. You can change the variable values in the Postman collection file to make REST API calls.

The generated Postman collection includes default values for all APIs in the controller. To learn more about Postman, see https://www.postman.com/. After you generate a Postman collection, you can directly import it and use it by changing the default values in the payload and variables.

Usage:

```
generate [options]

ochain generate -h
Usage: generate [options]

Generates the postman collection for the chaincode.

Options :
    -h, --help              output command usage information
    -D, --debug             enable debug logging
    -c, --collection        This option is mandatory to generate a Postman
collection.
    -p, --project <path>    Path to the chaincode project to generate the
Postman collection from. If not specified, it defaults to current directory.
    -o, --out <path>        Path to the generated Postman collection JSON
file. If not specified, it defaults to current directory.
```

To generate a Postman collection, navigate to the directory that contains the project and then enter the following command. You must run the `generate` command from the chaincode directory or an error will occur. If the specified Postman collection already exists, you are prompted whether to overwrite it.

```
ochain generate --collection --project path_to_chaincode_project --out
path_to_postman_collection_to_generate
```

Postman Collection Structure

The generated Postman collection includes two types of requests, invoke requests and query requests:

- Invoke requests include all write operations, which use the endpoint `/transactions`
- Query requests include all get operations, which use the endpoint `/chaincode-queries`

To differentiate between getter and non-getter methods in the controller APIs, a decorator is used in TypeScript chaincodes and a comment is used in Go chaincodes. If you define a getter method in the controller, you must use the GetMethod decorator for TypeScript or the GetMethod comment for Go, as shown in the following table.

Copyright © 2025, Oracle and/or its affiliates.

TypeScript	Go

Every getter method has a `GetMethod` decorator:
 	
`@GetMethod()`
`@Validator()`
`public async getAllTokenAdmins() {`
`  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ADMIN.getAllAdmins", "TOKEN");`
`  return await this.Ctx.ERC1155Admin.getAllAdmins();`
`}`
	Every getter method has a `GetMethod` comment block:
	
	`/**`
	` * GetMethod`
	` */`
	`func (t *Controller) GetAllTokenAdmins() (interface{}, error) {`
	`  auth, err := t.Ctx.Auth.CheckAuthorization("Admin.GetAllAdmins", "TOKEN")`
	`  if err != nil && !auth {`
	`  return nil, fmt.Errorf("error in authorizing the caller  %s", err.Error())`
	`  }`
	`  return t.Ctx.Admin.GetAllTokenAdmins()`
	`}`

Generated Postman collections include variables with default values, as shown in the following table.

Variable Name	Description	Default Value	Context
`bc-url`	The REST proxy URL of the Oracle Blockchain Platform instance where the chaincode is deployed	`https://test-xyz-abc.blockchain.ocp.oraclecloud.com:7443/restproxy`	all chaincodes
`bc-channel`	The channel where the chaincode is deployed	`default`	all chaincodes
`bc-admin-user`	The name of the administrator (a user with the `admin` role that can access all POST requests). By default, this user is the caller of all POST requests in the chaincode	`bc-admin-user value`	all chaincodes
`bc-admin-password`	The password for the administrator user	`bc-admin-password value`	all chaincodes
`bc-timeout`	The timeout value in the body of every POST request to indicate the timeout interval	`6000`	all chaincodes
`bc-sync`	The sync value in the body of every POST request to indicate whether the request is synchronous or asynchronous	`true`	all chaincodes

Copyright © 2025, Oracle and/or its affiliates.

Variable Name	Description	Default Value	Context
`bc-chaincode-name`	The chaincode name, which is used in every POST request	`chaincode name`	all chaincodes
`bc-org-id`	The default `orgId` parameter for all POST requests	`bc-org-id value`	token chaincodes only
`bc-user-id`	The default `userId` parameter for all POST requests	`bc-user-id value`	token chaincodes only
`bc-token-id`	The default `tokenId` parameter for all POST requests	`bc-token-id value`	token chaincodes only

In every generated request, all of the parameters with default values are generated. Functions that have struct/class parameters will have a placeholder object in the request body, as shown in the following examples.

API with a struct/class parameter

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "CreateArtCollectionToken",
        "{\"TokenId\":\"{{bc-token-id}}\",\"TokenDesc\":\"TokenDesc
value\",\"TokenUri\":\"TokenUri value\",\"TokenMetadata\":
{\"Painting_name\":\"Painting_name value\",\"Description\":\"Description
value\",\"Image\":\"Image value\",\"Painter_name\":\"Painter_name
value\"},\"Price\":999,\"On_sale_flag\":true}",
        "quantity value"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}}
}
```

API without a struct/class parameter

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "CreateAccount",
        "{{bc-org-id}}",
        "example_minter",
        "true",
        "true"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}}
}
```

The default value for most API parameters is *`parameter_name`* `value`, with some exceptions. The following examples show some of the exceptions.

Copyright © 2025, Oracle and/or its affiliates.

- The filters parameter in `GetAccountTransactionHistoryWithFilters`:

  ```
  "{\"PageSize\":20,\"Bookmark\":\"\",\"StartTime\":\"2022-01-16T15:16:36+00:00\",\"EndTime\":\"2022-01-17T15:16:36+00:00\"}"
  ```

- The filters parameter in `GetSubTransactionsByIdWithFilters`:

  ```
  "{\"PageSize\":20,\"Bookmark\":\"\"}"
  ```

A struct or class has different default values, as shown in the following table:

Data Type	Default Value
`boolean/bool`	`true`
`int/number`	`999`
`date`	`2022-01-16T15:16:36+00:00`
`other`	*`parameter_name`* `value`

ERC-1155 Token Projects

The ERC-1155 standard includes common methods for both fungible and non-fungible tokens. The generated Postman collection for an ERC-1155 project that uses both fungible and non-fungible tokens includes two different POST requests, one for each type of token, for these common methods. If an ERC-1155 project uses only fungible or non-fungible tokens but not both types, then the generated Postman collection includes only one POST request for these common methods. The following table illustrates the generated API for the `AddRole` method.

Request Element	Fungible Tokens	Non-Fungible Tokens
Request Name	`AddRole -For Fungible`	`AddRole -For NonFungible`

Request Body	<pre>{
	"chaincode": "{{bc-chaincode-name}}",
	"args": [
	"AddRole",
	"{{bc-org-id}}",
	"{{bc-user-id}}",
	"role value (for example, minter / burner)",
	
	"{\"TokenId\":\"{{bc-token-id}}\"}"
	],
	"timeout": {{bc-timeout}},
	"sync": {{bc-sync}}
	}</pre>
		<pre>{
		"chaincode": "{{bc-chaincode-name}}",
		"args": [
		"AddRole",
		"{{bc-org-id}}",
		"{{bc-user-id}}",
		"role value (for example, minter / burner)",
		
		"{\"TokenName\":\"TokenName value\"}"
		],
		"timeout": {{bc-timeout}},
		"sync": {{bc-sync}}
		}</pre>

Copyright © 2025, Oracle and/or its affiliates.

Troubleshoot Blockchain App Builder CLI

The following information can be used to troubleshoot system problems with Blockchain App Builder CLI.

Prerequisites issues
Errors can occur if you modify or upgrade any of the prerequisite software that is required by Blockchain App Builder. You can use the `preReqCheck` command to check that your installation of Blockchain App Builder still meets the prerequisites.
Usage:

```
preReqCheck [options]

ochain preReqCheck -h
Usage: patch [options]

Validates the pre-requisites of Blockchain App Builder

Options :
    -h, --help              output command usage information
    -D, --debug             enable debug logging
```

If the prerequisites check fails with errors and warnings when you attempt to install Blockchain App Builder, you might see an error similar to the following example:

```
npm ERR! code 1
npm ERR! path C:\Users\opc\AppData\Roaming\npm\node_modules\@oracle\ochain-cli
npm ERR! command failed
npm ERR! command C:\Windows\system32\cmd.exe /d /s /c node build/pre-
install.js
npm ERR! NodeJs version is correct.
npm ERR! NPM version is correct.
npm ERR! Error:
npm ERR! Found 1 error(s) in pre-requisites check, failed with following
errors:
npm ERR! 1. Golang version mismatch. Expected 1.21.x, but found 1.18.5.
npm ERR!
npm ERR! Found 3 warning(s) in pre-requisites check.
npm ERR! 1. Docker is not installed. Please install Docker >= 18.09.0. To
deploy chaincodes in the local environment, please install the recommended
version of Docker.
npm ERR! 2. Docker Compose is not installed. Please install Docker Compose >=
1.23.0. To deploy chaincodes in the local environment, please install the
recommended version of Docker Compose.
npm ERR! 3. Git is not Installed. To sync chaincodes, please install the Git
according to the documentation.
npm ERR!     at preRequsiteCheck
(C:\Users\opc\AppData\Roaming\npm\node_modules\@oracle\ochain-
cli\build\lib\util\pre-install.js:435:15)
npm ERR!     at Object.<anonymous>
(C:\Users\opc\AppData\Roaming\npm\node_modules\@oracle\ochain-cli\build\pre-
install.js:10:44)
npm ERR!     at Module._compile (node:internal/modules/cjs/loader:1254:14)
```


Copyright © 2025, Oracle and/or its affiliates.

```
npm ERR!     at Module._extensions..js (node:internal/modules/cjs/
loader:1308:10)
npm ERR!     at Module.load (node:internal/modules/cjs/loader:1117:32)
npm ERR!     at Module._load (node:internal/modules/cjs/loader:958:12)
npm ERR!     at Function.executeUserEntryPoint [as runMain] (node:internal/
modules/run_main:81:12)
npm ERR!     at node:internal/main/run_main_module:23:47

npm ERR! A complete log of this run can be found in:
npm ERR!     C:\Users\opc\AppData\Local\npm-
cache\_logs\2023-08-25T09_58_34_514Z-debug-0.log
```

Deployment failure
Due to deployment failure, corrupt deployment, a Docker peer container being full, or a Docker peer being killed in the local network, you may see an error similar to:

```
============ Started instantiate Chaincode ============
[2028-19-01T19:25:lO.372] [ERROR] default - Error instantiating Chaincode
GollGl on channel mychannel, detailed
error: Error: error starting container: error starting container: Failed to
generate platform-specific docker
build: Failed to pull hyperledger/fabric-ccenv:latest : API error (404):
manifest for hyperledger/
fabric-ccenv:latest not found: manifest unknown: manifest unknown
[2020-19-01T19:25:10.372] (INFO) default -
============ Finished instantiate Chaincode ============
[2020-19-01119:25:10.372] [ERROR] default - Error: Error instantiating
Chaincode Goll01 on channel mychannel,
detailed error: Error: error starting container: error starting container:
Failed to generate platfom-specific
docker build: Failed to pull hyperledger/fabric-ccenv: latest : API error
(404): manifest for hyperledger/
fabric-ccenv:lalest not found: manifest unknown: manifest unknown exited:
signal: terminated
INFO: exited: signal: terminated

ERROR: Error in Chaincode deployment
```

This is caused by a peer container that did not start up properly. To work around this behavior, complete the following step.

- Open a new terminal window in the chaincode project and run the `ochain run -b` command.

Blockchain App Builder CLI rebuilds the local fabric environment and deploys your chaincode to the new environment.

macOS: Xcode
After a macOS upgrade, or if Xcode is not installed, you might see an error similar to the following in the error log:

```
gyp: No Xcode or CLT version detected!
gyp ERR! configure error
```


Copyright © 2025, Oracle and/or its affiliates.

```
gyp ERR! stack Error: `gyp` failed with exit code: 1
gyp ERR! stack     at
```

- To work around this behavior, open a terminal window and run the following commands:

  ```
  sudo rm -rf $(xcode-select --print-path)
  xcode-select --install
  ```


Copyright © 2025, Oracle and/or its affiliates.

3

Using the Blockchain App Builder Extension for Visual Studio Code

The Blockchain App Builder extension for Visual Studio Code helps you build and scaffold a fully-functional chaincode project from a specification file.

After the project is built, you can run and test it on a local Hyperledger Fabric network or your provisioned Oracle Blockchain Platform network. You can then run SQL rich queries, debug the chaincode, or write and run unit tests using the generated code.

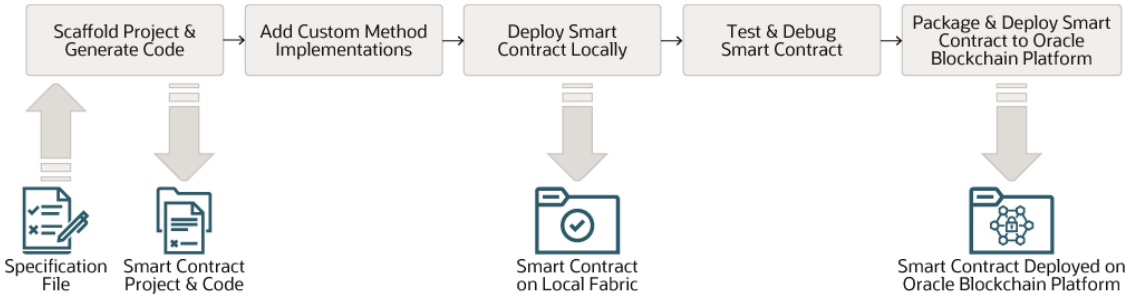


Table 3-1 Workflow When Using the Visual Studio Code Extension

Task	Description	More Information
Install and configure	Download the Blockchain App Builder Visual Studio Code extension from your Oracle Blockchain Platform console and install it and any prerequisite software.	• Install and Configure the Blockchain App Builder Extension for Visual Studio Code
Create the chaincode project	Create a specification file for the chaincode project.	• Create a Chaincode Project with the Blockchain App Builder Visual Studio Code Extension
Generate the chaincode	Edit the specification file to define the assets and chaincodes to generate, and then generate your chaincode from the specification file.	Detailed reference information about the structure and contents of the specification file and the generated chaincode project: • Input Specification File • Scaffolded TypeScript Chaincode Project • Scaffolded Go Chaincode Project Detailed information about tokenization support: • Tokenization Support Using Blockchain App Builder • Scaffolded TypeScript NFT Project for ERC-721 • Scaffolded Go NFT Project for ERC-721 • Scaffolded TypeScript Project for Token Taxonomy Framework • Scaffolded Go Project for Token Taxonomy Framework

Copyright © 2025, Oracle and/or its affiliates.

Table 3-1 (Cont.) Workflow When Using the Visual Studio Code Extension

Task	Description	More Information
Deploy the chaincode	Deploy the chaincode project locally to the included preconfigured Hyperledger Fabric network or remotely to Oracle Blockchain Platform Cloud or Enterprise Edition. You can also package the chaincode project for manual deployment to Oracle Blockchain Platform.	• Deploy the Chaincode to a Local Hyperledger Fabric Network • Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network • Package Your Chaincode Project for Manual Deployment to Oracle Blockchain Platform
Test the chaincode	After your chaincode is running on a network, you can test any of the generated methods. Additionally, If you chose to create the `executeQuery` method during your chaincode development, you can run SQL rich queries if your chaincode is deployed to an Oracle Blockchain Platform network.	• Test Your Chaincode on a Local Hyperledger Fabric Network • Testing Lifecycle Operations on a Remote Oracle Blockchain Platform Network • Execute Berkeley DB SQL Rich Queries
Debug the chaincode	You can do line-by-line debugging in Visual Studio Code.	• Debugging from Visual Studio Code
Synchronize your updates	When you update your specification file, you can synchronize the changes with the generated chaincode files.	• Synchronize Specification File Changes With Generated Source Code
Running unit tests	A basic unit test case setup is included in the project. Additional tests can be added and run.	• Writing Unit Test Cases and Coverage Reports for the Chaincode Project

Install and Configure the Blockchain App Builder Extension for Visual Studio Code

You can download the Blockchain App Builder extension for Visual Studio Code from the Oracle Blockchain Platform console.

The following platforms are supported:

- macOS
- Oracle Linux 8.0 or 9.0
- Microsoft Windows 10 or 11

Prerequisites

Before you install Blockchain App Builder on your local system, you must install the prerequisites.

Copyright © 2025, Oracle and/or its affiliates.

> **Note**
>
> Blockchain App Builder coordinates with Oracle Blockchain Platform and its compilers. If you use any versions of the prerequisites other than the ones mentioned in the following section, deploying your chaincode to a remote Oracle Blockchain Platform network might fail.

When you install Blockchain App Builder, a prerequisites check runs first. If the prerequisites check fails, the installation process stops.

- macOS
- Linux
- Windows

macOS

Prerequisites

- Rancher Desktop (tested with 1.19.3). Blockchain App Builder can also work with Docker, but it has been tested and verified with Rancher Desktop. If you plan to use Rancher Desktop, uninstall Docker completely before installing Rancher Desktop. After you install Rancher Desktop, ensure that the container runtime is set to `dockerd (moby)`. To verify the container runtime in Rancher Desktop 1.19.3, click **Kubernetes Settings** > **Container Runtime**.
- The latest release of Node.js version 18 (tested with 18.15.0 and 18.16.0), and npm v8.x or 9.x (tested with 9.5.0 and 9.5.1)
  Check the Node.js version by running the following command: `node --version`

  Check the npm version by running the following command: `npm --version`

  If you use a manager such as `nvm` or `nodenv` to install Node.js and npm, set the default/global version and then restart Visual Studio Code so that the version will be detected by the Prerequisites page.

  Do not use versions of Node.js earlier or later than version 18.
- Go v1.23.2 or later, but not Go v1.24 or later. After installing Blockchain App Builder, see Additional Setup for Go Chaincode Projects.
  Check the Go version by running the following command: `go version`
- If you plan to use the synchronization feature of Blockchain App Builder, install Git and configure your user name and email as shown in the following commands. Specify your user name and email address in the place of `<your_name>` and `<email>`.

  ```
  git config --global user.name "<your_name>"
  ```

  ```
  git config --global user.email "<email>"
  ```

- Visual Studio Code version 1.66.0 or later
  Check the Visual Studio Code version by running the following command: `code --version`

Install Node.js and npm by Using nvm

Copyright © 2025, Oracle and/or its affiliates.

Using nvm to install Node.js and npm gives you the ability to run more commands without `sudo`.

1. Enter the following command to install nvm:

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh |
   bash
   ```

2. Add the following code snippet to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

   ```
   export NVM_DIR="$HOME/.nvm"
   [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
   [ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This
   loads nvm bash_completion
   ```

3. Log out and then log back in to your operating system.
4. Enter the following command to verify the nvm installation:

   ```
   nvm version
   ```

5. Enter the following command to install Node.js and npm:

   ```
   nvm install 18.16.0
   ```

6. Enter the following command to set Node.js 18.16.0 as the default in nvm:

   ```
   nvm alias default 18.16.0
   ```

   The output of the command is the following text:

   ```
   default -> 18.16.0 (-> v18.16.0)
   ```

Linux

Prerequisites

- Docker v18.09.0 or later
- Docker Compose v1.23.0 or later
- The latest release of Node.js version 18 (tested with 18.15.0 and 18.16.0), and npm v8.x or v9.x (tested with 9.5.0 and 9.5.1)
  Check the Node.js version by running the following command: `node --version`

  Check the npm version by running the following command: `npm --version`

  If you use a manager such as `nvm` or `nodenv` to install Node.js and npm, set the default/global version and then restart Visual Studio Code so that the version will be detected by the Prerequisites page.

  Do not use versions of Node.js earlier or later than version 18.
- Go v1.23.2 or later, but not Go v1.24 or later. After installing Blockchain App Builder, see Additional Setup for Go Chaincode Projects.
  Check the Go version by running the following command: `go version`

Copyright © 2025, Oracle and/or its affiliates.

- If you plan to use the synchronization feature of Blockchain App Builder, install Git and configure your user name and email as shown in the following commands. Specify your user name and email address in the place of `<your_name>` and `<email>`.

  ```
  git config --global user.name "<your_name>"


  git config --global user.email "<email>"
  ```

- Visual Studio Code version 1.66.0 or later
  Check the Visual Studio Code version by running the following command: `code --version`

Install Node.js and npm by Using nvm

Using nvm to install Node.js and npm gives you the ability to run more commands without `sudo`.

1. Enter the following command to install nvm:

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh |
   bash
   ```

2. Add the following code snippet to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

   ```
   export NVM_DIR="$HOME/.nvm"
   [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
   [ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This
   loads nvm bash_completion
   ```

3. Log out and then log back in to your operating system.
4. Enter the following command to verify the nvm installation:

   ```
   nvm version
   ```

5. Enter the following command to install Node.js and npm:

   ```
   nvm install 18.16.0
   ```

6. Enter the following command to set Node.js 18.16.0 as the default in nvm:

   ```
   nvm alias default 18.16.0
   ```

   The output of the command is the following text:

   ```
   default -> 18.16.0 (-> v18.16.0)
   ```

Windows

Prerequisites

- Rancher Desktop (tested with 1.19.3). Blockchain App Builder can also work with Docker, but it has been tested and verified with Rancher Desktop.
- The latest release of Node.js version 18 (tested with 18.15.0 and 18.16.0). Do not use versions of Node.js earlier or later than version 18.

Copyright © 2025, Oracle and/or its affiliates.

- npm v8.x or v9.x (tested with 9.5.0 and 9.5.1)
- Go v1.23.2 or later, but not Go v1.24 or later. After installing Blockchain App Builder, see Additional Setup for Go Chaincode Projects.
- If you want to use the synchronization feature of Blockchain App Builder, install Git and configure your user name and email as shown in the following commands. Specify your user name and email address in the place of `<your_name>` and `<email>`.

```
git config --global user.name "<your_name>"


git config --global user.email "<email>"
```

Install Rancher Desktop

Complete the following steps to install Rancher Desktop on Microsoft Windows.

1. If Docker is installed on your local computer, uninstall it completely.
2. Download and install Rancher Desktop.
3. After the installation wizard completes, before you open Rancher Desktop, run the following commands:

```
wsl --install
wsl --set-default-version 2
wsl --setdefault rancher-desktop
```

4. Open Rancher Desktop to complete the setup process.
5. After you install Rancher Desktop, ensure that the container runtime is set to `dockerd (moby)`. To verify the container runtime in Rancher Desktop 1.19.3, click **Kubernetes Settings** > **Container Runtime**.

Install the Blockchain App Builder Extension

1. Download the extension from the **Developer Tools** tab on the **Blockchain App Builder** pane of the Oracle Blockchain Platform console. On the **Blockchain App Builder** pane, under the **Download** section, select **Visual Studio Code Extension**.
2. In Visual Studio Code, open the **Extensions** panel and then from the **More Actions** menu, select **Install from VSIX**.
3. Locate the downloaded `oracle-ochain-extension-x.x.x.vsix` file and then click **Install**. (Adjust the name of the `.vsix` file for the version that you are installing.)
4. Restart Visual Studio Code to complete installation of the extension.
5. To use the specification file validation functions, which automatically validate the specification file as you type, install the YAML extension from Red Hat. Open the **Extensions** panel, search for YAML, install the YAML Language Support extension, and then restart Visual Studio Code.

After installation, you can use the Oracle Blockchain App Builder icon on the left side of Visual Studio Code to open the Blockchain App Builder panel.

The Blockchain App Builder command line interface (CLI) is automatically installed as part of the extension for Visual Studio Code if you haven't already installed it separately. The CLI commands can be run from any terminal application, including the Visual Studio Code console

Copyright © 2025, Oracle and/or its affiliates.

window. Blockchain App Builder is installed globally, so you can run the CLI commands from any location in the file system.

Additional Setup for Go Chaincode Projects

To develop a Go project, you must set the `GOPATH` environment variable. This allows Go to locate your workspace and run your code.

- macOS
- Linux
- Windows

macOS

Before setting the `GOPATH` environment variable, make sure that a `go/` folder exists in your `$HOME` directory. If not, enter the following command to create a `go/` directory in your home directory:

```
mkdir $HOME/go
```

Set your `GOPATH` environment variable by adding the following variables to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

```
export PATH=$PATH:/usr/local/go/bin
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
```

After editing the file, run the following command to make your changes take effect immediately:

```
source ~/.bash_profile
```

Alternately, you can apply the change system-wide by adding the previous variables to the `/etc/bashrc` file.

Linux

Before setting the `GOPATH` environment variable, make sure that a `go/` folder exists in your `$HOME` directory. If not, enter the following command to create a `go/` directory in your home directory:

```
mkdir $HOME/go
```

Set your `GOPATH` environment variable by adding the following variables to the applicable file: `~/.bash_profile`, `~/.profile`, `~/.bashrc` or `~/.zshrc`.

```
export PATH=$PATH:/usr/local/go/bin
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
```


Copyright © 2025, Oracle and/or its affiliates.

After editing the file, run the following command to make your changes take effect immediately:

```
source ~/.bash_profile
```

Alternately, you can apply the change system-wide by adding the previous variables to the `/etc/bashrc` file.

Windows

Create a `go/` directory in your home directory: `C:\Users\<username>\go`.

Upgrade the Blockchain App Builder Extension for Visual Studio Code

To upgrade Blockchain App Builder, you must uninstall the previous version and then install the newer version.

1. In Visual Studio Code, open the **Extensions** panel and then select the Blockchain App Builder extension.
   The Blockchain App Builder extension page is displayed.
2. Click **Uninstall**.
3. After Blockchain App Builder is uninstalled, restart Visual Studio Code.
4. Download the latest version of the Blockchain App Builder extension from the **Developer Tools** tab on the Blockchain App Builder pane of the Oracle Blockchain Platform console, and then install Blockchain App Builder. For more information, see Install and Configure the Blockchain App Builder Extension for Visual Studio Code.

Create a Chaincode Project with the Blockchain App Builder Visual Studio Code Extension

To create a chaincode project with Blockchain App Builder, you scaffold a chaincode project from a detailed specification file. This generates a project with all the necessary files.

Background

Blockchain App Builder initializes and scaffolds a ready-to-use chaincode project. Based on simple input, the **Create New Chaincode** command can generate complex chaincode projects with features such as:

- Multiple assets (models) and their behaviors (controllers)
- Auto-generate CRUD (Create/Read/Update/Delete) and non-CRUD methods
- Automatic validation of arguments
- Marshalling/unmarshalling of arguments
- Transparent persistence capability (ORM)
- Rich query support
- Transient and private data support

Copyright © 2025, Oracle and/or its affiliates.

- Identity management

The generated project follows model/controller and decorator pattern, which enables an asset's properties that are maintained on the ledger to be specified as typed fields and extended with specific behaviors and validation rules. This reduces the number of lines of code which can help with readability and scalability.

Create a Specification File

You must first create an input specification file. You cannot alter the sample specification files that were installed as part of Blockchain App Builder, but you can duplicate them or use them as a reference file for your own specification files.

1. In the **Specifications** pane, select **Create New Spec File**.
2. In the **Specifications Details** pane, complete the following steps.
   - Enter the name for the specification file.
   - Select the file type. YAML and JSON formats are supported.
   - Optionally, enter a description for the file.
   - The **Reference File** drop-down list lets you generate your specification file from a file in your workspace, if you have a file you want to use as a template. If nothing is selected, the created file will be empty and you can enter your specification from scratch.
   - Enter the location where you want the specification file to be stored on your system.

   Click **Save**.

The specification file is created and shown in the **Specifications** pane. Click it to open it in the editor.

Import a Specification File

If you have a preexisting specification file, you can import it.

1. In the **Specifications** pane, click **More Actions** and then select **Import Specification**.
2. Browse to your file and click **Import Specification**.

The specification file is imported and shown in the **Specifications** pane. Click it to open it in the editor.

Duplicate a Specification File

You can also duplicate a specification file that's already in your **Specifications** pane by right-clicking it and selecting **Duplicate**. Right-click the file and select **Rename** to update the name.

For more details about the contents of specifications files, see Input Specification File. Use this information and the sample specification templates to create your specification content.

Validate a Specification File

If you installed the YAML Language Support extension, the specification file is automatically validated as you type. The following validation features are supported automatically:

- Command completion
- Completion of scalar nodes to schema defaults
- Context-sensitive help when you hover over a property or method
- Indents for array items as you type

Copyright © 2025, Oracle and/or its affiliates.

- Syntax error detection

Scaffolding the Chaincode Project

After you create a specification file you can generate a chaincode project.

1. In the **Chaincodes** pane, select **Create New Chaincode**.
2. In the **Chaincode Details** pane, complete the following steps.
   - Enter the name of your chaincode project.
   - Select the language: TypeScript or Go.
   - Select the specification file that you're using to create the chaincode.
   - Enter the location or Go domain where you want the project to be created in your local development environment.

   Click **Create**.

When your project is created, it will be shown in the **Chaincodes** pane. All the files required by the chaincode will be in the project. For a detailed overview of the files created, see:

- Scaffolded TypeScript Chaincode Project
- Scaffolded Go Chaincode Project

For a detailed overview of a token-based project, see also:

- Scaffolded TypeScript Token Project for ERC-1155
- Scaffolded Go Token Project for ERC-1155
- Scaffolded TypeScript NFT Project for ERC-721
- Scaffolded Go NFT Project for ERC-721
- Scaffolded TypeScript Project for Token Taxonomy Framework
- Scaffolded Go Project for Token Taxonomy Framework

> **Note**
>
> - The **Chaincodes** pane lets you open and work with content in the chaincode project, but won't let you add, delete, or rename files in the project. To do that, right-click your project and select **Open in Explorer**. This opens the project in the Visual Studio Code Explorer view.
> - Deleting or renaming files in the chaincode project can potentially break the link between the project files and the specification file used to create it. If you plan to synchronize your code between the two, don't change the file names.

Import an Existing Chaincode Project

If you've created a chaincode project through the CLI or you've cleaned your Visual Studio Code blockchain content and want to import a locally saved project, in the **Chaincodes** pane click the **More Actions...** icon and select **Import Chaincode**. Browse to the project and click **Import Chaincode**.

Copyright © 2025, Oracle and/or its affiliates.

Input Specification File

The Blockchain App Builder initialization command reads the input specification file and generates the scaffolded project, which includes tools to assist in the chaincode development process.

In the specification file you can specify multiple asset definitions and behavior, CRUD and non-CRUD method declaration, custom methods, validation of arguments, auto marshalling/unmarshalling, transparent persistence capability, and the ability to complete rich data queries using SQL SELECT statements or the CouchDB query language. These features are generated for you.

For information on specifying token assets see the following topics:

- Input Specification File for Token Taxonomy Framework
- Input Specification File for ERC-721
- Input Specification File for ERC-1155

The specification file can be written in either YAML or JSON format. You can see sample specification files in both formats in the Blockchain App Builder package download:

- `Fabcar-Typescript.yml`
- `Marbles-Go.yml`

> **Note**
>
> In accordance with Go conventions, exported names begin with a capital letter. Therefore all the asset properties and methods must have names starting with capital letters in the specification file.

Structure of the Specification File

Typically, you structure a specification file in the following way:

```
assets:
    name:
    type:
    properties:
        name:
        type:
        id:
        derived:
           strategy:
           algorithm:
           format:
        mandatory:
        default:
        validate:
    methods:
        crud:
        others:
customMethods:
```


Copyright © 2025, Oracle and/or its affiliates.

Blockchain App Builder supports two special asset types, embedded assets and token assets, in addition to generic assets with no specified type. The special assets are defined as `type: embedded` or `type: token` under the `assets:` section of the specification file.

Table 3-2 Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`assets:`	This property takes the definition and behavior of the asset. You can define multiple assets.	
`name:`	The name of the asset. The following names are reserved. Do not use these names for assets. • `account` • `role` • `hold` • `token` • `authorization` • `tokenAdmin` • `Account` • `Role` • `Hold` • `Token` • `Authorization` • `TokenAdmin`	`name: owner # Information about the owner`
`type:`	**Asset types** The following special asset types are supported: • `embedded` • `token` If you do not specify a `type` parameter in the `assets` section, the asset is of the generic type.	

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples

`type:`
type: embedded
	If this property is set to `embedded` the asset is defined as an embedded asset. Embedded assets do not have CRUD methods and must be part of another asset to be stored in the ledger.
	In the example, the property `address` is embedded, and is defined in another asset.
	Embedded assets do not support circular references. For instance, in the previous example the `address` asset cannot contain a reference to the `employee` asset.
		Asset: `employee`
		
		`name: employee`
		`properties:`
		`name: employeeId`
		`type: string`
		`mandatory: true`
		`id: true`
		
		`name: firstName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		
		`name: lastName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		
		`name: age`
		`type: number`
		`validate: positive(),min(18)`
		
		`name: address`
		`type: address`
		
		Asset: `address`
		
		`name: address`
		
		`type: embedded`
		
		`properties:`
		`name: street`
		`type: string`
		
		`name: city`
		`type: string`
		
		`name: state`
		`type: string`
		
		`name: country`
		`type: string`

`properties:`	Describes all of the properties of an asset.	

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`name:`	The name of the property.	`name: ownerId # Unique ID for each owner`

`id:`	• true
	This specifies the identifier of this asset. This property is mandatory.
		`name: owner # Information about the owner`
		`properties:`
		`  name: ownerId # Unique ID for each owner`
		`  type: string`
		`  mandatory: true`
		`  **id: true**`
		`  name: name # Name of the owner`
		`  type: string`
		`  mandatory: true`

`type:`	**Property types**
	The following basic property types are supported:
	• `number`
	• `float`
	• `string`
	• `boolean`
	• `date`
	• `array`
	For Go chaincodes, `number` is mapped to `int` and `float` is mapped to `float64`. Other types are not currently supported, including the following types:
	• `complex`
	• `unsigned/ signed int`
	• `8/16/32/64 bits`
		`name: year # Model year`
		`  **type: number**`
		`  mandatory: true`
		`  validate: min(1910),max(2020)`
		`  name: color # Color - no validation as color names are innumerable`
		`  **type: string**`
		`  mandatory: true`

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples

`derived:`	This property specifies that the id property is derived from other keys. Dependent properties must be of the `string` data type and not an embedded asset.
	This property has two mandatory parameters:
	• `strategy`: takes values of `concat` or `hash`.
	• `format`: takes an array of specification strings and values to be used by the strategy.
	Example 1:
	• The property `employeeID` is dependent on the `firstName` and `lastName` properties.
	• This property is a concatenation of the values listed in the `format` array.
	• `IND%1#%2%tIND` is the 0th index in the array and describes the final format.
	• %n is a position specifier that takes its values from the other indexes in the array.
	• `%t` indicates that the value is the `stub.timestamp` value from the channel header.
	• If you need to use the percent sign (%) in the `format` string, it must be escaped with another percent sign (%).
	• The final format in this example would
		Example 1
		`name: employee`
		`properties:`
		`name: employeeId`
		`type: string`
		`mandatory: true`
		`id: true`
		`derived:`
		`strategy: concat`
		`format:`
		`["IND%1#%2%tIND","firstName","lastName"]`
		`name: firstName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		`name: lastName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		`name: age`
		`type: number`
		`validate: positive(),min(18)`
		Example 2
		`name: account`
		`properties:`
		`name: accountId`
		`type: string`
		`mandatory: true`
		`id: true`
		`derived:`
		`strategy: hash`
		`algorithm: 'sha256'`
		`format: ["IND%1#%2%t","bankName","ifsccode"]`
		`name: bankName`
		`type: string`
		`validate: max(30)`
		`mandatory: true`
		`name: ifsccode`
		`type: string`
		`mandatory: true`

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
	be: `INDfirstName#lastName1606885454916IND` Example 2: • When using `hash`, you must also use the `algorithm` parameter. The default is `sha256`; `md5` is also supported. • `IND%1#%2%t` is the 0th index in the array and describes the final format. • `%n` is a position specifier that takes its values from the other indexes in the array. • `%t` indicates that the value is the `stub.timestamp` value from the channel header. • If you need to use the percent sign (%) in the `format` string, it must be escaped with another percent sign (%).	

`mandatory:`	• true
	• false
	The corresponding property is mandatory and cannot be skipped while creating an asset.
		`name: phone # Phone number - validate as (ddd)-ddd-dddd where dashes could also be periods or spaces`
		`type: string`
		`mandatory: true`
		`validate: /^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$/`
		`name: cars # The list of car VINs owned by this owner`
		`type: string[]`
		`mandatory: false`

`default:`	The default value of this property.	

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`validate:`	The given property is validated against some of the ready-to-use validations provided by Blockchain App Builder. You can chain validations if you ensure that the chain is valid. If the `validate` property is not provided, then the validation is done against only the property `type`.	

`validate:`
type: number
	• positive()
	• negative()
	• min()
	• max()
	These validations can be chained together, separated by commas.
		`name: offerApplied`
		`type: number`
		`validate: negative(),min(-4)`
		
		`name: year  # Model year`
		`type: number`
		`mandatory: true`
		`validate: min(1910),max(2020)`

`validate:`
type: string
	• `min()`
	• `max()`
	• `email()`
	• `url()`
	• `/regex/` supports PHP regex
	For Go chaincodes, regular expressions which contain certain reserved characters or whitespace characters must be properly escaped.
		`name: website`
		`type: string`
		`mandatory: false`
		`validate: url()`
		
		`name: phone  # Phone number - validate as (ddd)-ddd-dddd where dashes could also be periods or spaces`
		`type: string`
		`mandatory: true`
		`validate: /^\(?([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$/`
		
		`name: Color #Color can be red, blue, or green`
		`type: string`
		`mandatory: true`
		`validate: /^\\s*(red|blue|green)\\s*$/`

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
`validate:` *type: boolean*	• true • false In the example, the validation of property `active` is by the type itself (`boolean`)	`name: active` **`type: boolean`**
`validate:` *type: array*	By type itself, in the form of `type: number[]`, this conveys that the array is of type `number`. You can enter limits to the array in the format `number[1:5]`, which means that the minimum length is 1, and the maximum is 5. You can omit either one.	`name: items` **`type: number[:5]`**

`validate:`
type: date
	• `min()`
	• `max()`
	The date must be one of these formats:
	• `YYYY-MM-DD`
	• `YYYY-MM-DDTHH:MM:SSZ`, where T separates the date from the time, and the Z indicates UTC. Timezone offsets can replace the Z, as in -05:00 for Central Daylight Savings Time.
		`name: expiryDate`
		`type: date`
		`validate: max('2020-06-26')`
		
		`name: completionDate`
		`type: date`
		`validate: min('2020-06-26T02:30:55Z')`

`methods:`	Use this to state which of the CRUD (Create/Read/Update/Delete) or additional methods to generate.
	By default all CRUD and other methods are generated.
		`methods:`
		`  crud: [create, getById, update, delete]`
		`  others: [getHistoryById, getByRange]`

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples
crud:	• create • getByID (read) • update • delete If this array is left empty, no CRUD methods will be created. If the crud parameter is not specified, all four methods will be created by default. The crud parameter is not applicable to token and embedded assets.	methods: **crud: [create, getById, delete]** others: [] # no other methods will be created
others:	• getHistoryById • getByRange getHistoryById returns the history of the asset in a list. getByRange returns all of the assets in a given range. For more information, see getByRange (TypeScript) and GetByRange (Go). If this array is left empty, no other methods will be created. If the others parameter is not used at all, both methods will be created by default. The others parameter is not applicable to token and embedded assets.	methods: crud: [create, delete] **others: [] # no other methods will be created** methods: crud: [create, getById, update, delete] **others: [getHistoryById, getByRange]**

Copyright © 2025, Oracle and/or its affiliates.

Table 3-2 (Cont.) Specification File Parameter Descriptions and Examples

Entry	Description	Examples

`customMethods:`	This property creates invokable custom method templates in the main controller file. It takes the method signature and creates the function declaration in the controller file.
	You can provide language-specific function declarations here.
	A custom method named `executeQuery` is provided. If you add this method to the specification file, it details how Berkeley DB SQL and CouchDB rich queries can be executed. This method can be calld only when you are connected to Oracle Blockchain Platform Cloud or Enterprise Edition.
		TypeScript
		`customMethods:`
		`- executeQuery`
		`- "buyCar(vin: string, buyerId: string, sellerId: string, price: number, date: Date)"`
		`- "addCar(vin: string, dealerId: string, price: number, date: Date)"`
		Go
		`customMethods:`
		`- executeQuery`
		`- "BuyCar(vin string, buyerId string, sellerId string, price int)"`
		`- "AddCar(vin string, dealerId string, price int)"`

Scaffolded TypeScript Chaincode Project

Blockchain App Builder takes the input from your specification file and generates a fully-functional scaffolded chaincode project. The project contains automatically generated classes and functions, CRUD methods, SDK methods, automatic validation of arguments, marshalling/un-marshalling and transparent persistence capability (ORM).

If the chaincode project uses the TypeScript language, the scaffolded project contains three main files:

- `main.ts`
- `<chaincodeName>.model.ts`
- `<chaincodeName>.controller.ts`

All of the necessary libraries are installed and packaged. The `tsconfig.json` file contains the necessary configuration to compile and build the TypeScript project.

The `<chaincodeName>.model.ts` file in the `model` subdirectory contains multiple asset definitions and the `<chaincodeName>.controller.ts` file in the `controller` subdirectory contains the assets behavior and CRUD methods.

The decorators in `model.ts` and `controller.ts` provide support for features such as automatic validation of arguments, marshalling/unmarshalling of arguments, transparent persistence capability (ORM), and calling rich queries.

Reference:

Copyright © 2025, Oracle and/or its affiliates.

- Models
- Decorators
- ORM
- SDK Methods
- Controller
- Automatically Generated Methods
- Controller Method Details
- Custom Methods
- Init Method

Models

Every model class extends the `OchainModel` class, which has an additional read-only property called `assetType`. This property can be used to fetch only assets of this type. Any changes to this property are ignored during the creation and updating of the asset. The property value by default is `<modelName>`.

The `OchainModel` class enforces decorator behaviors on properties of the class.

```
@Id('supplierId')
export class Supplier extends OchainModel<Supplier> {
    public readonly assetType = 'supplier';
    @Mandatory()
    @Validate(yup.string())
    public supplierId: string;
```

Decorators

Class decorators

`@Id(identifier)`

This decorator identifies the property which uniquely defines the underlying asset. This property is used as a key of the record, which represents this asset in the chaincode's state. This decorator is automatically applied when a new TypeScript project is scaffolded. The `identifier` argument of the decorator takes the value from specification file.

```
@Id('supplierId')
export class Supplier extends OchainModel{
...
}
```

Property decorators

Multiple property decorators can be used. The decorators are resolved in top to bottom order.

```
@Mandatory()
```


Copyright © 2025, Oracle and/or its affiliates.

This decorator marks the following property as mandatory, so that it cannot be skipped while saving to the ledger. If skipped, an error is thrown.

```
@Mandatory()
public supplierID: string;
```

`@Default(param)`

This decorator indicates that the following property has a default value. The default value in the argument (`param`) is used when the property is skipped while saving to the ledger.

```
@Default('open for business')
@Validate(yup.string())
public remarks: string;
```

`@Validate(param)`

The property following this decorator is validated against the schema specified by the parameter. The argument `param` takes a yup schema, and multiple schema methods can be chained together to add complex validations. For more information, see https://www.npmjs.com/package/yup.

```
@Validate(yup.number().min(3))
public productsShipped: number;
```

`@ReadOnly(param)`

This property decorator marks the underlying property as having a read-only value. The value in the argument, for example *param*, is used when the property is saved in the ledger. After the value is set it cannot be edited or removed.

```
@ReadOnly('digicur')
public token_name: string;
```

`@Embedded(PropertyClass)`

This property decorator marks the underlying property as an embeddable asset. It takes the embeddable class as a parameter. This class must extend the `EmbeddedModel` class. This requirement is validated by the decorator.

In the following example, the `Employee` class has a property called `address` of type `Address`, which will be embedded in the `Employee` asset. This is denoted by the `@Embedded()` decorator.

```
export class Employee extends OchainModel<Employee> {

   public readonly assetType = 'employee';

   @Mandatory()
   @Validate(yup.string())
   public emplyeeID: string;
```


Copyright © 2025, Oracle and/or its affiliates.

```
   @Mandatory()
   @Validate(yup.string().max(30))
   public firstName: string;

   @Mandatory()
   @Validate(yup.string().max(30))
   public lastName: string;

   @Validate(yup.number().positive().min(18))
   public age: number;

   @Embedded(Address)
   public address: Address;
}


export class Address extends EmbeddedModel<Address> {

   @Validate(yup.string())
   public street: string;

   @Validate(yup.string())
   public city: string;

   @Validate(yup.string())
   public state: string;

   @Validate(yup.string())
   public country: string;
}
```

When a new instance of the `Address` class is created, all the properties of the `Address` class are automatically validated by the `@Validate()` decorator. Note that the `Address` class does not have the `assetType` property or `@Id()` class decorator. This asset and its properties are not saved in the ledger separately but are saved along with the `Employee` asset. Embedded assets are user-defined classes that function as value types. The instance of this class can only be stored in the ledger as a part of the containing object (`OchainModel` assets). All of the above decorators are applied automatically based on the input file while scaffolding the project.

```
@Derived(STRATEGY, ALGORITHM, FORMAT)
```

This decorator is used for defining the attribute derived from other properties. This decorator has two mandatory parameters:

- `STRATEGY`: Takes values of `CONCAT` or `HASH`. If `HASH` is specified, the additional parameter `ALGORITHM` is required. The default algorithm is `sha256`; `md5` is also supported.
- `FORMAT`: Takes an array of specification strings and values to be used by the strategy.

```
@Id('supplierID')
export class Supplier extends OchainModel<Supplier> {

   public readonly assetType = 'supplier';
```


Copyright © 2025, Oracle and/or its affiliates.

```
    @Mandatory()
    @Derived(STRATEGY.HASH.'sha256',['IND%1IND%2','license','name'])
    @Validate(yup.string())
    public supplierID: string;

    @Validate(yup.string().min(2).max(4))
    public license: string;

    @Validate(yup.string().min(2).max(4))
    public name: string;
```

Method decorators

`@Validator(…params)`

This decorator is applied on methods of the main controller class. This decorator is important for parsing the arguments, validating against all the property decorators, and returning a model/type object. Controller methods must have this decorator to be invokable. It takes multiple user-created models or yup schemas as parameters.

The order of the parameters must be exactly the same as the order of the arguments in the method.

In the following example, the `Supplier` model reference is passed in the parameter that corresponds to the `asset` type in the method argument. At run time, the decorator parses and converts the method argument to a JSON object, validates against the `Supplier` validators, and after successful validation converts the JSON object to a `Supplier` object and assigns it to the `asset` variable. Then the underlying method is finally called.

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
    return await this.Ctx.Model.save(asset);
}
```

In the following example, multiple asset references are passed; they correspond to the object types of the method arguments. Note the order of the parameters.

```
@Validator(Supplier, Manufacturer)
public async createProducts(supplier: Supplier, manufacturer: Manufacturer) {
}
```

Apart from asset references, yup schema objects can also be passed if the arguments are of basic types. In the following example, `supplierId` and `rawMaterialSupply` are of type `string` and `number` respectively, so the yup schema of similar type and correct order is passed to the decorator. Note the chaining of yup schema methods.

```
@Validator(yup.string(), yup.number().positive())
public async fetchRawMaterial(supplierID: string, rawMaterialSupply: number) {
    const supplier = await this.Ctx.Model.get(supplierID, Supplier);
    supplier.rawMaterialAvailable = supplier.rawMaterialAvailable +
rawMaterialSupply;
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Model.update(supplier);
}
```

ORM

Transparent Persistence Capability or simplified ORM is captured in the `Model` class of the Context (`Ctx`) object. If your model calls any of the following SDK methods, access them by using `this.Ctx.Model`.

The following SDK methods implement ORM:

- `save` calls the Hyperledger Fabric `putState` method.
- `get` calls the Hyperledger Fabric `getState` method.
- `update` calls the Hyperledger Fabric `putState` method.
- `delete` calls the Hyperledger Fabric `deleteState` method.
- `history` calls the Hyperledger Fabric `getHistoryForKey` method.
- `getByRange` calls the Hyperledger Fabric `getStateByRange` method.
- `getByRangeWithPagination` calls the Hyperledger Fabric `getStateByRangeWithPagination` method.

For more information, see: SDK Methods.

SDK Methods

> **Note**
>
> Beginning with version 21.3.2, the way to access the ORM methods has changed. Run the `ochain --version` command to determine the version of Blockchain App Builder.

In previous releases, the ORM methods were inherited from the `OchainModel` class. In version 21.3.2 and later, the methods are defined on the `Model` class of Context (`Ctx`) object. To call these methods, access them by using `this.Ctx.Model.<method_name>`.

The following example shows a method call in previous releases:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier){
    return await asset.save();
}
```

The following example shows a method call from the version 21.3.2 and later:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
      return await this.Ctx.Model.save(asset);
}
```

After you upgrade to version 21.3.2, make this change in all chaincode projects that you created with an earlier version of Blockchain App Builder. If you use the `sync` command to synchronize changes between the specification file and your source code, the changes are

Copyright © 2025, Oracle and/or its affiliates.

automatically brought to your controller for the ready-to-use methods. You still need to manually resolve any conflicts.

save
The save method adds the caller asset details to the ledger.

This method calls the Hyperledger Fabric putState internally. All marshalling/unmarshalling is handled internally. The save method is part of the Model class, which you access by using the Ctx object.

```
Ctx.Model.save(asset: <Instance of Asset Class> , extraMetadata?: any) :
Promise <any>
```

Parameters:

- extraMetadata : any (Optional) – Used to save metadata apart from the asset in the ledger.

Returns:

- Promise<any> - Returns a promise on completion.

Example:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
    return await this.Ctx.Model.save(asset);
}
```

get
The get method is a method of OchainModel class, which is inherited by the concrete model classes of {chaincodeName}.model.ts. The get method is part of the Model class, which you access by using the Ctx object.

To return any asset from a given id, use the generic controller method getAssetById.

```
Ctx.Model.get(id: string, modelName: <Model Asset Class Name>) :
Promise<asset>
```

Parameters:

- id : string – Key used to save data into the ledger.
- modelName: <Model Asset Class Name> – (Optional) Model asset class to return.

Returns:

- Promise: <Asset> - If the modelName parameter is not provided and data exists in ledger, then Promise<object> is returned. If the id parameter does not exist in ledger, an error message is returned. If the modelName parameter is provided, then an object of type <Asset> is returned. Even though any asset with given id is returned from the ledger, this method will handle casting into the caller Asset type. If the asset returned from the ledger is not of the Asset type, then the method throws an error. This check is done by the read-only assetType property in the Model class.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
@Validator(yup.string())
public async getSupplierById(id: string) {
    const asset = await this.Ctx.Model.get(id, Supplier);
    return asset;
}
```

In the example, `asset` is of the type `Supplier`.

`update`
The `update` method updates the caller `asset` details in the ledger. This method returns a promise.

This method calls the Hyperledger Fabric `putState` internally. All the marshalling/ unmarshalling is handled internally. The `update` method is part of the `Model` class, which you can access by using the `Ctx` object.

```
Ctx.Model.update(asset: <Instance of Asset Class> , extraMetadata?: any) :
Promise <any>
```

Parameters:

- `extraMetadata : any` (Optional) – Used to save metadata apart from the asset in the ledger.

Returns:

- `Promise<any>` - Returns a promise on completion.

Example:

```
@Validator(Supplier)
public async updateSupplier(asset: Supplier) {
    return await this.Ctx.Model.update(asset);
}
```

`delete`
This deletes the asset from the ledger given by `id` if it exists. This method calls the Hyperledger Fabric `deleteState` method internally. The `delete` method is part of the `Model` class, which you can access by using the `Ctx` object.

```
Ctx.Model.delete(id: string): Promise <any>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise <any>` - Returns a promise on completion.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
@Validator(yup.string())
public async deleteSupplier(id: string) {
    const result = await this.Ctx.Model.delete(id);
    return result;
}
```

`history`
The `history` method is part of the `Model` class, which you can access by using the `Ctx` object. This method returns the asset history given by `id` from the ledger, if it exists.

This method calls the Hyperledger Fabric `getHistoryForKey` method internally.

```
Ctx.Model.history(id: string): Promise <any>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise <any[]>` - Returns any [] on completion.

Example

```
@Validator(yup.string())
public async getSupplierHistoryById(id: string) {
    const result = await this.Ctx.Model.history(id);
    return result;
}
```

Example of the returned asset history for `getSupplierHistoryById`:

```
[
    {
        "trxId":
"8ef4eae6389e9d592a475c47d7d9fe6253618ca3ae0bcf77b5de57be6d6c3829",
        "timeStamp": 1602568005,
        "isDelete": false,
        "value": {
            "assetType": "supplier",
            "supplierId": "s01",
            "rawMaterialAvailable": 10,
            "license": "abcdabcdabcd",
            "expiryDate": "2020-05-28T18:30:00.000Z",
            "active": true
        }
    },
    {
        "trxId":
"92c772ce41ab75aec2c05d17d7ca9238ce85c33795308296eabfd41ad34e1499",
        "timeStamp": 1602568147,
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "isDelete": false,
        "value": {
            "assetType": "supplier",
            "supplierId": "s01",
            "rawMaterialAvailable": 15,
            "license": "valid license",
            "expiryDate": "2020-05-28T18:30:00.000Z",
            "active": true
        }
    }
]
```

getByRange
The `getByRange` method is a static method of `OchainModel` class which is inherited by the concrete `Model` classes of `{chaincodeName}.model.ts`.

This method returns a list of assets between the range `startId` and `endId`. This method calls the Hyperledger Fabric `getStateByRange` method internally.

If the `modelName` parameter is not provided, the method returns `Promise<Object [ ] >`. If the `modelName` parameter is provided, then the method handles casting into the caller `Model` type. In the following example, the result array is of the type `Supplier`. If the asset returned from the ledger is not of the `Model` type, then it will not be included in the list. This check is done by the read-only `assetType` property in the `Model` class.

To return all the assets between the range `startId` and `endId`, use the generic controller method `getAssetsByRange`.

```
Ctx.Model.getByRange(startId: string, endId: string, modelName: <Asset Model
Class Name> ): Promise <any>
```

Parameters:

- `startId : string` – Starting key of the range, which is included in the range.
- `endId : string` – Ending key of the range, which is excluded from the range.
- `modelName: <Model Asset Class Name>` – (Optional) Model asset class to return.

Returns:

- `Promise< Asset[ ] >` - Returns array of `<Asset>` on completion.

Example:

```
@Validator(yup.string(), yup.string())
public async getSupplierByRange(startId: string, endId: string) {
    const result = await this.Ctx.Model.getByRange(startId, endId, Supplier);
    return result;
}
```

getByRangeWithPagination
The `getByRangeWithPagination` method is a static method of `OchainModel` class which is inherited by the concrete `Model` classes of `{chaincodeName}.model.ts`.

Copyright © 2025, Oracle and/or its affiliates.

This method returns a list of asset between the range `startId` and `endId`. This method calls the Hyperledger Fabric `getStateByRangeWithPagination` method internally.

If the `modelName` parameter is not provided, the method returns `Promise<Object [ ] >`. If the `modelName` parameter is provided, then the method handles casting into the caller `Model` type. In the following example, the result array is of the type `Supplier`. If the asset returned from the ledger is not of the `Model` type, then it will not be included in the list. This check is done by the read-only `assetType` property in the `Model` class.

To return all the assets between the range `startId` and `endId`, filtered by page size and bookmarks, use the generic controller method `getAssetsByRange`.

```
public async getByRangeWithPagination<T extends OchainModel<T>>(startId:
string, endId: string, pageSize: number, bookmark?: string, instance?: new
(data: any, skipMandatoryCheck: boolean, skipReadOnlyCheck: boolean) => T):
Promise<T[]>
```

Parameters:

- `startId : string` – Starting key of the range, which is included in the range.
- `endId : string` – Ending key of the range, which is excluded from the range.
- `pageSize : number` - The page size of the query.
- `bookmark : string` - The bookmark of the query. Output starts from this bookmark.
- `modelName: <Model Asset Class Name>` – (Optional) Model asset class to return.

Returns:

- `Promise< Asset[ ] >` - Returns array of `<Asset>` on completion.

`getId`
When the asset has a derived key as `Id`, you can use this method to get a derived ID. This method returns an error if the derived key contains `%t` (time stamp).

Parameters:

- `object` – The object must contain all of the properties on which the derived key is dependent.

Returns:

- Returns the derived key as a string.

Example:

```
@Validator(yup.string(), yup.string())

public async customGetterForSupplier(license: string, name: string){
    let object = {
      license : license,
      name: name
    }
    const id = await this.Ctx.Model.getID(object);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return this.Ctx.Model.get(id);
}
```

For token SDK methods, see the topics under Tokenization Support Using Blockchain App Builder.

Controller

The main controller class extends the `OchainController` class. There is only one main controller.

```
export class TSProjectController extends OchainController{
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable from outside; the rest of them are hidden.

Automatically Generated Methods

As described in Input Specification File, you can specify which CRUD methods to generate in the specification file. For example, if you specified to generate all methods, the result would be similar to the following code:

```
@Validator(Supplier)
public async createSupplier(asset: Supplier) {
    return await this.Ctx.Model.save(asset);
}

@Validator(yup.string())
public async getSupplierById(id: string) {
    const asset = await this.Ctx.Model.get(id, Supplier);
    return asset;
}

@Validator(Supplier)
public async updateSupplier(asset: Supplier) {
    return await this.Ctx.Model.update(asset);
}

@Validator(yup.string())
public async deleteSupplier(id: string) {
    const result = await this.Ctx.Model.delete(id);
    return result;
}

@Validator(yup.string())
public async getSupplierHistoryById(id: string) {
    const result = await this.Ctx.Model.history(id);
    return result;
}

@Validator(yup.string(), yup.string())
public async getSupplierByRange(startId: string, endId: string) {
    const result = await this.Ctx.Model.getByRange(startId, endId, Supplier);
    return result;
}
```


Copyright © 2025, Oracle and/or its affiliates.

Controller Method Details

Apart from the previous model CRUD and non-CRUD methods, Blockchain App Builder provides ready-to-use support from our controller for the following Hyperledger Fabric methods:

- `getAssetById`
- `getAssetsByRange`
- `getAssetHistoryById`
- `query`
- `queryWithPagination`
- `generateCompositeKey`
- `getByCompositeKey`
- `getTransactionId`
- `getTransactionTimestamp`
- `getChannelID`
- `getCreator`
- `getSignedProposal`
- `getArgs`
- `getStringArgs`
- `getMspID`
- `getNetworkStub`

> **Note**
>
> These methods are available with the `this` context in any class that extends the `OChainController` class.

For example:

```
public async getModelById(id: string) {
    const asset = await this.getAssetById(id);
    return asset;
}
@Validator(yup.string(), yup.string())
public async getModelsByRange(startId: string, endId: string) {
    const asset = await this.getAssetsByRange(startId, endId);
    return asset;
}
public async getModelHistoryById(id: string) {
    const result = await this.getAssetHistoryById(id);
    return result;
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getAssetById`
The `getAssetById` method returns an asset based on a specified ID. This is a generic method and can be used to get asset of any type.

```
this.getAssetById(id: string): Promise<byte[]>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise <byte [ ]>` - Returns promise on completion. You have to convert `byte[]` into an object.

`getAssetsByRange`
The `getAssetsByRange` method returns all assets present from `startId` (inclusive) to `endId` (exclusive) irrespective of asset types. This is a generic method and can be used to get assets of any type.

```
this.getAssetsByRange(startId: string, endId: string):
Promise<shim.Iterators.StateQueryIterator>
```

Parameters:

- `startId : string` – Starting key of the range, which is included in the range.
- `endId : string` – Ending key of the range, which is excluded from the range.

Returns:

- `Promise< shim.Iterators.StateQueryIterator>` - Returns an iterator on completion. You have to iterate over it.

`getAssetHistoryById`
The `getAssetHistoryById` method returns history iterator of an asset for a specified ID.

```
this.getAssetHistoryById(id: string):
Promise<shim.Iterators.HistoryQueryIterator>
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Promise<shim.Iterators.HistoryQueryIterator>` - Returns a history query iterator. You have to iterate over it.

Copyright © 2025, Oracle and/or its affiliates.

`query`
The `query` method runs a rich SQL/Couch DB query over the ledger. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
this.query(queryStr: string):
Promise<shim.Iterators.StateQueryIterator>
```

Parameters:

- `queryStr : string` - Rich SQL/Couch DB query.

Returns:

- `Promise<shim.Iterators.StateQueryIterator>` - Returns a state query iterator. You have to iterate over it.

`queryWithPagination`
This method runs a rich SQL/Couch DB query over the ledger, filtered by page size and bookmarks. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
public async queryWithPagination(query: string, pageSize: number, bookmark?:
string)
```

Parameters:

- `query : string` - Rich SQL/Couch DB query.
- `pageSize : number` - The page size of the query.
- `bookmark : string` - The bookmark of the query. Output starts from this bookmark.

Returns:

- `Promise<shim.Iterators.StateQueryIterator>` - Returns a state query iterator. You have to iterate over it.

`generateCompositeKey`
This method generates and returns the composite key based on the `indexName` and the attributes given in the arguments.

```
this.generateCompositeKey(indexName: string, attributes:
string[]): string
```

Parameters:

- `indexName : string` - Object type of the key used to save data to the ledger.
- `attributes: string[ ]` - Attributes based on which a composite key will be formed.

Returns:

- `string` - Returns a composite key.

Copyright © 2025, Oracle and/or its affiliates.

`getByCompositeKey`
This method returns the asset that matches the key and the column given in the attribute parameter while creating a composite key. The `indexOfId` parameter indicates the index of the key returned in the array of the stub method `SplitCompositeKey`. Internally this method calls Hyperledger Fabric's `getStateByPartialCompositeKey`, `splitCompositeKey` and `getState` methods.

```
this.getByCompositeKey(key: string, columns: string[],
indexOfId: number): Promise<any []>
```

Parameters:

- `key: string` – Key used to save data to ledger.
- `columns: string[ ]` - Attributes based on which a key is generated.
- `indexOfId: number` - Index of attribute to be retrieved from key.

Returns:

- `Promise< any [ ]` - Returns any `[]` on completion.

`getTransactionId`
Returns the transaction ID for the current chaincode invocation request. The transaction ID uniquely identifies the transaction in the scope of the channel.

```
this.getTransactionId(): string
```

Parameters:

- none

Returns:

- `string` - Returns the transaction ID for the current chaincode invocation request.

`getTransactionTimestamp`
Returns the time stamp when the transaction was created. This is taken from the transaction `ChannelHeader`, therefore it will indicate the client's time stamp, and will have the same value across all endorsers.

```
this.getTransactionTimestamp(): Timestamp
```

Parameters:

- `id : string` – Key used to save data to the ledger.

Returns:

- `Timestamp` - Returns the time stamp when the transaction was created.

`getChannelID`
Returns the channel ID for the proposal for chaincode to process.

```
this.getChannelID(): string
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- none

Returns:

- `string` - Returns the channel ID.

`getCreator`
Returns the identity object of the chaincode invocation's submitter.

```
this.getCreator(): shim.SerializedIdentity
```

Parameters:

- none

Returns:

- `shim.SerializedIdentity` - Returns identity object.

`getSignedProposal`
Returns a fully decoded object of the signed transaction proposal.

```
this.getSignedProposal():
shim.ChaincodeProposal.SignedProposal
```

Parameters:

- none

Returns:

- `shim.ChaincodeProposal.SignedProposal` - Returns decoded object of the signed transaction proposal.

`getArgs`
Returns the arguments as array of strings from the chaincode invocation request.

```
this.getArgs(): string[]
```

Parameters:

- none

Returns:

- `string [ ]` - Returns arguments as array of strings from the chaincode invocation.

`getStringArgs`
Returns the arguments as array of strings from the chaincode invocation request.

```
this.getStringArgs(): string[]
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- none

Returns:

- `string [ ]` - Returns arguments as array of strings from the chaincode invocation.

`getMspID`
Returns the MSP ID of the invoking identity.

```
this.getMspID(): string
```

Parameters:

- none

Returns:

- `string` - Returns the MSP ID of the invoking identity.

`getNetworkStub`
You can access the shim stub by calling this method. This can help you write your own implementation of working directly with the assets.

```
this.getNetworkStub(): shim.ChaincodeStub
```

Parameters:

- none

Returns:

- `shim.ChaincodeStub` - Returns chaincode network stub.

`invokeCrossChaincode`
You can use this method in a chaincode to call a function in another chaincode. Both chaincodes must be installed on the same peer.

```
this.invokeCrossChaincode(chaincodeName: string, methodName: string, args:
string[], channelName: string): Promise<any>
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `Promise<any>` - Returns a JSON object that contains three fields:
  - `isValid` - `true` if the call is valid.

Copyright © 2025, Oracle and/or its affiliates.

  - `payload` - The output returned by the cross-chaincode call, as a JSON object.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

`invokeChaincode`
This method calls a function in another chaincode. Both chaincodes must be installed on the same peer.

```
this.invokeChaincode(chaincodeName: string, methodName: string, args:
string[], channelName: string): Promise<any>
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `Promise<any>` - Returns a JSON object that contains three fields:
  - `isValid` - `true` if the call is valid.
  - `payload` - The output returned by the cross-chaincode call, as a JSON object.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

Custom Methods

The following custom methods were generated from our example specification file.

The `executeQuery` method shows how SQL rich queries can be called. The validators against the arguments are added automatically by Blockchain App Builder based on the type of the argument specified in the specification file.

```
/**
*
*    BDB sql rich queries can be executed in OBP CS/EE.
*    This method can be invoked only when connected to remote OBP CS/EE
network.
*
*/
@Validator(yup.string()}
public async executeQuery(query: string) {
    const result = await OchainController.query(query);
    return result;
}
@Validator(yup.string(), yup.number()}
public async fetchRawMaterial(supplierId: string, rawMaterialSupply: number) {
}
@Validator(yup.string(), yup.string(), yup.number())
public async getRawMaterialFromSupplier(manufacturerId: string, supplierId:
string, rawMaterialSupply: number) {
}
@Validator(yup.string(), yup.number(), yup.number())
```


Copyright © 2025, Oracle and/or its affiliates.

```
public async createProducts(manufacturerId: string, rawMaterialConsumed:
number, productsCreated: number) {
}
public async sendProductsToDistribution() {
}
```

Init Method

A custom `init` method is provided in the controller with an empty definition. If you use Blockchain App Builder to deploy or upgrade, the `init` method is called automatically. If you deploy or upgrade from the Oracle Blockchain Platform console, you must call the `init` method manually. You can use a third-party tool such as Postman to call the `init` method manually.

```
export class TestTsProjectController extends OchainController {
    public async init(params: any) {
        return;
}
```

You can then use this method to initialize any application state.

Scaffolded Go Chaincode Project

Blockchain App Builder takes the input from your specification file and generates a fully-functional scaffolded chaincode project. The project contains automatically generated classes and functions, CRUD methods, SDK methods, automatic validation of arguments, marshalling/un-marshalling and transparent persistence capability (ORM).

If the chaincode project uses the Go language, the scaffolded project contains three main files:

- `main.go`
- `<chaincodeName>.model.go`
- `<chaincodeName>.controller.go`

All of the necessary libraries are installed and packaged.

The `<chaincodeName>.model.go` file in the `model` subdirectory contains multiple asset definitions and the `<chaincodeName>.controller.go` file in the `controller` subdirectory contains the asset's behavior and CRUD methods. The various Go struct tags and packages in `model.go` and `controller.go` provide support for features such as automatic validation of arguments, marshalling/unmarshalling of arguments, transparent persistence capability (ORM), and calling rich queries.

The scaffolded project is located in the `$GOPATH/src/example.com/<chaincodeName>` directory.

Reference:

- Model
- Validators
- ORM
- SDK Methods
- Composite Key Methods
- Stub Method

Copyright © 2025, Oracle and/or its affiliates.

- Other Methods
- Utility Package
- Controller
- Automatically Generated Methods
- Custom Methods
- Init Method

Model

Asset Type Property

By default every struct has an additional property called `AssetType`. You can use this property to fetch only assets of this type. Any changes to this property are ignored when assets are created or updated. The property value by default is `<modelName>`.

```
type Supplier struct {
AssetType string 'json:"AssetType" default:"TestGoProject.Supplier"'

SupplierId            string      'json:"SupplierId"
validate:"string,mandatory" id:"true'
RawMaterialAvailable  int         'json:"RawMaterialAvailable"
validate:"int,min=0"'
License               string      'json:"License" validate:"string,min=10"'
ExpiryDate            date.Date   'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
Active                bool      'json:"Active" validate:"bool" default:"true"'
Metadata              interface{} 'json:"Metadata,omitempty"'
}
```

Validators

Id

`id:"true"`

This validator identifies the property that uniquely defines the underlying asset. The asset is saved by the value in this key. This validator is automatically applied when a new Go project is scaffolded.

In the following example, `SupplierId` is the key for the supplier asset and has a tag property `id:"true"` for the `SupplierId` property.

```
type Supplier struct {
    Supplierld              string        'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable    int           'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License                 string        'json:"License"
validate:"string,min=10"'
    ExpiryDate              date.Date     'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                  bool          'json:"Active" validate:"bool"
default :"true"'
```


Copyright © 2025, Oracle and/or its affiliates.

```
    Metadata                interface{}   'json:"Metadata,omitempty"'
}
```

Derived

`derived:"strategy,algorithm,format"`

This decorator is used for defining the attribute derived from other properties. This decorator has two mandatory parameters:

- `strategy`: Takes values of `concat` or `hash`. If `hash` is specified, the additional parameter `algorithm` is required. The default algorithm is `sha256`; `md5` is also supported.
- `format`: Takes an array of specification strings and values to be used by the strategy.

```
type Supplier struct{
   AssetType string 'json:"AssetType" final:"chaincode1.Supplier"'
   SupplierId   string  'json:"SupplierId" validate:"string" id:"true"
mandatory:"true"
derived:"strategy=hash,algorith=sha256,format=IND%1%2,License,Name"'
   Name         string  'json:"Name" validate:"string,min=2,max=4"'
   License      string  'json:"License" validate:"string,min=2,max=4"'
}
```

Mandatory

`validate:"mandatory"`

This decorator marks the following property as mandatory, so that it cannot be skipped while saving to the ledger. If skipped, an error is thrown. In the following example, the `SupplierId` property has a `validate:"mandatory"` tag.

```
Type Supplier struct {
    Supplierld              string      'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable    int         'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License                 string      'json:"License"
validate:"string,min=10"'
    ExpiryDate              date.Date   'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                  bool        'json:"Active" validate:"bool"
default :"true"'
    Metadata                interface{} 'json:"Metadata,omitempty"'
}
```

Default

`default:"<param>"`

This decorator indicates that the following property has a default value. The value in the default tag is used when the property is skipped while saving to the ledger. In the following example, `Active` has a default value of `true`, specified as the tag `default:"true"`.

```
Type Supplier struct {
    Supplierld            string       'json:"Supplierld"
```


Copyright © 2025, Oracle and/or its affiliates.

```
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License               string       'json:"License"
validate:"string,min=10"'
    ExpiryDate            date.Date    'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                bool         'json:"Active" validate:"bool"
default :"true"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

Validate types
Basic Go types are validated for a property by defining a validate tag. The following validate tags are based on the basic types:

- string: `validate: "string"`
- date: `validate: "date"`
- number: `validate: "int"`
- boolean: `validate: "bool"`

Min validator
`validate:"min=<param>"`

You can use the `min` validator to set the minimum value for a property of type number or string.

For type int: In the following example, the `RawMaterialAvailable` property has a minimum value of 0. If a value less than 0 is applied to the `RawMaterialAvailable` property, an error will be returned.

For type string: The minimum validator checks the length of the string against the specified value. In the following example, the `License` property must be at least ten characters long.

```
Type Supplier struct {
    Supplierld            string       'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License               string       'json:"License"
validate:"string,min=10"'
    ExpiryDate            date.Date    'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                bool         'json:"Active" validate:"bool"
default :"true"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

Max validator
`validate:"max=<param>"`

You can use the max validator to set the maximum value for a property of type number and string.

Copyright © 2025, Oracle and/or its affiliates.

For type int: Similar to the min validator, if a value specified for a struct field is greater than the value provided in the validator, an error is returned.

For type string: Similar to the min validator, the max validator checks the length of the string against the given value. In the following example, the `Domain` property has a maximum value of 50, so if the `Domain` property has a string length of more than 50 characters, an error message will be returned.

```
type Retailer struct {
    Retailerld         string        'json:"Retailerld"
validate:"string,mandatory" id:"true"'
    ProductsOrdered    int           'json:"ProductsOrdered"
validate:"int,mandatory"'
    ProductsAvailable  int           'json:"ProductsAvailable" validate:"int"
default:"1"'
    ProductsSold       int           'json:"ProductsSold" validate:"int"'
    Remarks            string        'json:"Remarks" validate:"string"
default :"open for business"'
    Items              []int         'json:"Items"
validate:"array=int,range=l-5"'
    Domain             string        'json:"Domain"
validate:"url,min=30,max=50"'
    Metadata           interface{}   'json:"Metadata,omitempty"'
}
```

Date validators
Before validator:

```
validate:"before=<param>"
```

The before validator validates a property of type `date` to have a value less than the specified parameter.

In the following example, the `ExpiryDate` property must be before `"2020-06-26"` and if it is not, an error is returned.

```
Type Supplier struct {
    Supplierld            string       'json:"Supplierld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,min=0"'
    License               string       'json:"License"
validate:"string,min=10"'
    ExpiryDate            date.Date    'json:"ExpiryDate"
validate:"date,before=2020-06-26"'
    Active                bool         'json:"Active" validate:"bool"
default :"true"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

After validator:

```
validate:"after=<param>"
```


Copyright © 2025, Oracle and/or its affiliates.

The after validator validates a property of type `date` to have a value greater than the specified parameter.

In the following example, the `CompletionDate` property must be after `"2020-06-26"`, and if not it will return an error.

```
Type Supplier struct {
    Manufacturerld        string       'json:"Manufacturerld"
validate:"string,mandatory" id:"true"'
    RawMaterialAvailable  int          'json:"RawMaterialAvailable"
validate:"int,max=8"'
    ProductsAvailable     int          'json:"ProductsAvailable"
validate:"int"'
    CompletionDate        date.Date    'json:"CompletionDate"
validate:"date,after=2020-06-26"'
    Metadata              interface{}  'json:"Metadata,omitempty"'
}
```

URL validator
`validate:"url"`

The URL validator validates a property for URL strings.

In the following example, the `Domain` property must be a valid URL.

```
type Retailer struct {
    Retailerld         string        'json:"Retailerld"
validate:"string,mandatory" id:"true"'
    ProductsOrdered    int           'json:"ProductsOrdered"
validate:"int,mandatory"'
    ProductsAvailable  int           'json:"ProductsAvailable" validate:"int"
default:"1"'
    ProductsSold       int           'json:"ProductsSold" validate:"int"'
    Remarks            string        'json:"Remarks" validate:"string"
default :"open for business"'
    Items              []int         'json:"Items"
validate:"array=int,range=l-5"'
    Domain             string        'json:"Domain"
validate:"string,url,min=30,max=50"'
    Metadata           interface{}   'json:"Metadata,omitempty"'
}
```

Regexp validator
`validate:"regexp=<param>"`

The regexp validator validates a property with the specified regular expression.

In the following example, the `PhoneNumber` property will be validated for a mobile number in accordance with the regular expression.

```
type Customer struct {
Customerld      string       'json:"Customerld" validate:"string,mandatory"
id:"true"'
```


Copyright © 2025, Oracle and/or its affiliates.

```
Name            string       'json:"Name" validate:"string,mandatory"'
ProductsBought  int          'json:"ProductsBought" validate:"int"'
OfferApplied    int          'json:"OfferApplied" validate :"int,nax=0"'
PhoneNumber     string       'json:"PhoneNumber" validate:"string,regexp=A\(?
([0-9]{3})\)?[-. ]?([0-9]{3})[-. ]?([0-9]{4})$"'
Received        bool         'json:"Received" validate:"bool"'
Metadata        interface{}  'json:"Metadata,omitempty"'
}
```

Multiple validators
Multiple validators can be applied a property.

In the following example, the `Domain` property has validation for a string, a URL, and minimum and maximum string length.

```
type Retailer struct {
    Retailerld         string        'json:"Retailerld"
validate:"string,mandatory" id:"true"'
    ProductsOrdered    int           'json:"ProductsOrdered"
validate:"int,mandatory"'
    ProductsAvailable  int           'json:"ProductsAvailable" validate:"int"
default:"1"'
    ProductsSold       int           'json:"ProductsSold" validate:"int"'
    Remarks            string        'json:"Remarks" validate:"string"
default :"open for business"'
    Items              []int         'json:"Items"
validate:"array=int,range=l-5"'
    Domain             string        'json:"Domain"
validate:"string,url,min=30,max=50"'
    Metadata           interface{}   'json:"Metadata,omitempty"'
}
```

ORM

Transparent Persistence Capability or simplified ORM is captured in the `Model` class of the Context (`Ctx`) object. If your model calls any of the following SDK methods, access them by using `t.Ctx.Model`.

The following SDK methods implement ORM:

- `Save` calls the Hyperledger Fabric `PutState` method.
- `Get` calls the Hyperledger Fabric `GetState` method.
- `Update` calls the Hyperledger Fabric `PutState` method.
- `Delete` calls the Hyperledger Fabric `DeleteState` method.
- `History` calls the Hyperledger Fabric `GetHistoryForKey` method.
- `GetByRange` calls the Hyperledger Fabric `GetStateByRange` method.
- `GetByRangeWithPagination` calls the Hyperledger Fabric `GetStateByRangeWithPagination` method.

SDK Methods

Go chaincodes implement Transparent Persistence Capability (ORM) with the model package.

Copyright © 2025, Oracle and/or its affiliates.

> **Note**
>
> Beginning with version 21.2.3, the way to access the ORM methods has changed. Run the `ochain --version` command to determine the version of Blockchain App Builder.

In previous releases, the ORM methods were exposed as static methods in the model package. The methods are now defined on the model receiver, which holds the transaction stub. To call these methods, you use the model receiver held by the transaction context in the controller. You call these methods as `t.Ctx.Model.<method_name>` instead of `model.<method_name>`.

The following example shows `Save` and `Get` method calls in previous releases:

```
func (t *Controller) CreateSupplier(asset Supplier) (interface{}, error) {
  return model.Save(&asset)
}

func (t *Controller) GetSupplierById(id string) (Supplier, error) {
  var asset Supplier
  _, err := model.Get(id, &asset)
  return asset, err
}
```

The following example shows `Save` and `Get` method calls from the version 21.2.3 and later:

```
func (t *Controller) CreateSupplier(asset Supplier) (interface{}, error) {
  return t.Ctx.Model.Save(&asset)
}

func (t *Controller) GetSupplierById(id string) (Supplier, error) {
  var asset Supplier
  _, err := t.Ctx.Model.Get(id, &asset)
  return asset, err
}
```

After you upgrade to version 21.2.3, make this change in all chaincode projects that you created with an earlier version of Blockchain App Builder. If you use the `sync` command to synchronize changes between the specification file and your source code, the changes are automatically brought to your controller for the ready-to-use methods. You must still manually resolve any conflicts.

The following ORM methods are exposed by the model package:

`Get`
Queries the ledger for the stored asset based on the given ID.

```
func Get(Id string, result ...interface{}) (interface{}, error)
```

Parameters:

- `Id` - The ID of the asset to retrieve from the ledger.

Copyright © 2025, Oracle and/or its affiliates.

- `result (interface{})` - An empty asset object of a particular type, which is passed by reference. This object will contain the result from this method. Use this parameter only if a type-specific result is required.
- `asset (interface)` - An empty asset object, which is passed by reference. This object will contain the result from this method. Use this parameter only if a type-specific result is required.

Returns:

- `interface {}` - The interface contains the asset in `map[string]interface{}` form. Before operating on this map, you must assert the returned interface with type `map[string]interface{}`. To convert this map into an asset object, you can use the utility API `util.ConvertMaptoStruct` (see: Utility Package).
- `error` - Contains an error if returned, or is nil.

`Update`
Updates the specified asset in the ledger with new values.

```
func Update(args ...interface{}) (interface{}, error)
```

Parameters:

- `obj (interface)` - The object to update in the ledger is passed by reference into this API along with the new values. The input asset is validated and verified according to the struct tags in the model specification and then stored into the ledger.

Returns:

- `interface{}` - The saved asset is returned as an interface.
- `error` - Contains an error if returned, or is nil.

`Save`
Saves the asset to the ledger after validating on all the struct tags.

```
func Save(args ...interface{}) (interface{}, error)
```

Parameters:

- `obj/args[0] (interface{})` - The object to store in the ledger is passed by reference in this utility method.
- `metadata/args[1] (interface{})` - (Optional) You can use this parameter to store metadata to the ledger along with the asset at runtime.

Returns:

- `interface {}` - The asset is returned as an interface.
- `error` - Contains an error if returned, or is nil.

`Delete`
Deletes the asset from the ledger.

```
func Delete(Id string) (interface{}, error)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `id (string)` - The ID of the asset to delete from the ledger.

Returns:

- `interface {}` - Contains the asset being deleted in `map[string]interface{}` form.

`GetByRange`
Returns a list of assets specified by a range of IDs.

```
func GetByRange(startKey string, endKey string, asset ...interface{})
([]map[string]interface{}, error)
```

Parameters:

- `startkey (string)` - Starting ID for the range of objects to retrieve.
- `endkey (string)` - End ID for the range of objects to retrieve.
- `asset interface` - (Optional) An empty array of assets, passed by reference. This array will contain the method result. Use this parameter if a type-specific result is required.

Returns:

- `[]map[string]interface{}` - This array contains the list of assets obtained from the ledger. You can access the objects by iterating over this array and asserting the objects as `map[string]interface{}` and using a utility to convert to an asset object.
- `error` - Contains an error if returned, or is nil.

`GetByRangeWithPagination`
The `GetByRangeWithPagination` method is a static method of `OchainModel` class which is inherited by the concrete `Model` classes of `{chaincodeName}.model.ts`.

This method returns a list of assets between the range `startId` and `endId`, filtered by page size and bookmark. This method calls the Hyperledger Fabric `GetStateByRangeWithPagination` method internally.

If the `modelName` parameter is not provided, the method returns `Promise<Object [ ] >`. If the `modelName` parameter is provided, then the method handles casting into the caller `Model` type. In the following example, the result array is of the type `Supplier`. If the asset returned from the ledger is not of the `Model` type, then it will not be included in the list. This check is done by the read-only `assetType` property in the `Model` class.

To return all the assets between the range `startId` and `endId`, filtered by page size and bookmarks, use the generic controller method `getAssetsByRange`.

```
func (m *Model) GetByRangeWithPagination(startKey string, endKey string,
pageSize int32, bookmark string, asset ...interface{})
([]map[string]interface{}, error)
```

Parameters:

- `startkey : string` – Starting key of the range, which is included in the range.
- `endkey : string` – Ending key of the range, which is excluded from the range.

Copyright © 2025, Oracle and/or its affiliates.

- `pageSize : number` – The page size of the query.
- `Bookmark : string` – The bookmark of the query. Output starts from this bookmark.
- `asset interface` – (Optional) An empty array of assets, passed by reference. This array will contain the result from this method. Use this parameter to get type-specific results.

Returns:

- `[]map[string]interface{}` – An array that contains the list of assets retrieved from the ledger. You can access the objects by iterating over this array and asserting the objects as `map[string]interface{}` and using a utility for conversion to an asset object.
- `error` – Contains an error if an error is returned, otherwise nil.

`GetHistoryById`
Returns the history of the asset with the given ID.

```
func GetHistoryByID(Id string) ([]interface{}, error)
```

Parameters:

- `Id (string)` - ID of the asset.

Returns:

- `[]interface{}` - This slice contains the history of the asset obtained from the ledger in form of slice of `map[string]interface{}`. You can access each history element by iterating over this slice and asserting the objects as `map[string]interface{}` and using a utility to convert to asset object.
- `error` - Contains the error if an error is returned.

`Query`
This method runs a SQL/Couch DB query over the ledger. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
func Query(queryString string) ([]interface{}, error)
```

Parameters:

- `queryString (string)` - The query string.

Returns:

- `[]interface{}` - The query output in the form of a slice of interfaces. Iterate over the slice and use the elements by converting them to their proper types.
- `error` - Contains the error if an error is returned.

Copyright © 2025, Oracle and/or its affiliates.

`QueryWithPagination`
This method runs a SQL/Couch DB query over the ledger, filtered by page size and bookmark. This method is supported only for remote deployments on Oracle Blockchain Platform. This is a generic method for running SQL queries on the ledger.

```
func (m *Model) QueryWithPagination(queryString string, pageSize int32,
bookmark string) ([]interface{}, error)
```

Parameters:

- `queryString (string)` - Rich SQL/Couch DB query.
- `pageSize : number` - The page size of the query.
- `bookmark : string` - The bookmark of the query. Output starts from this bookmark.

Returns:

- `[]interface{}` - The query output in the form of a slice of interfaces. Iterate over the slice and use the elements by converting them to their proper types.
- `error` - Contains the error if an error is returned.

`InvokeCrossChaincode`
You can use this method in a chaincode to call a function in another chaincode. Both chaincodes must be installed on the same peer.

```
func InvokeCrossChaincode(chaincodeName string, method string, args []string,
channelName string) (interface{}, error)
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `interface{}` - Returns a `map[string]interface{}` object that contains three keys:
  - `isValid` - `true` if the call is valid.
  - `payload` - The output returned by the cross-chaincode call, as a JSON object.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

Return Value Example:

```
{
      "isValid": true,
      "message": "Successfully invoked method [CreateAccount] on sub-
chaincode [erc721_go_453]",
      "payload": {
                  "AccountId":
"oaccount~6b83b8ab931f99442897dd04cd7a2a55f808686f49052a40334afe3753fda4c4",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "AssetType": "oaccount",
                    "BapAccountVersion": 0,
                    "NoOfNfts": 0,
                    "OrgId": "appdev",
                    "TokenType": "nonfungible",
                    "UserId": "user2"
        }
}
```

InvokeChaincode
You can use this method in a chaincode to call a function in another chaincode. Both chaincodes must be installed on the same peer.

```
func InvokeChaincode(chaincodeName string, method string, args []string,
channelName string) (interface{}, error)
```

Parameters:

- `chaincodeName` – The name of the chaincode to call.
- `methodName` - The name of the method to call in the chaincode.
- `arg` - The argument of the calling method.
- `channelName` - The channel where the chaincode to call is located.

Returns:

- `interface{}` - Returns a `map[string]interface{}` object that contains three keys:
  - `isValid` - `true` if the call is valid.
  - `payload` - The output returned by the cross-chaincode call, in UTF-8 format.
  - `message` - The message returned by the cross-chaincode call, in UTF-8 format.

Return Value Example:

```
{
    "isValid": true,
    "message": "Successfully invoked method [CreateAccount] on sub-chaincode
[erc721_go_453]",
    "payload":
"{\"AssetType\":\"oaccount\",\"AccountId\":\"oaccount~c6bd7f8dcc339bf7144ea2e1
cf953f8c1df2f28482b87ad7895ac29e7613a58f\",\"UserId\":\"user1\",\"OrgId\":\"ap
pdev\",\"TokenType\":\"nonfungible\",\"NoOfNfts\":0,\"BapAccountVersion\":0}"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Composite Key Methods

`GenerateCompositeKey`
This method generates and returns the composite key based on the `indexName` and the attributes given in the arguments.

```
func GenerateCompositeKey(indexName string, attributes []string)
            (string, error)
```

Parameters:

- `indexName (string)` - Object type of the composite key.
- `attrbutes ([]string)` - Attributes of the asset based on which the composite key will be formed.

Returns:

- `string` - The composite key result.
- `error` - Contains the error if an error is returned.

`GetByCompositeKey`
This method returns the asset that matches the specified key and column. The `index` parameter indicates the index of the key returned in the array of stub method `SplitCompositeKey`.

Internally this method calls Hyperledger Fabric's `getStateByPartialCompositeKey`, `splitCompositeKey` and `getState` methods.

```
func GetByCompositeKey(key string, columns []string, index int)
                (interface{}, error)
```

Parameters:

- `key (string)` - Object type provided while creating composite key.
- `column ([]string)` - The slice of attributes on which the ledger will be queried using the composite key.
- `index(int)` - Index of the attribute.

Returns:

- `Interface{}` - The list of matching assets.
- `error` - Contains the error if an error is returned.

Stub Method

`GetNetworkStub`
This method returns the Hyperledger Fabric `chaincodeStub` value.

Copyright © 2025, Oracle and/or its affiliates.

You can access the shim stub by calling the `GetNetworkStub` method. This can assist you in writing your own implementation that works directly with assets.

```
func GetNetworkStub() shim.ChaincodeStubInterface
```

Parameters:

- none

Returns:

- `shim.ChaincodeStubInterface` - The Hyperledger Fabric chaincode stub.

Other Methods

- `GetTransactionId()`
- `GetTransactionTimestamp()`
- `GetChannelID()`
- `GetCreator()`
- `GetSignedProposal()`
- `GetArgs()`
- `GetStringArgs()`
- `GetCreatorMspId()`
- `GetId`

`GetTransactionId`
This method returns the transaction ID for the current chaincode invocation request. The transaction ID uniquely identifies the transaction in the scope of the channel.

```
func GetTransactionId() string
```

Parameters:

- none

Returns:

- `string` - The transaction ID.

`GetTransactionTimestamp`
Returns the time stamp when the transaction was created. Because the value is taken from the transaction `ChannelHeader`, it indicates the client's time stamp and has the same value across all endorsers.

```
func GetTransactionTimestamp() (*timestamp.Timestamp, error)
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- `timestamp.Timestamp` - The time stamp.
- `error` - Contains the error if an error is returned.

`GetChannelID`
Returns the channel ID for the proposal for the chaincode to process.

```
func GetChannelID() string
```

Parameters:

- none

Returns:

- `string` - The requested channel ID in string format.

`GetCreator`
Returns the identity object of the chaincode invocation's submitter.

```
func GetCreator() ([]byte, error)
```

Parameters:

- none

Returns:

- `[]byte` - The required identity object in serialized form.
- `error` - Contains the error if an error is returned.

`GetSignedProposal`
Returns a fully decoded object of the signed transaction proposal.

```
func GetSignedProposal() (*peer.SignedProposal, error)
```

Parameters:

- none

Returns:

- `*peer.SignedProposal` - The signed proposal object.
- `error` - Contains the error if an error is returned.

`GetArgs`
Returns the arguments as an array of strings from the chaincode invocation request.

```
func GetArgs() [][]byte
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- `[][]byte` - The passed arguments.

`GetStringArgs`
Returns the arguments intended for the chaincode `Init` and `Invoke` methods as a string array.

```
func GetStringArgs() []string
```

Parameters:

- none

Returns:

- `[]string` - The arguments as a string array.

`GetCreatorMspId`
Returns the MSP ID of the invoking identity.

```
func GetCreatorMspId() string
```

Parameters:

- none

Returns:

- `string` - The MSP ID of the invoking identity.

`GetId`
When the asset has a derived key as `Id`, you can use this method to get a derived ID. This method returns an error if the derived key contains `%t` (time stamp).

Parameters:

- `object` - The object must contain all of the properties on which the derived key is dependent.

Returns:

- Returns the derived key as a string.

Example:

```
func (t *Controller) CustomGetterForSupplier(License string, Name string)
(interface{}, error){
   var asset Supplier
   asset.License = License
   asset.Name = Name
   id,err := t.Ctx.Model.GetId(&asset)

   if err !=nil {
      return nil, fmt.Errorf("error in getting ID %v", err.Error())
   }
   return t.GetSupplierById(id)
}
```


Copyright © 2025, Oracle and/or its affiliates.

Utility Package

The following methods are provided in the utility package.

`Util.CreateModel`
Parses the specified JSON string and creates an asset object of the specified type.

```
func CreateModel(obj interface{}, inputString string) error
```

Parameters:

- `inputString (string)` - The input JSON string to use to create the object.
- `obj (interface{})` - The reference of the object to create from the JSON string. This object will store the created model, which is also validated in accordance with validator tags.

Returns:

- `error` - Contains the error if an error is returned while creating or validating the asset.

`util.ConvertMapToStruct`
Convert the specified map into an object of the specified type.

```
func ConvertMapToStruct(inputMap map[string](interface{}), resultStruct
interface{}) error
```

Parameters:

- `inputMap (map[string](interface{}))` - The map to convert into an asset object.
- `resultStruct (interface{})` - The reference of the asset object to create from the map.

Returns:

- `error` - Contains the error if an error is returned while creating or validating the asset.

For token SDK methods, see the topics under Tokenization Support Using Blockchain App Builder.

Controller

The `Controller.go` file implements the CRUD and custom methods for the assets.

You can create any number of classes, functions, or files, but only those methods that are defined on chaincode struct are invokable from outside; the rest of them are hidden.

Automatically Generated Methods

As described in Input Specification File, you can specify which CRUD methods you want generated in the specification file. For example, if you chose to generate all methods, the result would look similar to the following code:

```
//
//Supplier
//
func (t *ChainCode) CreateSupplier(inputString string) (interface{}, error) {
    var obj Supplier
```


Copyright © 2025, Oracle and/or its affiliates.

```
    err := util.CreateModel(&obj, inputString)
    if err != nil {
        return nil, err
    }
    return model.Save(&obj)
}

func (t *ChainCode) GetSupplierById(id string) (interface{}, error) {
    asset, err := model.Get(id)
    return asset, err
}

func (t *ChainCode) UpdateSupplier(inputString string) (interface{}, error) {
    var obj Supplier
    err := util.CreateModel(&obj, inputstring)
    if err != nil {
        return nil, err
    }
return model.Update(&obj)
}

func (t *ChainCode) DeleteSupplier(id string) (interface{}, error) {
    return model.Delete(id)
}

func (t *ChainCode) GetSupplierHistoryById(id string) (interface{}, error) {
    historyArray, err := model.GetHistoryByld(id)
    return historyArray, err
}

func (t *ChainCode) GetSupplierByRange(startkey string, endKey string)
(interface{}, error) {
    assetArray, err := model.GetByRange(startkey, endKey)
    return assetArray, err
}
```

Custom Methods

The following custom methods were generated from the example specification file.

The `executeQuery` function shows how SQL rich queries can be called. The validators against the arguments are added automatically by Blockchain App Builder based on the type of the argument specified in the specification file.

You can implement functionality according to the necessary business logic. If you add custom methods, add them to the controller file. If you add custom methods to the library instead of the controller file, your changes will be lost when the library folder contents are updated during the synchronization or chaincode upgrade processes.

```
//
//Custom Methods
//
/*
*    BDB sql rich queries can be executed in OBP CS/EE.
*    This method can be invoked only when connected to remote OBP CS/EE
network.
```


Copyright © 2025, Oracle and/or its affiliates.

```
*/
func (t *ChainCode) ExecuteQuery(inputQuery string) (interface{}, error) {
    resultArray, err := model.Query(inputQuery)
    return resultArray, err
}

func (t *ChainCode) FetchRawMaterial(supplierId string, rawMaterialSupply
int) (interface{}, error) {
    return nil, nil
}

func (t *ChainCode) GetRawMaterialFromSupplier(manufacturerId string,
supplierId string, rawMaterialSupply int) (interface{} error) {
    return nil, nil
}

Func (t *ChainCode) CreateProducts(manufacturerId string, rawMaterialConsumed
int, productsCreated int) (interface{}, error) {
    return nil, nil
}

func (t *ChainCode) SendProductsToDistribution() (interface{}, error) {
    return nil, nil
    }
```

For Go chaincodes, every custom method must return two values: an *empty interface* and an *error*, as shown in the following example:

```
func (t *Controller) FetchRawMaterial(supplierId string, rawMaterialSupply
int) (interface{}, error) {
    return nil, nil
}
```

Init Method

A custom `Init` method is provided in the controller with an empty definition. If you use Blockchain App Builder to deploy or upgrade, the `Init` method is called automatically. If you deploy or upgrade from the Oracle Blockchain Platform console, you must call the `Init` method manually. You can use a third-party tool such as Postman to call the `Init` method manually.

```
type Controller struct {
}
func (t *Controller) Init(args string) (interface{}, error)
    { return nil, nil
}
```

You can then use this method to initialize any application state.

Deploy Your Chaincode Using Visual Studio Code

After you create a chaincode project, you can deploy it locally to the automatically generated Hyperledger Fabric network or remotely to your Oracle Blockchain Platform Cloud or

Copyright © 2025, Oracle and/or its affiliates.

Enterprise Edition. You can also package the chaincode project for manual deployment to Oracle Blockchain Platform.

Deploy the Chaincode to a Local Hyperledger Fabric Network

After you create your chaincode project, you can test it in a local Hyperledger Fabric basic network.

When you install the Blockchain App Builder extension for Visual Studio Code, it automatically creates a Hyperledger Fabric network with a single channel. This is listed as `Local Environment` in the **Environments** pane. You can't delete or modify this environment; you can just deploy chaincodes to it and rebuild it if it stops working correctly.

The test network that is created when you install Blockchain App Builder runs Hyperledger Fabric v2.5.7 and Hyperledger Fabric CA v1.5.9.

Blockchain App Builder chaincode deployment starts the Hyperledger Fabric basic network, other services, and installs and deploys the chaincode for you.

1. In the **Chaincode Details** pane, select **Deploy**.
2. In the deployment wizard:
   - Ensure that the correct chaincode name is selected.
   - Select your target environment. For local deployment choose **Local Environment**.
   - Select the channel that you want to deploy to. A channel named "`mychannel`" is created by default when the extension is installed, and can be used for testing.
   - Optionally, enter any initial parameters that are required.
   - For token projects, the first time that you deploy you must enter a list of token administrators as a parameter. The list is an array of `{user_id, org_id}` information that specifies the token administrators. For the local Hyperledger Fabric network, use the value `Org1MSP` for the `org_id` field. For NFT chaincodes, the keys for the `adminList` parameter are `userId` and `orgId` for TypeScript and `UserId` and `OrgId` for Go. After the first time you deploy, you can supply an empty array for the `adminList` parameter or you can use the `adminList` parameter to add token administrators. Other deployers who are not the first-time deployer must supply an empty array for the `adminList` parameter. To do so, open the `Init` parameter list in the deployment pane and then click the minus sign (–) button next to the `adminList` parameter, which will select an empty array.
3. Click **Deploy**.

When the chaincode has finished deploying, the **Output** console will show that the chaincode is successfully installed and deployed on the given channel.

Troubleshooting

You might encounter the following issues when running your chaincode project on a local network.

Copyright © 2025, Oracle and/or its affiliates.

Missing Go permissions
While installing Go chaincode project in local network, you might see an error similar to the following text in the **Output** console:

```
INFO (Runtime): 2020/06/22 22:57:09 build started

INFO (Runtime): Building ....

INFO (Runtime): go build runtime/cgo: copying /Users/myname/Library/Caches/go-
build/f8/.….….d: open /usr/local/go/pkg/darwin_amd64/runtine/
cgo.a: permission denied

ERROR (Runtime): go build runtine/cgo: copying /Users/myname/Library/
Caches/go-build/f8/.….….d: open /usr/local/go/pkg/darwin_amd64/runtime/
cgo.a: permission denied

INFO (Runtime): An error occurred while building: exit status 1
```

This is due to missing permissions for Go. This error has been seen only in Mac OS. This is a known issue.

- https://github.com/golang/go/issues/37962
- https://github.com/golang/go/issues/24674
- https://github.com/udhos/update-golang/issues/15

Solution: Run the following command to change the permissions of your `$GOROOT`, and then try deploying again.

```
sudo chmod -R 777 /usr/local/go
```

Deployment failure
Due to deployment failure, corrupt deployment, a Docker peer container being full, or a Docker peer that was stopped in the local network, you may see an error similar to the following text:

```
============ Started instantiate Chaincode ============
[2028-19-01T19:25:lO.372] [ERROR] default - Error instantiating Chaincode
GollGl on channel mychannel, detailed
error: Error: error starting container: error starting container: Failed to
generate platform-specific docker
build: Failed to pull hyperledger/fabric-ccenv:latest : API error (404):
manifest for hyperledger/
fabric-ccenv:latest not found: manifest unknown: manifest unknown
[2020-19-01T19:25:10.372] (INFO) default -
============ Finished instantiate Chaincode ============
[2020-19-01119:25:10.372] [ERROR] default - Error: Error instantiating
Chaincode Goll01 on channel mychannel,
detailed error: Error: error starting container: error starting container:
Failed to generate platfom-specific
docker build: Failed to pull hyperledger/fabric-ccenv: latest : API error
(404): manifest for hyperledger/
fabric-ccenv:lalest not found: manifest unknown: manifest unknown exited:
signal: terminated
```


Copyright © 2025, Oracle and/or its affiliates.

```
INFO: exited: signal: terminated

ERROR: Error in Chaincode deployment
```

This is due to a peer container that is not able to start up properly again.

Solution: Rebuild your runtime by selecting your local environment in the **Environments** pane, right-clicking and selecting **Rebuild Local Environment**. Attempt to deploy again.

Environment Rebuild Required
You might see an error similar to the following text:

```
Starting ca.example.com ...
Starting orderer.example.com ...
Starting orderer.example.com ... error
ERROR: for orderer.example.com
Cannot start service orderer.example.com:
error while creating mount source path '/host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0/node_modules/@oracle/ochain-cli/
runtime/network/basic-network/config': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
Starting ca.example.com... error
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
ERROR: for orderer.example.com
Cannot start service orderer.example.com: error while creating mount source
path '/host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/config': mkdir /
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0:
operation not permitted
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
Encountered errors while bringing up the project.
ERROR: Starting ca.example.com ...
Starting orderer.example.com ...
Starting orderer.example.com ... error
ERROR: for orderer.example.com
Cannot start service orderer.example.com: error while creating mount source
path '/host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/config': mkdir /
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0:
operation not permitted
Starting ca.example.com ... error
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
```


Copyright © 2025, Oracle and/or its affiliates.

```
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
ERROR: for orderer.example.com
Cannot start service orderer.example.com: error while creating mount source
path '/host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/config': mkdir /
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0:
operation not permitted
ERROR: for ca.example.com
Cannot start service ca.example.com: error while creating mount source path '/
host_mnt/c/Users/opc/.vscode/extensions/oracle.oracle-blockchain-1.4.0/
node_modules/@oracle/ochain-cli/runtime/network/basic-network/crypto-config/
peerOrganizations/org1.example.com/ca': mkdir /host_mnt/c/Users/opc/.vscode/
extensions/oracle.oracle-blockchain-1.4.0: operation not permitted
Encountered errors while bringing up the project.
ERROR: Error in Chaincode deployment
```

You must rebuild your local environment. In the App Builder **Environments** pane, right-click your local environment and select **Rebuild Local Environment**.

Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network

After you've deployed and tested your chaincode project on a local network to ensure it's working as designed, you can deploy it to Oracle Blockchain Platform.

Create a Connection Configuration to an Oracle Blockchain Platform Instance

You must have an Oracle Blockchain Platform instance up and running to complete this step.

1. In the Visual Studio Code **Environments** pane, click the **Create Environment** icon.
2. On the **Environments Details** wizard:
   - Enter the name for your remote environment.
   - Enter a description.
   - In **REST Proxy URL**, enter the REST proxy URL of the remote Oracle Blockchain Platform instance.
   - Enter the Oracle Identity Cloud Service user name and password for an Oracle Blockchain Platform user with the `ADMIN` or `REST_CLIENT` roles. To call the chaincode, only the `REST_CLIENT` role is necessary. To deploy or upgrade the chaincode, the Identity Cloud Service user must also be assigned the `ADMIN` role. For more information about users and roles, see *Set Up Users and Application Roles* in *Administering Oracle Blockchain Platform*.
3. Click **Create** to save the environment. The user name and password are saved temporarily in the local Visual Studio Code session. If you close Visual Studio Code and then start a new Visual Studio Code session, you must enter the user name and password again.

Deploy Your Chaincode

1. Select the chaincode project that you want to deploy in the **Chaincodes** pane.
2. In the **Chaincode Details** pane, select **Deploy**.
3. In the deployment wizard, the name of the chaincode project is already specified.

Copyright © 2025, Oracle and/or its affiliates.

- Select your target environment. For remote deployment, choose the Oracle Blockchain Platform environment that you set up previously.
- Enter the name of the channel that you want to deploy to.
- Optionally, set any required initial parameters.

4. Click **Deploy**.

After the chaincode is successfully deployed to the remote instance of Oracle Blockchain Platform, the console log will show that the following events occurred:

- The Oracle Blockchain Platform details were successfully fetched.
- The list of peers was successfully fetched.
- The chaincode project was successfully installed.
- The chaincode project was successfully approved and committed.
- The chaincode was successfully deployed on each peer and the channel.

In an environment with multiple organizations, to deploy the chaincode again on the same channel through a participant instance, use the console to deploy the chaincode.

Upgrading the Chaincode Project

Upgrading the chaincode is handled automatically by Blockchain App Builder. After you make changes to your chaincode, when you deploy again the project is automatically upgraded. When you run the upgrade process, you can pass an empty array for the `adminList` parameter or use the `adminList` parameter to add token administrators. If you are not the first-time deployer, you must supply an empty array for the `adminList` parameter. To pass an empty array for the `adminList` parameter, open the `Init` parameter list in the deployment pane and then click the minus sign (–) button next to the `adminList` parameter, which will select an empty array.

If your upgrade is successful, the log will show that the following events occurred:

- The Oracle Blockchain Platform details were successfully fetched.
- The list of peers was successfully fetched.
- A check was made to determine if the chaincode project is already installed, and if so, the version was fetched.
- The chaincode version was successfully upgraded (for example, from version 1.0 to 2.0).

In an environment with multiple organizations, to upgrade the chaincode, use the console and manually approve the chaincode from the participants.

Package Your Chaincode Project for Manual Deployment to Oracle Blockchain Platform

You can package your chaincode projects for manual deployment to Oracle Blockchain Platform Cloud or Enterprise Edition.

The **Package** function creates an archive file that contains only the build and distribution files. The `binary`, `libs`, `node_modules`, and `test` folders from your chaincode project are not included. This archive file can be manually uploaded to Oracle Blockchain Platform for deployment.

1. Select your chaincode project in the **Chaincodes** pane.
2. Right-click and select **Package**.

Copyright © 2025, Oracle and/or its affiliates.

3. Select a location to save the package to, and then click **Select Output Folder**.

Because of changes to software prerequisites, when you run the **Package** function for TypeScript chaincode, you are prompted for the provisioning date of the Oracle Blockchain Platform instance that you want to create the package for. The TypeScript chaincode created in Blockchain App Builder is not compatible with previous versions of Oracle Blockchain Platform without changes to the underlying infrastructure. Blockchain App Builder packages the chaincode infrastructure accordingly based on your selection.

When the command completes successfully, the location of the package is returned.

Test Your Chaincode Using Visual Studio Code

If your chaincode is running on a network, you can test any of the generated methods. Additionally, If you chose to create the `executeQuery` method during your chaincode development, you can run SQL rich queries if your chaincode is deployed to an Oracle Blockchain Platform network.

Test Your Chaincode on a Local Hyperledger Fabric Network

After your chaincode project is running on a local network, you can test it.

Blockchain App Builder contains a built-in wizard to assist you with invoking or querying your chaincode.

1. Select your chaincode project in the **Chaincodes** pane. In the **Chaincode Details** pane, select **Execute**. The chaincode name is already selected. Ensure the target environment is set to **Local Environment**, so that the channel defaults to the only channel available.
2. In the **Function** field, select a method from the drop-down list. Every method available in the chaincode is listed.
3. In the **Function Param** field, click **More Actions** button. This launches a window showing available properties for your selected method. Enter the properties, and click **Omit** for any non-mandatory property that you don't want submitted when you call the method. Click **Save**.
4. Click **Invoke**.

The **Output** console window will show that the function has been called. Alternately, in the **Chaincode Actions** pane, the Function Output window shows the output. Click **More Actions** to see this output formatted.

To save the method and parameters that you just ran, click **Save** and enter a name and description. The function is saved in your chaincode project in the `Queries` folder. To use it again, right-click it and select **Open**.

If you make any changes to the controller file that alter the methods, select the **Reload** icon at the top of the **Chaincode Execute** pane. The change will then be reflected in the **Function** drop-down list.

> **Note**
>
> If you don't want to use the wizard for testing, you can also run the Blockchain App Builder command line tools in the Visual Studio Code **Terminal** window. For more information on testing from the command line, see Test Your Chaincode on a Local Hyperledger Fabric Network.

Copyright © 2025, Oracle and/or its affiliates.

Testing Multiple Token Users Locally

To test a token project with multiple users locally, you can use the `tokenUser` property to change the caller of each transaction. Every scaffolded chaincode project includes a `.ochain.json` file, which stores metadata of the chaincode. You change the caller by updating the value of `tokenUser` field in the `.ochain.json` file.

```
{
  "name": "digiCurrCC",
  "description": "Chaincode package for digiCurrCC",
  "chaincodeName": "digiCurrCC",
  "chaincodeType": "node",
  "configFileLocation": "/Users/user1/token.yml",
  "appBuilderVersion": "21.2.3",
  "nodeVersion": "v12.18.1",
  "tokenUser": "admin"
}
```

When a project is scaffolded, the `tokenUser` property is set to the default `admin` user of the local network. To change the caller of a transaction, change the `tokenUser` property to match the `user_id` property that was set when the account was created when the `createAccount` (TypeScript) or `CreateAccount` (Go) method was called.

Automatic Installation and Deployment After Update

Whenever you update and save your chaincode, the changes will be compiled, installed and deployed automatically. There is no need to take down or bring up the local network again. All projects will be automatically compiled and deployed for every change.

Testing Lifecycle Operations on a Remote Oracle Blockchain Platform Network

After your chaincode project has successfully deployed to your remote Oracle Blockchain Platform network, you can test it.

You test chaincode running on a remote Oracle Blockchain Platform network similar to testing it on a local network, as described in Test Your Chaincode on a Local Hyperledger Fabric Network.

You can use the same commands to complete all method transactions on a remote Oracle Blockchain Platform Cloud or Enterprise Edition network; everything supported on the local network is also supported on the remote network. Select the Oracle Blockchain Platform instance as your target environment when running your tests.

Testing Token Projects on a Remote Oracle Blockchain Platform Network

You can test chaincode projects that work with tokens by using Blockchain App Builder, the Oracle Blockchain Platform REST proxy, or the Hyperledger Fabric SDK.

Blockchain App Builder

You can use the Visual Studio Code extension to call transactions with multiple users to test token chaincodes. To test with multiple users, change the authorization parameters (user name and password) in the Environments tab and then save the environment. While invoking transactions, select the same environment from the drop-down list and then run the transaction.

Copyright © 2025, Oracle and/or its affiliates.

Oracle Blockchain Platform REST Proxy

You can use the REST proxy in Oracle Blockchain Platform to run your a token chaincode on a remote Oracle Blockchain Platform network. Use any REST Proxy client, such as Postman REST Client, to test your chaincode project.

To test multiple users, change the authorization parameters (user name and password) in your REST client, or connect to a different instance of Oracle Blockchain Platform.

Execute Berkeley DB SQL Rich Queries

If you chose to create the `executeQuery` method during chaincode development, you can run SQL rich queries if your chaincode is deployed to an Oracle Blockchain Platform network.

If you have used `executeQuery` in the `customMethods` section of the specification file, a corresponding `executeQuery` method will be created in the controller.

Specification file:

```
customMethods:
    - executeQuery
    - "fetchRawMaterial(supplierid: string, rawMaterialSupply: number)"
    - "getRawMaterialFromSupplier(manufacturerId: string, supplierld: string,
rawMaterialSupply: number)"
    - "createProducts(manufacturerId: string, rawMaterialConsumed: number,
productsCreated: number)"
    - "sendProductsToDistribution()"
```

Controller file:

```
**
*
* BDB sql rich queries can be executed in OBP CS/EE.
* This method can be invoked only when connected to remote OBP CS/EE network.
*
*/
@Validator(yup.string())
public async executeQuery(query: string) {
    const result = await OchainController.query(query);
    return result;
}
```

You can call this method to run Berkeley DB SQL rich queries on the Oracle Blockchain Platform network. You must select the Oracle Blockchain Platform environment that you created as your target environment when running the queries.

Example:

1. In the **Chaincode Details** pane, select **Execute**. The chaincode name, target environment, and channel are already specified from the deployment step.
2. In the **Function Name** field, select `executeQuery` from the drop-down list.
3. In the **Function Param** field, click **More Actions**. This launches a window where you can enter the query string. Enter the arguments for your query, and then click **Save**.
4. Click **Query**.

Copyright © 2025, Oracle and/or its affiliates.

The **Output** window opens and shows the query being run, and the result.

```
ochain query executeQuery "SELECT key, valueJson FROM <STATE> WHERE
json_extract(valueJson, '$.rawMaterialAvailable') = 4"
```

The entire SQL query is taken in the argument, so you can make changes to your query on the fly.

Generate CLI Commands from Queries

If you have saved queries in a chaincode project in Visual Studio Code, you can automatically generate the equivalent CLI commands.

You must have at least one saved query in a chaincode project to generate CLI commands for macOS and Linux and for Microsoft Windows.

1. Expand the project in the **Chaincodes** pane.
2. Right-click **Queries**.
3. Click **Generate CLI Commands**.

Two text files are generated and displayed in the **Queries** section of the **Chaincodes** pane: `CLIcommandsForLinux.txt` and `CLIcommandsForWindows.txt`. Select the file name to open the file and show the corresponding CLI commands.

Upgrading Chaincode Projects in Visual Studio Code

You can upgrade existing chaincode projects in Visual Studio Code to use the new features of the updated Blockchain App Builder.

For Go projects, upgrade to a supported version of Go before you run the command to upgrade your chaincode project. Blockchain App Builder supports Go v1.23.2 or later, but not Go v1.24 or later.

To upgrade a chaincode project, right-click the project in the Oracle Blockchain Platform pane of Visual Studio Code, and then click **Upgrade**.

When you open the detail view of a chaincode project from a previous version of Blockchain App Builder, you are prompted with the following message: `New chaincode library is available. Would you like to upgrade`? You can select from three options:

- **Yes**. The chaincode project is upgraded. The chaincode files in the `lib` folder are replaced. If you made any changes to these library files, back up the modified files or track the changes that you made before you run the upgrade.
- **Later**. Upgrade notification is postponed for 24 hours. You are notified again after 24 hours.
- **No**. You are not prompted again to upgrade. You can still upgrade the project at any time by right-clicking and selecting **Upgrade**, as described previously.

After you upgrade a chaincode project, synchronize the specification file with the generated source code. For more information, see Synchronize Specification File Changes With Generated Source Code.

Copyright © 2025, Oracle and/or its affiliates.

Synchronize Specification File Changes With Generated Source Code

You can use the synchronization function to bring new changes from the specification file to the chaincode source files (model and controller). The function works with both TypeScript and Go projects.

Note:

- Synchronization is unidirectional: you can bring changes from your specification file into your chaincode project, but not the other way around. Changes made in your chaincode project remain as-is after the synchronizing process.
- The command works only if the chaincode project was scaffolded by using a specification file. Do not delete, rename or move the specification file if you plan to synchronize any changes from the specification file to the source code in future.
- During synchronization, the chaincode files in the `lib` folder are automatically upgraded. If you make any changes to these library files, back up the modified files or track the changes that you make before you use the synchronization function, so that you can apply those changes again after synchronization.

To synchronize your specification and chaincode files:

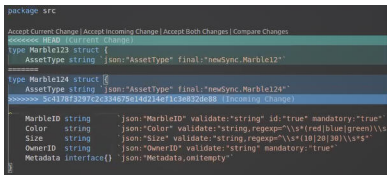
1. In the **Specifications** pane, select the specification file that you updated to open its **Specification Details** pane. At the top of the pane, click **Chaincodes** to open the pane showing which chaincodes were generated from the specification file.
2. Select the **Sync** checkbox beside each chaincode that you want to update with the new changes. You can synchronize more than one chaincode that was generated from a specification file at a time. Click **Synchronize**.

The chaincode projects now contain updated files.

Resolving Conflicts

Because you can edit both the synchronization files and chaincode files, it's possible to end up with conflicts where the updated specification file could overwrite a change that you've made to the chaincode file. In these cases, when you attempt to synchronize an error is displayed stating that there's a conflict. You can use the **Conflicts** pane to resolve these errors.

1. On the **Conflicts** pane, click the name of the chaincode file where the conflicts exist. The file opens in an editor with the conflicts highlighted.



   In the example shown, `Marble124` is in the specification file, and `Marble123` is in the chaincode model file.

2. Above the conflict is a list of options. Click **Accept Current Change** to override the specification file and use what is currently in the chaincode file. Click **Accept Incoming Change** to override the chaincode file and use what is currently in the specification file.

Copyright © 2025, Oracle and/or its affiliates.

3. Return to the **Conflicts** pane. Select the **Sync** checkbox next to the conflict name, and then click **Confirm Changes**. If you have multiple conflicts, resolve all of them before before clicking **Confirm Changes**.

Debugging from Visual Studio Code

Blockchain App Builder includes line-by-line debug support from Visual Studio Code for both TypeScript and Go projects.

On Microsoft Windows, configure Visual Studio Code to use Command Prompt as the default terminal instead of PowerShell. In the terminal menu in Visual Studio Code, click **Select Default Profile**, and then select **Command Prompt**.

Before you can debug your Go chaincode project, you must install the required Go tools in Visual Studio Code. In Visual Studio Code, open the Command Palette and then run the `Go: Install/Update Tools` command. Install all of the Go extensions that are listed.

Visual Studio Code uses Delve to debug Go. When you debug a Go chaincode for the first time, you will be prompted to install Delve. Accept the Delve installation before continuing. Visual Studio Code includes a built-in debugger for TypeScript.

To run line-by-line debugging:

1. Open your chaincode project in Visual Studio Code Explorer. In the **Chaincodes** pane, right-click your chaincode and select **Open in Explorer**.
2. Attach breakpoints to your code wherever necessary.
3. Go to the **Run** menu and click **Start Debugging**. This attaches the debugger. It may take several seconds for the debugger to attach to the chaincode.
4. Call any command from the Terminal that would execute your code.
   If you've been using the Visual Studio Code interface to test your chaincode so far, you can follow the invocation syntax outlined in Test Your Chaincode on a Local Hyperledger Fabric Network.

   The debugger will stop at your breakpoints. You can then start debugging.
5. Restart debugging to reflect new changes.

Because the chaincode is running in debug mode, the hot deployment of new changes does not happen automatically. You must manually restart the debugging process, using the debug controls in Visual Studio Code, in order to make the latest changes take effect.

Troubleshooting

On Windows 11, you might encounter an error similar to the following text:

```
dlv: failed to install dlv(github.com/go-delve/delve/cmd/dlv@latest): Error:
Command failed:
C:\Program Files (x86)\Go\bin\go.exe get -x github.com/go-delve/delve/cmd/
dlv@latest
# get https://proxy.golang.org/github.com/go-delve/delve/cmd/dlv/@v/list
# get https://proxy.golang.org/github.com/@v/list
# get https://proxy.golang.org/github.com/go-delve/@v/list
# get https://proxy.golang.org/github.com/go-delve/delve/cmd/@v/list
# get https://proxy.golang.org/github.com/go-delve/delve/@v/list
# get https://proxy.golang.org/github.com/@v/list: 410 Gone (0.420s)
# get https://proxy.golang.org/github.com/go-delve/delve/cmd/@v/list: 410
Gone (1.040s)
```


Copyright © 2025, Oracle and/or its affiliates.

```
# get https://proxy.golang.org/github.com/go-delve/@v/list: 410 Gone (1.062s)
# get https://proxy.golang.org/github.com/go-delve/delve/cmd/dlv/@v/list: 410
Gone (1.066s)
# get https://proxy.golang.org/github.com/go-delve/delve/@v/list: 200 OK
(1.448s)
go: found github.com/go-delve/delve/cmd/dlv in github.com/go-delve/delve
v1.8.3C:\Users\<UserName>\go\pkg\mod\github.com\go-
delve\delve@v1.8.3\service\debugger\debugger.go:28:2:found packages native
(proc.go) and
your_operating_system_and_architecture_combination_is_not_supported_by_delve(s
upport_sentinel.go) in C:\Users\Asus\go\pkg\mod\github.com\go-
delve\delve@v1.8.3\pkg\proc\native
```

There is no workaround for this error at this time.

Generate a Postman Collection Using Visual Studio Code

You can create a Postman collection that includes example payloads for all of your chaincode controller APIs.

Postman is a tool that you can use to work with and test REST APIs. The generate command creates a Postman collection that is based on the chaincode that was automatically generated from a declarative specification file. The Postman collection contains the payloads for all of the methods that are specified in the chaincode controller file. You can change the variable values in the Postman collection file to make REST API calls.

The generated Postman collection includes default values for all APIs in the controller. To learn more about Postman, see https://www.postman.com/. After you generate a Postman collection, you can directly import it and use it by changing the default values in the payload and variables.

To generate a Postman collection for a chaincode project in Visual Studio Code, complete the following steps.

1. Select the chaincode project in the Chaincodes pane.
2. Right-click the chaincode name and then select **Generate Postman Collection**.
3. Select a location to save the Postman collection to, and then click **Select Output Folder**.

If the specified Postman collection already exists, you are prompted whether to overwrite it.

Postman Collection Structure

The generated Postman collection includes two types of requests, invoke requests and query requests:

- Invoke requests include all write operations, which use the endpoint `/transactions`
- Query requests include all get operations, which use the endpoint `/chaincode-queries`

To differentiate between getter and non-getter methods in the controller APIs, a decorator is used in TypeScript chaincodes and a comment is used in Go chaincodes. If you define a getter method in the controller, you must use the GetMethod decorator for TypeScript or the GetMethod comment for Go, as shown in the following table.

Copyright © 2025, Oracle and/or its affiliates.

TypeScript	Go

Every getter method has a `GetMethod` decorator:
 	
`@GetMethod()`
`@Validator()`
`public async getAllTokenAdmins() {`
`  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ADMIN.getAllAdmins", "TOKEN");`
`  return await this.Ctx.ERC1155Admin.getAllAdmins();`
`}`
	Every getter method has a `GetMethod` comment block:
	
	`/**`
	` * GetMethod`
	` */`
	`func (t *Controller) GetAllTokenAdmins() (interface{}, error) {`
	`  auth, err := t.Ctx.Auth.CheckAuthorization("Admin.GetAllAdmins", "TOKEN")`
	`  if err != nil && !auth {`
	`  return nil, fmt.Errorf("error in authorizing the caller  %s", err.Error())`
	`  }`
	`  return t.Ctx.Admin.GetAllTokenAdmins()`
	`}`

Generated Postman collections include variables with default values, as shown in the following table.

Variable Name	Description	Default Value	Context
`bc-url`	The REST proxy URL of the Oracle Blockchain Platform instance where the chaincode is deployed	`https://test-xyz-abc.blockchain.ocp.oraclecloud.com:7443/restproxy`	all chaincodes
`bc-channel`	The channel where the chaincode is deployed	`default`	all chaincodes
`bc-admin-user`	The name of the administrator (a user with the `admin` role that can access all POST requests). By default, this user is the caller of all POST requests in the chaincode	`bc-admin-user value`	all chaincodes
`bc-admin-password`	The password for the administrator user	`bc-admin-password value`	all chaincodes
`bc-timeout`	The timeout value in the body of every POST request to indicate the timeout interval	`6000`	all chaincodes
`bc-sync`	The sync value in the body of every POST request to indicate whether the request is synchronous or asynchronous	`true`	all chaincodes

Copyright © 2025, Oracle and/or its affiliates.

Variable Name	Description	Default Value	Context
`bc-chaincode-name`	The chaincode name, which is used in every POST request	`chaincode name`	all chaincodes
`bc-org-id`	The default `orgId` parameter for all POST requests	`bc-org-id value`	token chaincodes only
`bc-user-id`	The default `userId` parameter for all POST requests	`bc-user-id value`	token chaincodes only
`bc-token-id`	The default `tokenId` parameter for all POST requests	`bc-token-id value`	token chaincodes only

In every generated request, all of the parameters with default values are generated. Functions that have struct/class parameters will have a placeholder object in the request body, as shown in the following examples.

API with a struct/class parameter

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "CreateArtCollectionToken",
        "{\"TokenId\":\"{{bc-token-id}}\",\"TokenDesc\":\"TokenDesc
value\",\"TokenUri\":\"TokenUri value\",\"TokenMetadata\":
{\"Painting_name\":\"Painting_name value\",\"Description\":\"Description
value\",\"Image\":\"Image value\",\"Painter_name\":\"Painter_name
value\"},\"Price\":999,\"On_sale_flag\":true}",
        "quantity value"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}}
}
```

API without a struct/class parameter

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "CreateAccount",
        "{{bc-org-id}}",
        "example_minter",
        "true",
        "true"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}}
}
```

The default value for most API parameters is *`parameter_name`* `value`, with some exceptions. The following examples show some of the exceptions.

Copyright © 2025, Oracle and/or its affiliates.

- The filters parameter in `GetAccountTransactionHistoryWithFilters`:

```
"{\"PageSize\":20,\"Bookmark\":\"\",\"StartTime\":\"2022-01-16T15:16:36+00:00\",\"EndTime\":\"2022-01-17T15:16:36+00:00\"}"
```

- The filters parameter in `GetSubTransactionsByIdWithFilters`:

```
"{\"PageSize\":20,\"Bookmark\":\"\"}"
```

A struct or class has different default values, as shown in the following table:

Data Type	Default Value
`boolean/bool`	`true`
`int/number`	`999`
`date`	`2022-01-16T15:16:36+00:00`
`other`	*`parameter_name`* `value`

ERC-1155 Token Projects

The ERC-1155 standard includes common methods for both fungible and non-fungible tokens. The generated Postman collection for an ERC-1155 project that uses both fungible and non-fungible tokens includes two different POST requests, one for each type of token, for these common methods. If an ERC-1155 project uses only fungible or non-fungible tokens but not both types, then the generated Postman collection includes only one POST request for these common methods. The following table illustrates the generated API for the `AddRole` method.

Request Element	Fungible Tokens	Non-Fungible Tokens
Request Name	`AddRole -For Fungible`	`AddRole -For NonFungible`

Request Body	`{`
	`"chaincode": "{{bc-chaincode-name}}",`
	`"args": [`
	`"AddRole",`
	`"{{bc-org-id}}",`
	`"{{bc-user-id}}",`
	`"role value (for example, minter / burner)",`
	`"{\"TokenId\":\"{{bc-token-id}}\"}"`
	`],`
	`"timeout": {{bc-timeout}},`
	`"sync": {{bc-sync}}`
	`}`
		`{`
		`"chaincode": "{{bc-chaincode-name}}",`
		`"args": [`
		`"AddRole",`
		`"{{bc-org-id}}",`
		`"{{bc-user-id}}",`
		`"role value (for example, minter / burner)",`
		`"{\"TokenName\":\"TokenName value\"}"`
		`],`
		`"timeout": {{bc-timeout}},`
		`"sync": {{bc-sync}}`
		`}`

Copyright © 2025, Oracle and/or its affiliates.

Troubleshoot Blockchain App Builder Visual Studio Code Extension

Use the following information to troubleshoot system problems with the Blockchain App Builder Visual Studio Code extension.

Prerequisites issues
Errors can occur if you modify or upgrade any of the prerequisite software that is required by Blockchain App Builder. You can use the **Installed Dependencies** function to check that your installation of Blockchain App Builder still meets the prerequisites. On the Blockchain App Builder welcome page in Visual Studio Code, click **Installed Dependencies**. A table is displayed that shows you the required version and the actual installed version of components that Blockchain App Builder uses. Required components are indicated by an asterisk (*).

If the prerequisites check fails with errors and warnings when you attempt to install Blockchain App Builder, you might see an error similar to the following example in the output pane in Visual Studio Code:

```
Error:
Aborting installation. Error:
Found 1 error(s) in pre-requisites check, failed with following errors:
1. Golang version mismatch. Expected 1.21.x, but found 1.18.5.

Found 3 warning(s) in pre-requisites check.
1. Docker is not installed. Please install Docker >= 18.09.0. To deploy
chaincodes in the local environment,
please install the recommended version of Docker.
2. Docker Compose is not installed. Please install Docker Compose >= 1.23.0.
To deploy chaincodes in the
local environment, please install the recommended version of Docker Compose.
3. Git is not installed. To sync chaincodes, please install the Git according
to the documentation.
```

Deployment failure
Due to deployment failure, corrupt deployment, a Docker peer container being full, or a Docker peer that was stopped in the local network, you may see an error similar to the following text:

```
============ Started instantiate Chaincode ============
[2028-19-01T19:25:lO.372] [ERROR] default - Error instantiating Chaincode
GollGl on channel mychannel, detailed
error: Error: error starting container: error starting container: Failed to
generate platform-specific docker
build: Failed to pull hyperledger/fabric-ccenv:latest : API error (404):
manifest for hyperledger/
fabric-ccenv:latest not found: manifest unknown: manifest unknown
[2020-19-01T19:25:10.372] (INFO) default -
============ Finished instantiate Chaincode ============
[2020-19-01119:25:10.372] [ERROR] default - Error: Error instantiating
Chaincode Goll01 on channel mychannel,
detailed error: Error: error starting container: error starting container:
Failed to generate platfom-specific
```


Copyright © 2025, Oracle and/or its affiliates.

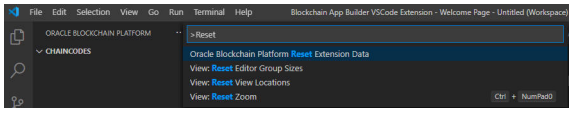
```
docker build: Failed to pull hyperledger/fabric-ccenv: latest : API error
(404): manifest for hyperledger/
fabric-ccenv:lalest not found: manifest unknown: manifest unknown exited:
signal: terminated
INFO: exited: signal: terminated

ERROR: Error in Chaincode deployment
```

This is due to a peer container that is not able to start up properly again.

Solution: Rebuild your runtime by selecting your local environment in the **Environments** pane, right-clicking and selecting **Rebuild Local Environment**. Attempt to deploy again.

Resetting Extension Data
Blockchain App Builder user data can get corrupted. Resetting the extension data clears your data from Blockchain App Builder without affecting anything stored locally.

1. Open the Command Palette from the **View** menu.
2. In the Command Palette, type `Reset Extension`.



3. Select **Oracle Blockchain Platform Reset Extension Data**. Visual Studio Code clears the existing blockchain data and reloads the default installation data. This does not affect the files stored locally in your system, but you must import them back into Visual Studio Code and reconfigure any environments you had previously set up.

macOS: Xcode
After a macOS upgrade, or if Xcode is not installed, you might see an error similar to the following in the error log:

```
gyp: No Xcode or CLT version detected!
gyp ERR! configure error
gyp ERR! stack Error: `gyp` failed with exit code: 1
gyp ERR! stack     at
```

- To work around this behavior, open a terminal window and run the following commands:

  ```
  sudo rm -rf $(xcode-select --print-path)
  xcode-select --install
  ```


Copyright © 2025, Oracle and/or its affiliates.

4

Tokenization Support Using Blockchain App Builder

You can use Blockchain App Builder to manage the complete life cycle of a token. You can tokenize existing assets and automatically generate token classes and methods to use for token lifecycle management.

Tokenization

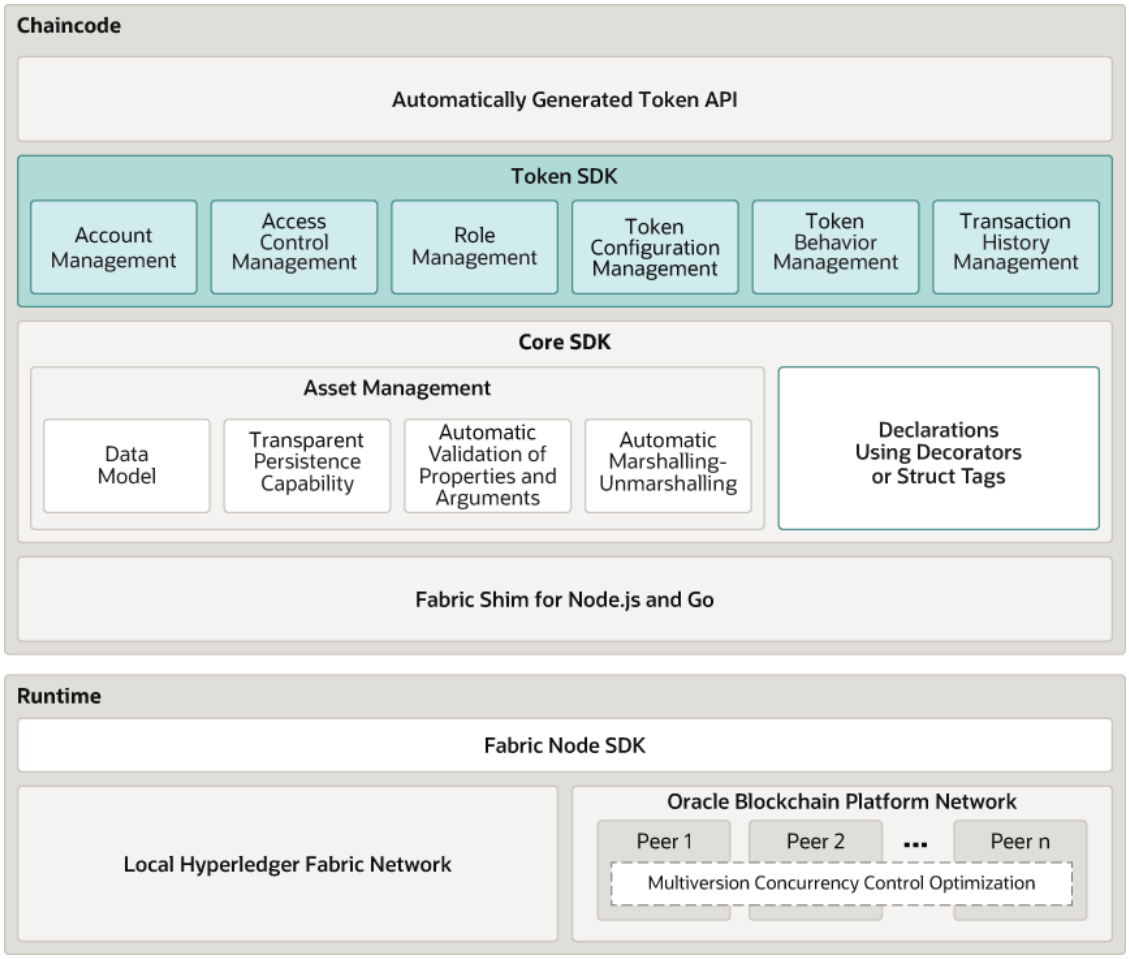
Tokenization is a process where physical or digital assets are represented by tokens, which can be transferred, tracked, and stored on a blockchain.

By representing assets as tokens, you can use the blockchain ledger to establish the state and ownership of an asset, and use standard blockchain platform functions to transfer ownership of an asset.

Blockchain App Builder includes tokenization support: token classes and methods are automatically generated, and additional token methods are provided so that developers can create complex business logic for tokens. The automatically generated project contains token lifecycle classes and functions, CRUD methods, and additional token SDK methods, and supports automatic validation of arguments, marshalling/unmarshalling, and transparent persistence capability. You can use these controller methods to initialize tokens, control access, set up accounts, manage roles, and manage the life cycle of tokens.

Copyright © 2025, Oracle and/or its affiliates.

The following diagram shows the token architecture implemented by Blockchain App Builder, including the token API and token SDK.



Automatically Generated Token API
Blockchain App Builder automatically generates methods to support tokens and token life cycles. You can use these methods to initialize tokens, manage roles and accounts, and complete other token lifecycle tasks without any additional coding.

Token SDK
The Token SDK includes methods that help you develop complex business logic for token applications.

Multiversion Concurrency Control (MVCC) Optimization
The MVCC optimization for token chaincode can reduce errors for transfer, mint, burn, and hold operations.

Tokens and the Account/Balance Model

Blockchain App Builder supports fungible and non-fungible tokens.

Fungible tokens have an interchangeable value. Any quantity of fungible tokens has the same value as any other equal quantity of the same class of token. Non-fungible tokens are unique. Tokens can also be either whole or fractional. Fractional tokens can be subdivided into smaller parts, based on a specified number of decimal places.

Tokens can also be described by behaviors. Supported behaviors for fungible tokens include: `mintable`, `transferable`, `divisible`, `holdable`, `burnable`, and `roles` (`minter`, `burner`, and

Copyright © 2025, Oracle and/or its affiliates.

holder). Supported behaviors for non-fungible tokens include: mintable, transferable, singleton, indivisible, burnable, and roles (minter and burner).

The tokenization feature uses an account/balance model to represent tokenized assets as balances in an account. Accounts are similar to typical banking accounts, where deposits and transfers and other state transitions affect the balance of an account. The balance of every account is tracked globally, to ensure that transaction amounts are valid. The on-hold balance (for fungible tokens) and transaction history are also tracked.

Any user who possesses tokens or completes token-related operations at any point must have an account on the network. Every account is identified by a unique ID (account_id). The account ID is created by combining a user name or email ID (user_id) of the instance owner or the user who is logged in to the instance with the membership service provider ID (org_id) of the user in the current network organization. Ready-to-use methods are provided for account creation. Because the account ID includes the organization ID, users can be supported across multiple organizations.

Token Standards

Blockchain App Builder extends the standards and classifications of the Token Taxonomy Framework, the ERC-721 standard, and the ERC-1155 standard to define the anatomy and behavior of tokens.

ERC-1155 is a standard that supports both fungible and non-fungible tokens (NFTs). ERC-721 is a standard for NFTs. The Token Taxonomy Framework was developed by the Token Taxonomy Initiative. For more information, see Token Taxonomy Framework.

The following table describes the token types that Blockchain App Builder supports:

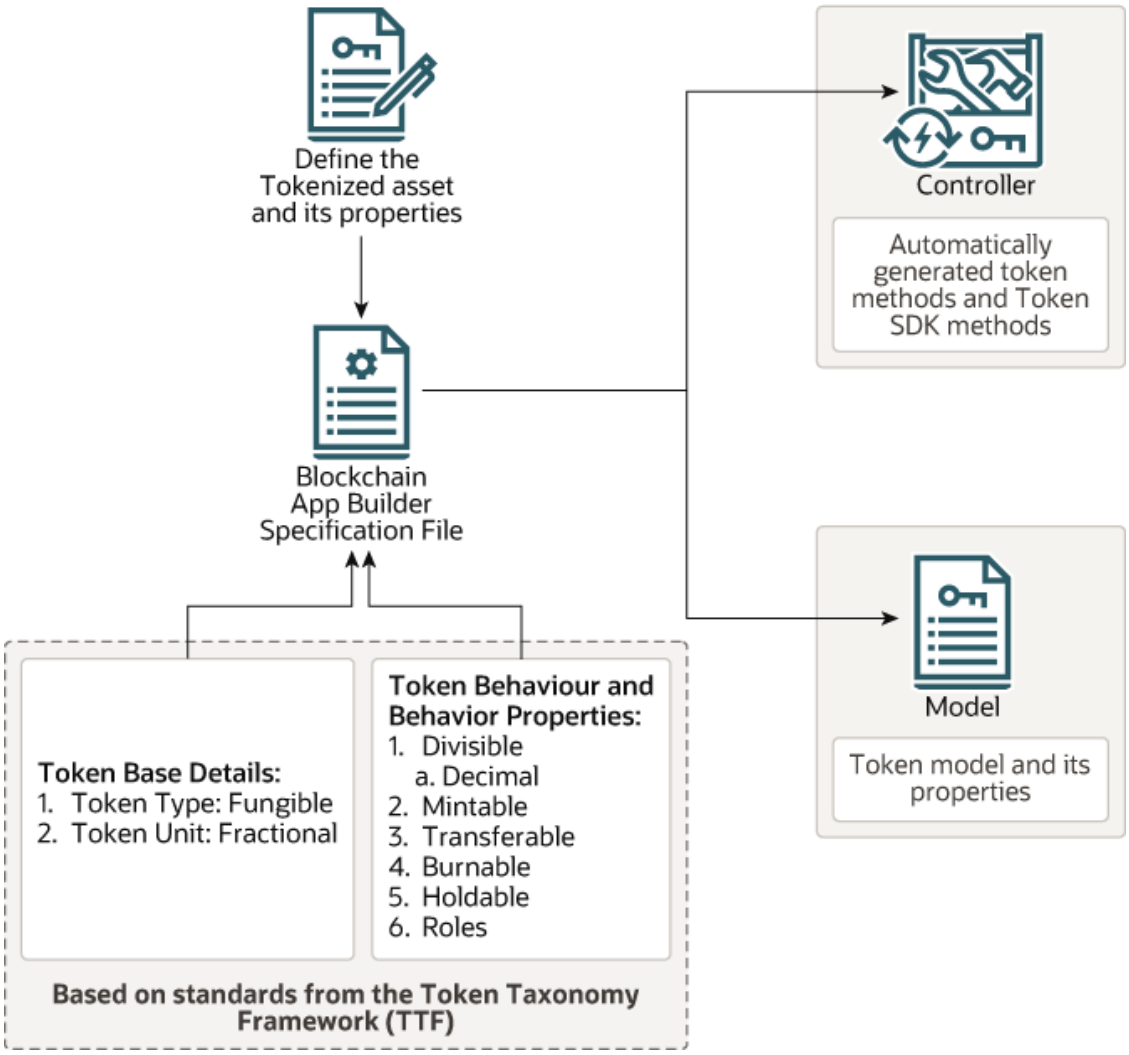
Standard	Supported Token Types
Token Taxonomy Framework	• fractional fungible tokens
ERC-721	• whole non-fungible tokens
ERC-1155	• whole fungible tokens • fractional fungible tokens • whole non-fungible tokens • fractional non-fungible tokens

Tokenization Flow

Because Blockchain App Builder supports tokenization by extending the input specification file syntax, you create token-specific projects the same way that you create other projects, either by using the CLI or in Visual Studio Code.

For more information about creating projects with Blockchain App Builder, see Input Specification File.

Copyright © 2025, Oracle and/or its affiliates.



A typical tokenization flow follows these basic steps:

- Decide which token standard to use.
- Decide what token behaviors to specify (`mintable`, `transferable`, `divisible`, `indivisible`, `singleton`, `holdable`, `burnable`, and `roles`).
- Define the token asset and its properties in the input specification file.
- Scaffold the chaincode project from the input specification file. This creates a scaffolded project, including a model that contains the token asset definition and its properties and a controller that contains the token's behavior and methods.
- Deploy and test the chaincode project.

After you deploy a token-based project, the typical flow for creating tokens and completing lifecycle operations follows these steps:

- A token chaincode is deployed; the users in the list passed to the initialization method become `Token Admin` users of the chaincode.
- A tokenized asset is initialized, which creates the `token_id`, a unique identifier for that particular instance of token.
- Accounts must be created for every user who will possess tokens or complete token-related operations.

Copyright © 2025, Oracle and/or its affiliates.

- If the `roles` behavior is specified for the token, then roles must be assigned to users before they can complete token-related operations.
- Token lifecycle methods can then be used, based on the behaviors that were specified for the token asset. For example, you can call a method to mint tokens for an account.

Access Control

Tokenization support includes an access control feature that supports both role-based and ownership-based control mechanisms.

With role-based control, users can call specific methods with an associated role such as `Token Admin` or `Token Minter`. With ownership-based control, you can restrict users from accessing assets that they do not own. With ownership-based access control, specific methods can be called by the users who own the assets, such as the `Token Owner` or `Account Owner`. For specific information on access control for methods, see the individual entries for the methods documented in the following topics:

- Scaffolded TypeScript Token Project for ERC-1155
- Scaffolded Go Token Project for ERC-1155
- Scaffolded TypeScript NFT Project for ERC-721
- Scaffolded Go NFT Project for ERC-721
- Scaffolded TypeScript Project for Token Taxonomy Framework
- Scaffolded Go Project for Token Taxonomy Framework

Role-based access control supports the following personas:

Token Admin
`Token Admin` users can be assigned when a token chaincode is deployed. The `Token Admin` user information is saved in the state database. A `Token Admin` user can grant and remove `Token Admin` privileges for other users. A `Token Admin` user cannot remove their own `Token Admin` privileges. A `Token Admin` user can assign the `Org Admin`, minter, burner, or notary role to any user.

Org Admin
The extended Token Taxonomy Framework methods support the `Org Admin` role. A `Token Admin` user can assign the `Org Admin` role to any user. `Org Admin` users have administrative privileges, but only within their organization. They can create accounts or see account balances, but only for users in their organization. `Org Admin` users who have a minter, burner, or notary role can assign that role to other users in their organization.

Token Minter
A user who is assigned the minter role is a `Token Minter`, and can mint tokens.

Token Burner
A user who is assigned the burner role is a `Token Burner`, and can burn tokens.

Token Notary
A user who is assigned the notary role is a `Token Notary`. A `Token Notary` acts as a third party in transactions between payers and payees. A `Token Notary` can either trigger the completion of a token transfer from a payer to a payee, or can instead return the tokens to the payer's account.

Copyright © 2025, Oracle and/or its affiliates.

Vault Manager
A user who is assigned the vault role is the `Vault Manager`. The `Vault Manager` can unlock a locked NFT. The vault role is applicable only for the extended ERC-721 and ERC-1155 standards. Assigning the vault role to a user is a prerequisite for locking NFTs. Only one user per chaincode can be assigned the vault role.

Token Auditor
Digital Assets Edition only: The extended Token Taxonomy Framework methods support the `Token Auditor` role. This role is similar to the `Token Admin` role, but with read access only.

Org Auditor
Digital Assets Edition only: The extended Token Taxonomy Framework methods support the `Org Auditor` role. This role is similar to the `Org Admin` role, but with read access only.

Ownership-based access control supports the following personas:

Account Owner
Any user that has an account is an `Account Owner`.

Token Owner
Any user that currently owns a non-fungible token is the `Token Owner` of that token.

Role-based access control and ownership-based access control are also combined in some methods. For example, role-based access control lets a user with the minter role create tokens. With ownership-based access control, a non-fungible token owner can modify the custom properties of a token, but cannot modify the token metadata. When a user with the minter role creates a non-fungible token (NFT), they become the owner of the NFT. As the owner of that NFT, they can modify the custom properties (for an art collection token, the token price is a custom property). After the token creator transfers the NFT to another user, the second user becomes the owner, and the user who created the token is no longer the owner of the token. Because of ownership-based access control, the new owner can now update a custom property value, but the previous owner no longer can. Because of role-based access control, the previous owner can still mint an NFT, but the new user cannot.

You can also write your own access control functions, or disable access control. The automatically generated code that controls access is shown in the following examples.

TypeScript:

```
await this.Ctx.<Token Standard>Auth.checkAuthorization(...)
```

Go:

```
auth, err := t.Ctx.<Token Standard>Auth.CheckAuthorization(...)
```

> **Note**
>
> To remove the automatically generated access control function, remove the previous line of code from your TypeScript or Go project.

MVCC Optimization

Hyperledger Fabric databases use multi-version concurrency control (MVCC) to avoid double-spending and data inconsistency.

Copyright © 2025, Oracle and/or its affiliates.

When the same state is updated, a new version of the record overwrites the old version. If there are concurrent requests to update the same key in a block, an `MVCC_READ_CONFLICT` error might be generated.

To reduce MVCC errors for transfer, mint, burn, and hold operations, you can enable the MVCC optimization for token chaincode. This optimization works on Oracle Blockchain Platform only. By default, the optimization is disabled. To enable the optimization, complete the applicable following step.

- CLI: Specify the Boolean `-m` or `--enable_mvcc_optimization` parameter with the `ochain init` command. By default, the parameter is set to `false`. To enable the optimization, add `-m true` to the `ochain init` command line.
- Visual Studio Code: When you create a chaincode, select **Enable MVCC optimization** on the Create Chaincode window.

To use the optimization with chaincode created in previous versions of Blockchain App Builder, complete the following steps:

1. After you install the latest version of Blockchain App Builder, upgrade the chaincode as described in Upgrading Chaincode Projects in the CLI and Upgrading Chaincode Projects in Visual Studio Code.
2. Edit the `.ochain.json` file in the root folder of the chaincode to set `enableMvccOptimization` to `true`.
3. Synchronize the chaincode, which adds the optimization and creates two new folders in the root folder of the the chaincode: `statedb` and `tokens`. For more information about synchronization, see Synchronize Specification File Changes With Generated Source Code and Synchronize Specification File Changes With Generated Source Code.

Other methods to work around `MVCC_READ_CONFLICT` errors include having the client application retry requests when this error is generated, or using a queue to capture concurrent requests before they are sent to the blockchain network. For more information, see Read-Write set semantics in the Hyperledger Fabric documentation.

> **Note**
>
> The MVCC optimization does not work on hybrid networks that include both Oracle Blockchain Platform and Hyperledger Fabric peers, or when testing on a local Hyperledger Fabric network. Do not enable the MVCC optimization on hybrid networks, as this might lead to inconsistencies between peers.

Token Taxonomy Framework

Blockchain App Builder supports an extended version of the Token Taxonomy Framework to work with fractional fungible tokens.

- Input Specification File for Token Taxonomy Framework
- Scaffolded TypeScript Project for Token Taxonomy Framework
- Scaffolded Go Project for Token Taxonomy Framework

Copyright © 2025, Oracle and/or its affiliates.

Input Specification File for Token Taxonomy Framework

The Blockchain App Builder initialization command reads the input specification file and generates the scaffolded project with several tools to assist in the chaincode development process.

You can define standard assets and token assets that are based on the Token Taxonomy Framework in the same specification file. You cannot define token assets based on more than one standard in the same specification file.

For information on including standard assets in the specification file, see Input Specification File.

The following sample specification files for fungible token assets are available in the Blockchain App Builder package:

- `FiatMoneyToken.yml`
- `LoyaltyToken-Go.yml`

In addition to the standard properties and sections, fungible token assets support the `behavior` and `anatomy` sections in the specification file. Fungible token assets also support the `standard` property. The following example shows the structure of a specification file for a fungible token asset based on the Token Taxonomy Framework:

```
assets:
    - name: OBPTOK # Asset name
      type: token  # Asset type

      anatomy:
          type: fungible # Token type
          unit: fractional # Token unit

      behavior: # Token behaviors
          - divisible:
                decimal: 2
          - mintable:
                max_mint_quantity: 1000
          - transferable
          - burnable
          - roles:
                minter_role_name: minter

      properties:
          - name: currency_name # Custom attribute to represent the token in
certain currency. This attribute is helpful for exchanging the tokens with
fiat currency.
            type: string

          - name: token_to_currency_ratio # Custom attribute to specify the
token to currency ratio. This attribute is helpful for exchanging the tokens
with fiat currency.
            type: number
```


Copyright © 2025, Oracle and/or its affiliates.

Table 4-1 Parameter Descriptions and Examples for a Fungible Token Specification File

Entry	Description	Examples

`type:`	You must specify `type: token` in the `assets` section.	`assets:`
		`  - name: OBPTOK # Asset name`
		`  type: token  # Asset type`

`standard:`	The `standard` property represents the token standard to follow during chaincode generation. Only the `ttf+` value is supported for fungible tokens. If the `standard` property is not specified for a fungible token, the Token Taxonomy Framework (TTF) standard is followed.	`standard: ttf+ # Token standard`

`anatomy:`	The `anatomy` section has two mandatory parameters for fungible tokens:
	• `type: fungible` A quantity of fungible tokens has the same value as another equal quantity of the same class of tokens.
	• `unit: fractional` A fractional token can be subdivided into smaller units based on a specified number of decimal places.
		`anatomy:`
		`  type: fungible # Token type`
		`  unit: fractional # Token unit`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-1 (Cont.) Parameter Descriptions and Examples for a Fungible Token Specification File

Entry	Description	Examples

`behavior:`	This section describes the capabilities and restrictions of the token. The `mintable` and `transferable` behaviors are mandatory for fungible tokens.
	• `mintable`: This mandatory behavior supports minting new token instances. The optional `max_mint_quantity` parameter specifies the total number of tokens that can be minted. If you do not specify the `max_mint_quantity` parameter, any number of tokens can be minted.
	• `transferable`: This mandatory behavior supports transferring ownership of tokens.
	• `divisible`: This optional behavior describes how tokens can be subdivided. The `decimal` parameter specifies the number of decimal places that can be used. The smallest fraction possible with the number of decimal places is the smallest unit of the token that can be owned. If the `decimal` parameter is not specified, the default is zero decimal places.
		`behavior:`
		`- mintable:`
		`max_mint_quantity: 20000`
		`- transferable`
		`- divisible:`
		`decimal: 1`
		`- burnable`
		`- holdable`
		`- roles:`
		`minter_role_name: minter`
		`burner_role_name: burner`
		`notary_role_name: notary`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-1 (Cont.) Parameter Descriptions and Examples for a Fungible Token Specification File

Entry	Description	Examples
	• `burnable`: This optional behavior supports deactivating, or burning, tokens. Burning does not delete a token but instead places it in a permanent state where it cannot be used. Burning is not reversible. • `holdable`: This optional behavior indicates whether token balances can be put on hold between a payer and payee. • `roles`: This optional behavior restricts token behaviors to users with specific roles. Currently three roles are available: `minter_role_name`, `burner_role_name`, and `notary_role_name`. If you do not specify roles, then any user can act as a minter, burner, or notary. For example, if the burner role is not specified, any account user implicitly has the burner role. If the burner role is specified, then during the token setup process, the `Token Admin` user must assign the burner role to other users explicitly.	

To create multiple fungible token IDs that use different `max_mint_quantity` parameters, create a separate token asset for each token ID in the specification file, with a 1:1 relationship between token asset and token ID.

Copyright © 2025, Oracle and/or its affiliates.

To create multiple fungible token IDs that use the same `max_mint_quantity` parameter or no `max_mint_quantity` parameter, create a single token asset in the specification file to use for all of the token IDs.

Scaffolded TypeScript Project for Token Taxonomy Framework

Blockchain App Builder takes the input from your token specification file and generates a fully-functional scaffolded chaincode project.

The project automatically generates token lifecycle classes and functions, including CRUD and non-CRUD methods. Validation of arguments, marshalling/unmarshalling, and transparent persistence capability are all supported automatically.

For information on the scaffolded project and methods that are not directly related to tokens, see Scaffolded TypeScript Chaincode Project.

Reference:

- Model
- Controller
  - Automatically Generated Token Methods
  - Custom Methods
- Token SDK Methods

Model

Every tokenized model class extends the `Token` class, which in turn extends the `OchainModel` class. The `Token` class is imported from `../lib/token`. Transparent Persistence Capability, or simplified ORM, is captured in the `OchainModel` class.

```
import * as yup from 'yup';
import { Id, Mandatory, Validate, ReadOnly } from '../lib/decorators';
import { Token } from '../lib/token';

@Id('token_id')
export class Digicur extends Token<Digicur> {

    public readonly assetType = 'otoken';

    @Mandatory()
    @Validate(yup.string().required().matches(/^[A-Za-z0-9][A-Za-
z0-9_-]*$/).max(16))
    public token_id: string;

    @ReadOnly('digicur')
    public token_name: string;

    @Validate(yup.string().trim().max(256))
    public token_desc: string;

    @ReadOnly('fungible')
    public token_type: string;


@ReadOnly(["divisible","mintable","transferable","burnable","holdable","roles"
```


Copyright © 2025, Oracle and/or its affiliates.

```
])
    public behaviors: string[];

    @ReadOnly({minter_role_name: "minter", burner_role_name: "burner",
notary_role_name: "notary"})
    public roles: object;

    @ReadOnly({max_mint_quantity: 20000})
    public mintable: object;

    @ReadOnly({decimal: 1})
    public divisible: object;

    @Validate(yup.number())
    public token_to_currency_ratio: number;

    @Validate(yup.string())
    public currency_representation: string;

}
```

Controller

The main controller class extends the `OchainController` class. There is only one main controller.

```
export class DigiCurrCCController extends OchainController{
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable. The other methods are hidden.

You can use the token SDK methods to write custom methods for your business application.

Automatically Generated Token Methods

Blockchain App Builder automatically generates methods to support tokens and token life cycles. You can use these methods to initialize tokens, manage roles and accounts, and complete other token lifecycle tasks without any additional coding. Controller methods must have a `@Validator(...params)` decorator to be invokable.

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Holdable Behavior
  - Burnable Behavior

Methods for Access Control Management

Copyright © 2025, Oracle and/or its affiliates.

- Base
- Digital Assets

Base

addTokenAdmin
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async addTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization('ADMIN.addAdmin', 'TOKEN');
    return await this.Ctx.Admin.addAdmin(org_id, user_id);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfully added Admin (Org_Id: Org1MSP, User_Id: User1)"}
```

removeTokenAdmin
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async removeTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization('ADMIN.removeAdmin', 'TOKEN');
    return await this.Ctx.Admin.removeAdmin(org_id, user_id);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg": "Successfully removed Admin (Org_Id: Org1MSP, User_Id: User1)"}
```

isTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. A `Token Admin` or `Org Admin` can call this function on any other user in the blockchain network. Other users can call this method only on their own accounts.

```
@Validator(yup.string(), yup.string())
  public async isTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.isUserTokenAdmin", "TOKEN");
    return await this.Ctx.Auth.isUserTokenAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

getAllTokenAdmins
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by the `Token Admin` or any `Org Admin` of the chaincode.

```
@Validator()
public async getAllTokenAdmins() {
    await this.Ctx.Auth.checkAuthorization('ADMIN.getAllAdmins', 'TOKEN');
    return await this.Ctx.Admin.getAllAdmins();
}
```

Parameters:

- none

Returns:

- On success, an `admins` array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{"admins":[{"org_id":"Org1MSP","user_id":"admin"}]}
```


Copyright © 2025, Oracle and/or its affiliates.

addOrgAdmin
This method adds a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string())
  public async addOrgAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.addOrgAdmin", "TOKEN",
{ org_id });
    return await this.Ctx.Admin.addOrgAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

removeOrgAdmin
This method removes a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string())
  public async removeOrgAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.removeOrgAdmin", "TOKEN",
{ org_id });
    return await this.Ctx.Admin.removeOrgAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

getOrgAdmins
This method returns a list of all users who are an `Org Admin` of an organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of any organization.

```
  @Validator()
  public async getOrgAdmins() {
    await this.Ctx.Auth.checkAuthorization("ADMIN.getOrgAdmins", "TOKEN");
    return await this.Ctx.Admin.getAllOrgAdmins();
  }
```

Parameters:

- none

Returns:

- On success, an array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin1"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin2"
        }
    ]
}
```

Digital Assets

addTokenAdmin
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async addTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization('ADMIN.addAdmin', 'TOKEN');
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Admin.addAdmin(org_id, user_id);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfully added Admin (Org_Id: Org1MSP, User_Id: User1)"}
```

`removeTokenAdmin`
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async removeTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization('ADMIN.removeAdmin', 'TOKEN');
    return await this.Ctx.Admin.removeAdmin(org_id, user_id);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg": "Successfully removed Admin (Org_Id: Org1MSP, User_Id: User1)"}
```

`isTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. A `Token Admin`, `Token Auditor`, any `Org Admin`, or any `Org Auditor` can call this function on any other user in the blockchain network. Other users can call this method only on their own accounts.

```
@Validator(yup.string(), yup.string())
  public async isTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.isUserTokenAdmin", "TOKEN");
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Auth.isUserTokenAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

`getAllTokenAdmins`
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by the `Token Admin` or `Token Auditor` of the chaincode.

```
@Validator()
public async getAllTokenAdmins() {
    await this.Ctx.Auth.checkAuthorization('ADMIN.getAllAdmins', 'TOKEN');
    return await this.Ctx.Admin.getAllAdmins();
}
```

Parameters:

- none

Returns:

- On success, an `admins` array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{"admins":[{"org_id":"Org1MSP","user_id":"admin"}]}
```

`addOrgAdmin`
This method adds a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string())
  public async addOrgAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.addOrgAdmin", "TOKEN",
{ org_id });
    return await this.Ctx.Admin.addOrgAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`removeOrgAdmin`
This method removes a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string())
  public async removeOrgAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.removeOrgAdmin", "TOKEN",
{ org_id });
    return await this.Ctx.Admin.removeOrgAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
  "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

`getOrgAdmins`
This method returns a list of all users who are an `Org Admin` of an organization. This method can be called only by a `Token Admin`, `Token Auditor`, any `Org Admin`, or any `Org Auditor`.

```
  @Validator()
  public async getOrgAdmins() {
    await this.Ctx.Auth.checkAuthorization("ADMIN.getOrgAdmins", "TOKEN");
    return await this.Ctx.Admin.getAllOrgAdmins();
  }
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, an array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin1"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin2"
        }
    ]
}
```

`addTokenAuditor`
This method adds a user as a `Token Auditor` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
public async addTokenAuditor(org_id: string, user_id: string)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"cd81f6c4c9e7c18ece357dbf5c139ef66ef2d6566be3b14de5f6d0a3fd4bb2f0",
        "payload": {
            "msg": "Successfully added Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20009",
        "blockNumber": 196
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
}
```

removeTokenAuditor
This method removes a user as a `Token Auditor` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
public async removeTokenAuditor(org_id: string, user_id: string)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"a886a6040fbc76374a3c78c89ab0ffc9f7b8391cc5239b169bf3b878cf40c67b",
        "payload": {
            "msg": "Successfully removed Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 219
    }
}
```

getTokenAuditors
This method returns all `Token Auditors` of the chaincode. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode.

```
public async getTokenAuditors()
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "payload": {
            "auditors": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
                {
                    "org_id": "CB",
                    "user_id": "auditor_user_cb"
                }
            ]
        },
        "encode": "JSON"
    }
}
```

addOrgAuditor
This method adds a user as a `Org Auditor` of the chaincode. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
public async addOrgAuditor(org_id: string, user_id: string)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"44bbad35a1478cb714e32f7cfd551897868a203520aab9cea5771d3aadc1cf03",
        "payload": {
            "msg": "Successfully added Org Auditor (Org_Id: CB, User_Id: cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20009",
        "blockNumber": 198
    }
}
```

removeOrgAuditor
This method removes a user as a `Org Auditor` of the chaincode. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
public async removeOrgAuditor(org_id: string, user_id: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"c3bc720461004a53b37c68d4bb264858b88d980bc093a0a3ebb62a32974fb306",
        "payload": {
            "msg": "Successfully removed Org Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 221
    }
}
```

`getOrgAuditors`
This method returns all `Org Auditors` of the chaincode. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
public async getOrgAuditors()
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "payload": {
            "auditors": [
                {
                    "org_id": "FI1",
                    "user_id": "auditor_user_fi1"
                },
                {
                    "org_id": "FI2",
                    "user_id": "auditor_user_fi2"
                }
            ]
        },
        "encode": "JSON"
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
}
```

Methods for Token Configuration Management

- Base
- Digital Assets

Base

`init`
This method is called when the chaincode is deployed or upgraded. Every `Token Admin` is identified by the `user_id` and `org_id` information in the mandatory `adminList` parameter. The `user_id` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `org_id` is the membership service provider (MSP) ID of the user in the current network organization.

Any `Token Admin` user can add and remove other `Token Admin` users by calling the `addAdmin` and `removeAdmin` methods.

```
public async init(adminList: TokenAdminAsset[]) {
    await this.Ctx.Admin.initAdmin(adminList);
    return;
}
```

Parameters:

- `adminList array` – An array of `{user_id, org_id}` information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

Parameter example, macOS and Linux CLI:

```
'[{"user_id":"userid", "org_id":"OrgMSPId"}]'
```

Parameter example, Microsoft Windows CLI:

```
"[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"
```

Parameter example, Oracle Blockchain Platform console:

```
["[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"]
```


Copyright © 2025, Oracle and/or its affiliates.

`initialize<Token Name>Token`
This method creates a token and initializes the token properties. The asset and its properties are saved in the state database. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(Digicur)
    public async initializeDigicurToken(token_asset: Digicur) {
        await this.Ctx.Auth.checkAuthorization('TOKEN.save', 'TOKEN');
        return await this.Ctx.Token.save(token_asset)
    }
```

Parameters:

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

  ```
  "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
  value\",\"Currency_Name\":\"Currency_Name value\"}"
  ```

Returns:

- On success, a JSON representation of the token asset that was created.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 1000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "DOLLAR",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "token_to_currency_ratio": 1
}
```

update<Token Name>Token
This method updates token properties. After a token asset is created, only the `token_desc` property and custom properties can be updated. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(Digicur)
public async updateDigicurToken(token_asset: Digicur) {
    await this.Ctx.Auth.checkAuthorization('TOKEN.update', 'TOKEN');
    return await this.Ctx.Token.update(token_asset);
}
```

Parameters:

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

  ```
  "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
  value\",\"Currency_Name\":\"Currency_Name value\"}"
  ```

Returns:

- On success, an updated JSON representation of the token asset.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_desc": "Digital Currency equiv of dollar",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 1000
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "DOLLAR",
    "token_to_currency_ratio": 1
}
```

getTokenDecimals
This method returns the number of decimal places that were configured for a fractional token. If the `divisible` behavior was not specified for the token, then the default value is 0. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
@Validator(yup.string())
public async getTokenDecimals(token_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Auth.checkAuthorization('TOKEN.getDecimals', 'TOKEN');
    return {
        msg: `Token Id: ${token_id} has $
{this.Ctx.Token.getDecimals(token_asset)} decimal places.`
    };
}
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON string showing the number of token decimal places.

Return Value Example:

```
{"msg":"Token Id: digiCurr101 has 1 decimal places."}
```

getTokenById
This method returns a token object if it is present in the state database. This method can be called only by a `Token Admin` or an `Org Admin` of the chaincode.

```
@Validator(yup.string())
public async getTokenById(token_id: string) {
    await this.Ctx.Auth.checkAuthorization('TOKEN.get', 'TOKEN');
    const token = await this.getTokenObject(token_id);
    return token;
}
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token asset.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_desc": "Digital Currency equiv of dollar",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
        "burner_role_name": "burner",
        "notary_role_name": "notary"
    },
    "mintable": {
        "max_mint_quantity": 2000
    },
    "divisible": {
        "decimal": 1
    },
    "currency_name": "DOLLAR",
    "token_to_currency_ratio": 1
}
```

`getTokenHistory`
This method returns the token history for a specified token ID. Any user can call this method.

```
  @Validator(yup.string())
  public async getTokenHistory(tokenId: string) {
    await this.Ctx.Auth.checkAuthorization("TOKEN.getTokenHistory", "TOKEN");
    return await this.Ctx.Token.history(tokenId);
  }
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token history.

Return Value Example:

```
[
    {
        "trxId":
"0d75f09446a60088afb948c6aca046e261fddcd43df416076201cdc5565f1a35",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "timeStamp": "2023-09-01T16:48:41.000Z",
            "value": {
                "assetType": "otoken",
                "token_id": "token",
                "token_name": "fiatmoneytok",
                "token_desc": "updatedDesc",
                "token_standard": "ttf+",
                "token_type": "fungible",
                "token_unit": "fractional",
                "behaviors": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter"
                },
                "mintable": {
                    "max_mint_quantity": 1000
                },
                "divisible": {
                    "decimal": 2
                }
            }
        },
        {
            "trxId":
    "3666344878b043b65d5b821cc79c042ba52aec467618800df5cf14eac69f72fa",
            "timeStamp": "2023-08-31T20:24:55.000Z",
            "value": {
                "assetType": "otoken",
                "token_id": "token",
                "token_name": "fiatmoneytok",
                "token_standard": "ttf+",
                "token_type": "fungible",
                "token_unit": "fractional",
                "behaviors": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter"
                },
                "mintable": {
                    "max_mint_quantity": 1000
                },
                "divisible": {
                    "decimal": 2
                }
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    }
]
```

`getAllTokens`
This method returns all tokens that are stored in the state database. This method can be called only by a `Token Admin` or an `Org Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@Validator()
public async getAllTokens() {
    await this.Ctx.Auth.checkAuthorization('TOKEN.getAllTokens', 'TOKEN');
    return await this.Ctx.Token.getAllTokens();
}
```

Parameters:

- none

Returns:

- On success, a JSON object that represents all token assets.

`getTokensByName`
This method returns all token objects with a specified name. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@Validator(yup.string())
public async getTokensByName(token_name: string) {
    await this.Ctx.Auth.checkAuthorization('TOKEN.getTokensByName', 'TOKEN');
    return await this.Ctx.Token.getTokensByName(token_name);
}
```

Parameters:

- `token_name: string` – The name of the tokens to retrieve. The name corresponds to the `token_name` property in the specification file. The value is the class name of the token.

Returns:

- On success, a JSON object of all token assets that match the name.

Digital Assets

`init`
This method is called when the chaincode is deployed or upgraded. Every `Token Admin` is identified by the `user_id` and `org_id` information in the mandatory `adminList` parameter. The `user_id` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `org_id` is the membership service provider (MSP) ID of the user in the current network organization.

Copyright © 2025, Oracle and/or its affiliates.

Any `Token Admin` user can add and remove other `Token Admin` users by calling the `addAdmin` and `removeAdmin` methods.

```
public async init(adminList: TokenAdminAsset[]) {
    await this.Ctx.Admin.initAdmin(adminList);
    return;
}
```

Parameters:

- `adminList array` – An array of `{user_id, org_id}` information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

Parameter example, macOS and Linux CLI:

```
'[{"user_id":"userid", "org_id":"OrgMSPId"}]'
```

Parameter example, Microsoft Windows CLI:

```
"[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"
```

Parameter example, Oracle Blockchain Platform console:

```
["[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"]
```

`initialize<Token Name>Token`
This method creates a token and initializes the token properties. The asset and its properties are saved in the state database. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(Digicur)
    public async initializeDigicurToken(token_asset: Digicur) {
        await this.Ctx.Auth.checkAuthorization('TOKEN.save', 'TOKEN');
        return await this.Ctx.Token.save(token_asset)
    }
```

Parameters:

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

  ```
  "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
  value\",\"Currency_Name\":\"Currency_Name value\"}"
  ```

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON representation of the token asset that was created.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 1000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "DOLLAR",
    "token_to_currency_ratio": 1
}
```

update<Token Name>Token
This method updates token properties. After a token asset is created, only the `token_desc` property and custom properties can be updated. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(Digicur)
public async updateDigicurToken(token_asset: Digicur) {
    await this.Ctx.Auth.checkAuthorization('TOKEN.update', 'TOKEN');
    return await this.Ctx.Token.update(token_asset);
}
```

Parameters:

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

Copyright © 2025, Oracle and/or its affiliates.

```
    "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
    value\",\"Currency_Name\":\"Currency_Name value\"}"
```

Returns:

- On success, an updated JSON representation of the token asset.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_desc": "Digital Currency equiv of dollar",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 1000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "DOLLAR",
    "token_to_currency_ratio": 1
}
```

`getTokenDecimals`
This method returns the number of decimal places that were configured for a fractional token. If the `divisible` behavior was not specified for the token, then the default value is 0. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
@Validator(yup.string())
public async getTokenDecimals(token_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Auth.checkAuthorization('TOKEN.getDecimals', 'TOKEN');
    return {
        msg: `Token Id: ${token_id} has $
{this.Ctx.Token.getDecimals(token_asset)} decimal places.`
    };
}
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON string showing the number of token decimal places.

Return Value Example:

```
{"msg":"Token Id: digiCurr101 has 1 decimal places."}
```

`getTokenById`
This method returns a token object if it is present in the state database. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
@Validator(yup.string())
public async getTokenById(token_id: string) {
    await this.Ctx.Auth.checkAuthorization('TOKEN.get', 'TOKEN');
    const token = await this.getTokenObject(token_id);
    return token;
}
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token asset.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_desc": "Digital Currency equiv of dollar",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
        "burner_role_name": "burner",
        "notary_role_name": "notary"
    },
    "mintable": {
        "max_mint_quantity": 2000
    },
    "divisible": {
        "decimal": 1
    },
    "currency_name": "DOLLAR",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "token_to_currency_ratio": 1
}
```

`getTokenHistory`
This method returns the token history for a specified token ID. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
  @Validator(yup.string())
  public async getTokenHistory(tokenId: string) {
    await this.Ctx.Auth.checkAuthorization("TOKEN.getTokenHistory", "TOKEN");
    return await this.Ctx.Token.history(tokenId);
  }
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token history.

Return Value Example:

```
[
    {
        "trxId":
"0d75f09446a60088afb948c6aca046e261fddcd43df416076201cdc5565f1a35",
        "timeStamp": "2023-09-01T16:48:41.000Z",
        "value": {
            "assetType": "otoken",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "token_desc": "updatedDesc",
            "token_standard": "ttf+",
            "token_type": "fungible",
            "token_unit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
            "mintable": {
                "max_mint_quantity": 1000
            },
            "divisible": {
                "decimal": 2
            }
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "trxId":
"3666344878b043b65d5b821cc79c042ba52aec467618800df5cf14eac69f72fa",
        "timeStamp": "2023-08-31T20:24:55.000Z",
        "value": {
            "assetType": "otoken",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "token_standard": "ttf+",
            "token_type": "fungible",
            "token_unit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
            "mintable": {
                "max_mint_quantity": 1000
            },
            "divisible": {
                "decimal": 2
            }
        }
    }
]
```

`getAllTokens`
This method returns all tokens that are stored in the state database. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@Validator()
public async getAllTokens() {
    await this.Ctx.Auth.checkAuthorization('TOKEN.getAllTokens', 'TOKEN');
    return await this.Ctx.Token.getAllTokens();
}
```

Parameters:

- none

Returns:

- On success, a JSON object that represents all token assets.

`getTokensByName`
This method returns all token objects with a specified name. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`. This method uses Berkeley

Copyright © 2025, Oracle and/or its affiliates.

DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@Validator(yup.string())
public async getTokensByName(token_name: string) {
    await this.Ctx.Auth.checkAuthorization('TOKEN.getTokensByName', 'TOKEN');
    return await this.Ctx.Token.getTokensByName(token_name);
}
```

Parameters:

- `token_name: string` – The name of the tokens to retrieve. The name corresponds to the `token_name` property in the specification file. The value is the class name of the token.

Returns:

- On success, a JSON object of all token assets that match the name.

Methods for Account Management

- Base
- Digital Assets

Base

`createAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track balances, on-hold balances, and transaction history. An account ID is an alphanumeric set of characters, prefixed with `oaccount~<`*`token asset name`*`>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async createAccount(org_id: string, user_id: string, token_type:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.createAccount", "TOKEN",
{ org_id });
    return await this.Ctx.Account.createAccount(org_id, user_id, token_type);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

- `token_type: string` – The type of the token, which must be `fungible`.

Returns:

- On success, a JSON object of the account that was created. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
  "assetType": "oaccount",
  "account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
  "bapAccountVersion": 0,
  "user_id": "admin",
  "org_id": "Org1MSP",
  "token_type": "fungible",
  "token_id": "",
  "token_name": "",
  "balance": 0,
  "onhold_balance": 0
}
```

`associateTokenToAccount`
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the relevant organization.

```
  @Validator(yup.string(), yup.string())
  public async associateTokenToAccount(account_id: string, token_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.associateToken", "TOKEN",
{ account_id });
    return await this.Ctx.Account.associateToken(account_id, token_id);
  }
```

Parameters:

- `account_id: string` – The ID of the account.
- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON object of the updated account. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
    "assetType": "oaccount",
    "account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
    "bapAccountVersion": 0,
    "user_id": "admin",
    "org_id": "Org1MSP",
    "token_type": "fungible",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "token_id": "fungible",
    "token_name": "fiatmoneytok",
    "balance": 0,
    "onhold_balance": 0
}
```

`getAccount`
This method returns account details for a specified user and token. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
@Validator(yup.string(), yup.string(), yup.string())
public async getAccount(token_id: string, org_id: string, user_id: string) {
  const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
  await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccount", "TOKEN",
{ account_id });
  return await this.Ctx.Account.getAccountWithStatus(account_id);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `account_id` – The ID of the user account.
- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current organization.
- `token_id` – The ID of the token.
- `token_name` – The name of the token.
- `balance` – The current balance of the account.
- `onhold_balance` – The current on-hold balance of the account.
- `bapAccountVersion` – An account object parameter for internal use.
- `status` – The current status of the user account.

Return Value Example:

```
{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "account_id":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "user_id": "idcqa",
  "org_id": "appdev",
  "token_type": "fungible",
  "token_id": "t1",
  "token_name": "obptok",
  "balance": 0,
  "onhold_balance": 0
}
```

`getAccountHistory`
This method returns account history details for a specified user and token. This method can be called only by a `Token Admin` of the chaincode or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountHistory(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.history", "TOKEN",
{ account_id });
    return await this.Ctx.Account.history(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, an array of JSON account objects that includes the following properties:
- `trxId` – The transaction ID of the transaction as returned by the ledger.
- `timeStamp` – The time of the transaction.
- `value` – A JSON string of the account object.

Return Value Example:

```
[
    {

"trxId":"2gsdh17fff222467e5667be042e33ce18e804b3e065cca15de306f837e416d7c3e",
      "timeStamp":1629718288,
      "value":{
         "assetType":"oaccount",

"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
         "user_id":"user1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "org_id":"Org1MSP",
            "token_id":"digiCurr101",
            "token_name":"digicur",
            "balance":100,
            "onhold_balance":0,
            "bapAccountVersion": 1
      },
      {

"trxId":"9fd07fff222467e5667be042e33ce18e804b3e065cca15de306f837e416d7c3e",
         "timeStamp":1629718288,
         "value":{
            "assetType":"oaccount",

"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
            "user_id":"user1",
            "org_id":"Org1MSP",
            "token_id":"digiCurr101",
            "token_name":"digicur",
            "balance":0,
            "onhold_balance":0,
            "bapAccountVersion": 0
         }
      }
]
```

`getAccountOnHoldBalance`
This method returns the current on-hold balance for a specified account and token. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountOnHoldBalance(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountOnHoldBalance",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountOnHoldBalance(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current on-hold balance.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"Total Holding Balance is: 0","holding_balance":0}
```

`getAllAccounts`
This method returns a list of all accounts. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@Validator()
public async getAllAccounts() {
    await this.Ctx.Auth.checkAuthorization('ACCOUNT.getAllAccounts', 'TOKEN');
    return await this.Ctx.Account.getAllAccounts();
}
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

`getUserByAccountId`
This method returns user details (`org_id` and `user_id`) for a specified account. This method can be called by any user of the chaincode.

```
@Validator(yup.string())
public async getUserByAccountId(account_id: string) {
    return await this.Ctx.Account.getUserByAccountId(account_id);
}
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`org_id`, `token_id`, and `user_id`).

Return Value Example:

```
{
    "token_id": "digiCurr101",
    "user_id": "user1",
    "org_id": "Org1MSP"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getAccountBalance`
This method returns the current balance for a specified account and token. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountBalance(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountBalance",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountBalance(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current account balance.

Return Value Example:

```
{"msg":"Current Balance is: 0","user_balance":0}
```

`getAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
  @Validator(yup.string())
  public async getAllOrgAccounts(org_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAllOrgAccounts",
"TOKEN", { org_id });
    return await this.Ctx.Account.getAllOrgAccounts(org_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "user_id": "idcqa",
            "org_id": "appdev",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "balance": 0,
            "onhold_balance": 0
        }
    },
    {
        "key":
"oaccount~620fcf5deb5fd5a65c0b5b10fda129de0f629ccd232c5891c130e24a574df50a",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~620fcf5deb5fd5a65c0b5b10fda129de0f629ccd232c5891c130e24a574df50a",
            "user_id": "example_minter",
            "org_id": "appdev",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "balance": 0,
            "onhold_balance": 0
        }
    }
]
```

Digital Assets

`createAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track balances, on-hold balances, and transaction history. An account ID is an alphanumeric set of characters, prefixed with `oaccount~<`*`token asset name`*`>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.object().nullable())
```


Copyright © 2025, Oracle and/or its affiliates.

```
public async createAccount(org_id: string, user_id: string, token_type:
string, daily_limits: DailyLimits) {
await this.Ctx.Auth.checkAuthorization("ACCOUNT.createAccount", "TOKEN",
{ org_id });
return await this.Ctx.Account.createAccount(org_id, user_id, token_type,
daily_limits);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `token_type: string` – The type of the token, which must be `fungible`.
- `daily_limits: JSON` – An object specifying the maximum amount of tokens that can be used in transactions daily (`max_daily_amount`) and the maximum number of transactions that can be completed daily (`max_daily_transactions`).

  You specify the `daily_limits` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "max_daily_amount": 1000, "max_daily_transactions": 100 }`

  CLI / Postman: `"{\"max_daily_amount\":1000,\"max_daily_transactions\":100}"`

Returns:

- On success, a JSON object of the account that was created. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
  "assetType": "oaccount",
  "account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
  "bapAccountVersion": 0,
  "user_id": "admin",
  "org_id": "Org1MSP",
  "token_type": "fungible",
  "token_id": "",
  "token_name": "",
  "balance": 0,
  "onhold_balance": 0,
  "onhold_burn_balance": 0,
  "max_daily_amount": 1000,
  "max_daily_transactions": 100
}
```


Copyright © 2025, Oracle and/or its affiliates.

associateTokenToAccount
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the relevant organization.

```
  @Validator(yup.string(), yup.string())
  public async associateTokenToAccount(account_id: string, token_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.associateToken", "TOKEN",
{ account_id });
    return await this.Ctx.Account.associateToken(account_id, token_id);
  }
```

Parameters:

- `account_id: string` – The ID of the account.
- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON object of the updated account. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
    "assetType": "oaccount",
    "account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
    "bapAccountVersion": 0,
    "user_id": "admin",
    "org_id": "Org1MSP",
    "token_type": "fungible",
    "token_id": "fungible",
    "token_name": "fiatmoneytok",
    "balance": 0,
    "onhold_balance": 0
}
```

getAccount
This method returns account details for a specified user and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@Validator(yup.string(), yup.string(), yup.string())
public async getAccount(token_id: string, org_id: string, user_id: string) {
  const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
  await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccount", "TOKEN",
{ account_id });
  return await this.Ctx.Account.getAccountWithStatus(account_id);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `account_id` – The ID of the user account.
- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current organization.
- `token_id` – The ID of the token.
- `token_name` – The name of the token.
- `balance` – The current balance of the account.
- `onhold_balance` – The current on-hold balance of the account.
- `bapAccountVersion` – An account object parameter for internal use.
- `status` – The current status of the user account.

Return Value Example:

```
{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "account_id":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "user_id": "idcqa",
  "org_id": "appdev",
  "token_type": "fungible",
  "token_id": "t1",
  "token_name": "obptok",
  "balance": 0,
  "onhold_balance": 0
}
```

`getAccountHistory`
This method returns account history details for a specified user and token. This method can be called only by a `Token Admin` of the chaincode or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountHistory(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.history", "TOKEN",
{ account_id });
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Account.history(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, an array of JSON account objects that includes the following properties:
- `trxId` – The transaction ID of the transaction as returned by the ledger.
- `timeStamp` – The time of the transaction.
- `value` – A JSON string of the account object.

Return Value Example:

```
[
    {

"trxId":"2gsdh17fff222467e5667be042e33ce18e804b3e065cca15de306f837e416d7c3e",
      "timeStamp":1629718288,
      "value":{
         "assetType":"oaccount",

"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
         "user_id":"user1",
         "org_id":"Org1MSP",
         "token_id":"digiCurr101",
         "token_name":"digicur",
         "balance":100,
         "onhold_balance":0,
         "bapAccountVersion": 1
   },
   {

"trxId":"9fd07fff222467e5667be042e33ce18e804b3e065cca15de306f837e416d7c3e",
      "timeStamp":1629718288,
      "value":{
         "assetType":"oaccount",

"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
         "user_id":"user1",
         "org_id":"Org1MSP",
         "token_id":"digiCurr101",
         "token_name":"digicur",
         "balance":0,
         "onhold_balance":0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "bapAccountVersion": 0
      }
    }
]
```

getAccountOnHoldBalance
This method returns the current on-hold balance for a specified account and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountOnHoldBalance(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountOnHoldBalance",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountOnHoldBalance(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current on-hold balance.

Return Value Example:

```
{"msg":"Total Holding Balance is: 0","holding_balance":0}
```

getAllAccounts
This method returns a list of all accounts. This method can be called only by a `Token Admin` or `Token Auditor`. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@Validator()
public async getAllAccounts() {
    await this.Ctx.Auth.checkAuthorization('ACCOUNT.getAllAccounts', 'TOKEN');
    return await this.Ctx.Account.getAllAccounts();
}
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON array of all accounts.

`getUserByAccountId`
This method returns user details (`org_id` and `user_id`) for a specified account. This method can be called by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
@Validator(yup.string())
public async getUserByAccountId(account_id: string) {
    return await this.Ctx.Account.getUserByAccountId(account_id);
}
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`org_id`, `token_id`, and `user_id`).

Return Value Example:

```
{
    "token_id": "digiCurr101",
    "user_id": "user1",
    "org_id": "Org1MSP"
}
```

`getAccountBalance`
This method returns the current balance for a specified account and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountBalance(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountBalance",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountBalance(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current account balance.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"Current Balance is: 0","user_balance":0}
```

`getAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
  @Validator(yup.string())
  public async getAllOrgAccounts(org_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAllOrgAccounts",
"TOKEN", { org_id });
    return await this.Ctx.Account.getAllOrgAccounts(org_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "user_id": "idcqa",
            "org_id": "appdev",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "balance": 0,
            "onhold_balance": 0
        }
    },
    {
        "key":
"oaccount~620fcf5deb5fd5a65c0b5b10fda129de0f629ccd232c5891c130e24a574df50a",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~620fcf5deb5fd5a65c0b5b10fda129de0f629ccd232c5891c130e24a574df50a",
            "user_id": "example_minter",
            "org_id": "appdev",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "balance": 0,
            "onhold_balance": 0
        }
    }
]
```

Methods for Role Management

- Base
- Digital Assets

Base

`addRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also holds the specified role.

```
  @Validator(yup.string(), yup.string(), yup.string(), yup.string())
  public async addRole(token_id: string, role: string, org_id: string,
user_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.addRoleMember", "TOKEN",
{ token_id, org_id, role });
    return await this.Ctx.Token.addRoleMember(role, account_id, token_asset);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"Successfully added role 'minter' to Account Id:
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)"}
```

`removeRole`
This method removes a role from a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also holds the specified role.

```
  @Validator(yup.string(), yup.string(), yup.string(), yup.string())
  public async removeRole(token_id: string, role: string, org_id: string,
user_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.removeRoleMember", "TOKEN",
{ token_id, org_id, role });
    return await this.Ctx.Token.removeRoleMember(role, account_id,
token_asset);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
{"msg":"Successfully removed role 'minter' from Account Id:
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)"}
```

`getAccountsByRole`
This method returns a list of all account IDs for a specified role and token. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async getAccountsByRole(token_id: string, role: string) {
   await this.Ctx.Auth.checkAuthorization('ROLE.getAccountsByRole', 'TOKEN');
```


Copyright © 2025, Oracle and/or its affiliates.

```
   return await this.Ctx.Role.getAccountsByRole(token_id, role);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{"accounts":
["oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642
f622f"]}
```

`getAccountsByUser`
This method returns a list of all account IDs for a specified organization ID and user ID. This method can be called only by the `Token Admin` of the chaincode, by the `Org Admin` of the specified organization, or by the `Account Owner` specified in the parameters.

```
  @Validator(yup.string(), yup.string())
  public async getAccountsByUser(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountsByUser",
"TOKEN", { org_id, user_id });
    return await this.Ctx.Account.getAccountsByUser(org_id, user_id);
  }
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{"accounts":
["oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642
f622f"]}
```

`getUsersByRole`
This method returns a list of all users for a specified role and token. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async getUsersByRole(token_id: string, role: string) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    await this.Ctx.Auth.checkAuthorization('ROLE.getUsersByRole', 'TOKEN');
    return await this.Ctx.Role.getUsersByRole(token_id, role);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of the user objects (`org_id`, `token_id`, and `user_id`).

Return Value Example:

```
{"users":[{"token_id":"digiCurr101","user_id":"user1","org_id":"Org1MSP"}]}
```

`isInRole`
This method returns a Boolean value to indicate if a user and token has a specified role. This method can be called only by a `Token Admin` of the chaincode, the `AccountOwner` of the account, or an `Org Admin` of the specified organization.

```
  @Validator(yup.string(), yup.string(), yup.string(), yup.string())
  public async isInRole(token_id: string, org_id: string, user_id: string,
role: string) {
    const token_asset = await this.getTokenObject(token_id);
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.isInRole", "TOKEN",
{ account_id });
    return { result: await this.Ctx.Token.isInRole(role, account_id,
token_asset) };
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Return Value Example:

```
{"result":"false"}
```


Copyright © 2025, Oracle and/or its affiliates.

`getOrgAccountsByRole`
This method returns information about all accounts that have a specified role in a specified organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
   @Validator(yup.string(), yup.string(), yup.string())
  public async getOrgAccountsByRole(token_id: string, role: string, org_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getOrgAccountsByRole",
"TOKEN", { org_id });
    return await this.Ctx.Role.getOrgAccountsByRole(token_id, role, org_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts with the specified role in the specified organization.

Return Value Example:

```
{
    "accounts": [

"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",

"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850"
    ]
}
```

`getOrgUsersByRole`
This method returns information about all users that have a specified role in a specified organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getOrgUsersByRole(token_id: string, role: string, org_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getOrgUsersByRole", "TOKEN",
{ org_id });
    return await this.Ctx.Role.getOrgUsersByRole(token_id, role, org_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.

Copyright © 2025, Oracle and/or its affiliates.

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all users with the specified role in the specified organization.

Return Value Example:

```
{
    "users": [
        {
            "token_id": "token",
            "user_id": "admin",
            "org_id": "Org1MSP"
        },
        {
            "token_id": "token",
            "user_id": "orgAdmin",
            "org_id": "Org1MSP"
        }
    ]
}
```

Digital Assets

`addRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also holds the specified role.

```
  @Validator(yup.string(), yup.string(), yup.string(), yup.string())
  public async addRole(token_id: string, role: string, org_id: string,
user_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.addRoleMember", "TOKEN",
{ token_id, org_id, role });
    return await this.Ctx.Token.addRoleMember(role, account_id, token_asset);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message with account details.

Return Value Example:

```
{"msg":"Successfully added role 'minter' to Account Id:
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)"}
```

`removeRole`
This method removes a role from a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also holds the specified role.

```
  @Validator(yup.string(), yup.string(), yup.string(), yup.string())
  public async removeRole(token_id: string, role: string, org_id: string,
user_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.removeRoleMember", "TOKEN",
{ token_id, org_id, role });
    return await this.Ctx.Token.removeRoleMember(role, account_id,
token_asset);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
{"msg":"Successfully removed role 'minter' from Account Id:
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)"}
```


Copyright © 2025, Oracle and/or its affiliates.

getAccountsByRole
This method returns a list of all account IDs for a specified role and token. This method can be called only by a `Token Admin` or `Token Auditor`.

```
@Validator(yup.string(), yup.string())
public async getAccountsByRole(token_id: string, role: string) {
   await this.Ctx.Auth.checkAuthorization('ROLE.getAccountsByRole', 'TOKEN');
   return await this.Ctx.Role.getAccountsByRole(token_id, role);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{"accounts":
["oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642
f622f"]}
```

getAccountsByUser
This method returns a list of all account IDs for a specified organization ID and user ID. This method can be called only by the `Token Admin` or `Token Auditor`, by the `Org Admin` or `Org Auditor` of the specified organization, or by the `Account Owner` specified in the parameters.

```
  @Validator(yup.string(), yup.string())
  public async getAccountsByUser(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountsByUser",
"TOKEN", { org_id, user_id });
    return await this.Ctx.Account.getAccountsByUser(org_id, user_id);
  }
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{"accounts":
["oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642
f622f"]}
```


Copyright © 2025, Oracle and/or its affiliates.

`getUsersByRole`
This method returns a list of all users for a specified role and token. This method can be called only by a `Token Admin` or `Token Auditor`.

```
@Validator(yup.string(), yup.string())
public async getUsersByRole(token_id: string, role: string) {
    await this.Ctx.Auth.checkAuthorization('ROLE.getUsersByRole', 'TOKEN');
    return await this.Ctx.Role.getUsersByRole(token_id, role);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of the user objects (`org_id`, `token_id`, and `user_id`).

Return Value Example:

```
{"users":[{"token_id":"digiCurr101","user_id":"user1","org_id":"Org1MSP"}]}
```

`isInRole`
This method returns a Boolean value to indicate if a user and token has a specified role. This method can be called only by a `Token Admin` or `Token Auditor`, the `AccountOwner` of the account, or an `Org Admin` or `Org Auditor` of the specified organization.

```
  @Validator(yup.string(), yup.string(), yup.string(), yup.string())
  public async isInRole(token_id: string, org_id: string, user_id: string,
role: string) {
    const token_asset = await this.getTokenObject(token_id);
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.isInRole", "TOKEN",
{ account_id });
    return { result: await this.Ctx.Token.isInRole(role, account_id,
token_asset) };
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"result":"false"}
```

`getOrgAccountsByRole`
This method returns information about all accounts that have a specified role in a specified organization. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin` or `Org Auditor`.

```
   @Validator(yup.string(), yup.string(), yup.string())
  public async getOrgAccountsByRole(token_id: string, role: string, org_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getOrgAccountsByRole",
"TOKEN", { org_id });
    return await this.Ctx.Role.getOrgAccountsByRole(token_id, role, org_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts with the specified role in the specified organization.

Return Value Example:

```
{
    "accounts": [

"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",

"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850"
    ]
}
```

`getOrgUsersByRole`
This method returns information about all users that have a specified role in a specified organization. This method can be called only by a `Token Admin` or `Token Auditor`, by an `Org Admin` or `Org Auditor` of the specified organization.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getOrgUsersByRole(token_id: string, role: string, org_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getOrgUsersByRole", "TOKEN",
{ org_id });
    return await this.Ctx.Role.getOrgUsersByRole(token_id, role, org_id);
  }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all users with the specified role in the specified organization.

Return Value Example:

```
{
    "users": [
        {
            "token_id": "token",
            "user_id": "admin",
            "org_id": "Org1MSP"
        },
        {
            "token_id": "token",
            "user_id": "orgAdmin",
            "org_id": "Org1MSP"
        }
    ]
}
```

Methods for Transaction History Management

- Base
- Digital Assets

Base

`getAccountTransactionHistory`
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
 @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountTransactionHistory(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistory",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountTransactionHistory(account_id,
```


Copyright © 2025, Oracle and/or its affiliates.

```
org_id, user_id.toLowerCase());
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, an array of JSON account transaction objects that includes the following properties:
- `transaction_id` – The ID of the transaction.
- `transacted_account` – The account with which the transaction took place.
- `transaction_type` – The type of transaction.
- `transacted_amount` – The amount of the transaction.
- `timestamp` – The time of the transaction.
- `balance` – The account balance at the time of the transaction.
- `onhold_balance` – The on-hold balance at the time of the transaction.
- `token_id` – The ID of the token.
- `holding_id` – A unique identifier returned by the `holdTokens` method.

Return Value Example:

```
[
    {
        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
        "transacted_amount": 20,
        "timestamp": "2021-08-17T06:04:24.000Z",
        "balance": 930,
        "onhold_balance": 0,
        "token_id": "digiCurr101",
        "transaction_type": "BULKTRANSFER",
        "sub_transactions": [
            {
                "transacted_account":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "transaction_type": "DEBIT",
                "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
                "transacted_amount": 10
            },
            {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "transacted_account":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
                "transaction_type": "DEBIT",
                "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c81e728d9d4c2f636f067f89cc14862c",
                "transacted_amount": 10
            }
        ]
    },
    {
        "transaction_id":
"otransaction~757864d5369bd0539d044caeb3bb4898db310fd7aa740f45a9938771903d43da
",
        "transacted_amount": 50,
        "timestamp": "2021-08-17T06:02:44.000Z",
        "balance": 950,
        "onhold_balance": 0,
        "token_id": "digiCurr101",
        "transacted_account":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
        "transaction_type": "DEBIT"
    }
]
```

`getAccountTransactionHistoryWithFilters`
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
  @Validator(yup.string(), yup.string(), yup.string(),
yup.object().nullable())
  public async getAccountTransactionHistoryWithFilters(token_id: string,
org_id: string, user_id: string, filters?: Filters) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistoryWithFilt
ers", "TOKEN", { account_id });
    return await
this.Ctx.Account.getAccountTransactionHistoryWithFilters(account_id, org_id,
user_id.toLowerCase(), filters);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `user_id: string` – The user name or email ID of the user.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Example:

```
ochain invoke GetAccountTransactionHistoryWithFilters 'token1' 'appbuilder12'
'user_minter'
'{"PageSize":10,"Bookmark":"1","StartTime":"2022-01-25T17:41:42Z","EndTime":"202
2-01-25T17:59:10Z"}'
```


```
[
  {
    "transaction_id":
"otransaction~672897b5a4fa78b421c000e4d6d4f71f3d46529bfbb5b4be10bf5471dc35ce89
",
    "transacted_amount": 5,
    "timestamp": "2022-04-20T15:46:04.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
    "transaction_type": "DEBIT",
    "balance": 90,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~467bb67a33aaffca4487f33dcd46c9844efdb5421a2e7b6aa2d53152eb2c6d85
",
    "transacted_amount": 5,
    "timestamp": "2022-04-20T15:45:47.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
    "transaction_type": "DEBIT",
    "balance": 95,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~c6d56ce54a9bbe24597d1d10448e39316dc6f16328bf3c5b0c8ef10e1dfeb397
",
    "transacted_amount": 100,
    "timestamp": "2022-04-20T15:44:26.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~deb5fb0906c40506f6c2d00c573b774e01a53dd91499e651d92ac4778b6add6a",
    "transaction_type": "MINT",
    "balance": 100,
    "onhold_balance": 0
```


Copyright © 2025, Oracle and/or its affiliates.

```
  }
]
```

getSubTransactionsById
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` of the chaincode or the `AccountOwner` of the account.

```
  @Validator(yup.string())
  public async getSubTransactionsById(transaction_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getSubTransactionsById",
"TOKEN", { transaction_id });
    return await this.Ctx.Account.getSubTransactionsById(transaction_id);
  }
```

Parameters:

- `transaction_id: string` – The ID of the bulk transfer transaction.

Returns:

- An array of account subtransaction objects in JSON format for a specified bulk transfer transaction ID.

Example:

```
ochain invoke GetAccountSubTransactionHistory
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'

[
    {
        "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
        "transaction_type": "DEBIT",
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c81e728d9d4c2f636f067f89cc14862c",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "balance": 80,
        "onhold_balance": 0
    },
    {
        "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
        "transaction_type": "DEBIT",
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c4ca4238a0b923820dcc509a6f75849b",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "balance": 85,
        "onhold_balance": 0
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
]
```

`getSubTransactionsByIdWithFilters`
This method returns an array of account subtransaction history details for a specified transaction.

```
  @Validator(yup.string(), yup.object().nullable())
  public async getSubTransactionsByIdWithFilters(transaction_id: string,
filters?: SubTransactionFilters) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getSubTransactionsByIdWithFilters",
"TOKEN", { transaction_id });
    return await
this.Ctx.Account.getSubTransactionsByIdWithFilters(transaction_id, filters);
  }
```

Parameters:

- `transaction_id: string` – The ID of the transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Returns:

- An array of account subtransaction objects in JSON format for a specified bulk transfer transaction ID.

Example:

```
ochain invoke GetAccountSubTransactionHistoryWithFilters
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
'{"PageSize":10,"Bookmark":"1"}'
```

```
[
  {
    "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c81e728d9d4c2f636f067f89cc14862c",
    "transacted_amount": 5,
    "timestamp": "2022-04-20T15:52:21.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
    "transaction_type": "DEBIT",
    "balance": 80,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
```


Copyright © 2025, Oracle and/or its affiliates.

```
~c4ca4238a0b923820dcc509a6f75849b",
    "transacted_amount": 5,
    "timestamp": "2022-04-20T15:52:21.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
    "transaction_type": "DEBIT",
    "balance": 85,
    "onhold_balance": 0
  }
]
```

`getTransactionById`
This method returns the history of a `Transaction` asset.

```
@Validator(yup.string())
    public async getTransactionById(transaction_id: string) {
        return await this.Ctx.Transaction.getTransactionById(transaction_id);
    }
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Returns:

- On success, an JSON array of the history for the transaction.

Return Value Example:

```
{
    "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
    "history": [
        {
            "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
            "timeStamp": 1629180264,
            "value": {
                "assetType": "otransaction",
                "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
                "token_id": "digiCurr101",
                "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                "to_account_id": "",
                "transaction_type": "BULKTRANSFER",
                "amount": 20,
                "timestamp": "2021-08-17T06:04:24.000Z",
                "number_of_sub_transactions": 2,
                "holding_id": ""
```


Copyright © 2025, Oracle and/or its affiliates.

```
            }
        }
    ],
    "sub_transactions": [
        {
            "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
            "history": [
                {
                    "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
                    "timeStamp": 1629180264,
                    "value": {
                        "assetType": "otransaction",
                        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
                        "token_id": "digiCurr101",
                        "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                        "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                        "transaction_type": "TRANSFER",
                        "amount": 10,
                        "timestamp": "2021-08-17T06:04:24.000Z",
                        "number_of_sub_transactions": 0,
                        "holding_id": ""
                    }
                }
            ]
        },
        {
            "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c81e728d9d4c2f636f067f89cc14862c",
            "history": [
                {
                    "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
                    "timeStamp": 1629180264,
                    "value": {
                        "assetType": "otransaction",
                        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c81e728d9d4c2f636f067f89cc14862c",
                        "token_id": "digiCurr101",
                        "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                        "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
```


Copyright © 2025, Oracle and/or its affiliates.

```
706e",
                        "transaction_type": "TRANSFER",
                        "amount": 10,
                        "timestamp": "2021-08-17T06:04:24.000Z",
                        "number_of_sub_transactions": 0,
                        "holding_id": ""
                    }
                }
            ]
        }
    ]
}
```

deleteHistoricalTransactions
This method deletes older transactions from the state database.

```
@Validator(yup.date())
    public async deleteHistoricalTransactions(time_to_expiration: Date) {
        await
this.Ctx.Auth.checkAuthorization('TRANSACTION.deleteTransactions', 'TOKEN');
        return await
this.Ctx.Transaction.deleteTransactions(time_to_expiration);
    }
```

Parameters:

- `time_to_expiration Date` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted.

Return Value Example:

```
"payload": {
    "msg": "Successfuly deleted transaction older than date: Thu Aug 19 2021
11:19:36 GMT+0000 (Coordinated Universal Time).",
    "transactions": [

"otransaction~ec3366dd48b4ce2838f820f2f138648e6e55a07226713e59b411ff31b0d21058
"
    ]
}
```

Digital Assets

getAccountTransactionHistoryWithFiltersFromRichHistDB
You can synchronize data to the rich history database and then fetch the data using chaincode API calls. This method fetches transaction history from the rich history database. Before you can use this method, you must run Oracle Autonomous Database with Oracle REST Data Services (ORDS) and OAuth enabled, as described in Oracle Database View Definitions for Wholesale CBDC in *Oracle Blockchain Platform Digital Assets Edition*.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
```


Copyright © 2025, Oracle and/or its affiliates.

```
yup.string(), yup.object().nullable())
public async getAccountTransactionHistoryWithFiltersFromRichHistDB(token_id:
string, org_id: string, user_id: string, custom_endpoint: string,
bearer_token: string, filters?: Filters) {
const account_id = await this.Ctx.Account.generateAccountId(token_id, org_id,
user_id);
await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistoryWithFilt
ers", "TOKEN", { account_id });
return await
this.Ctx.Account.getAccountTrxHistoryWithFiltersFromRichHistDB(account_id,
org_id, user_id.toLowerCase(), custom_endpoint, bearer_token, filters);
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

`getAccountTransactionHistory`
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async getAccountTransactionHistory(token_id: string, org_id: string,
user_id: string) {
const account_id = await this.Ctx.Account.generateAccountId(token_id, org_id,
user_id);
await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistory",
"TOKEN", { account_id });
return await this.Ctx.Account.getAccountTransactionHistory(account_id,
org_id, user_id.toLowerCase());
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
[
            {
                "transaction_id":
"otransaction~64c5a4830949eae1424600f3d4a438c6f603a7c3ea31a68e374b899803999e22
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:37:28.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REJECT_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~a4537ef34a955b023b7c205b9abf06a6c79e4fdd761fb24f41b8eb34126b66c0
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:32.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "APPROVE_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~6237a759422bd9fb112742e8cd7e6450df5a74a32236d9b1005571afed8904a4
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:18.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
```


Copyright © 2025, Oracle and/or its affiliates.

```
            },
            {
                "transaction_id":
"otransaction~06b35071415d74aa1a7c18449149c937d886cae76a832c44cf8d98e84586e76e
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:35:46.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            }
 ]
```

`getAccountTransactionHistoryWithFilters`
This method returns a filtered array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string(), yup.object().nullable())
public async getAccountTransactionHistoryWithFilters(token_id: string,
org_id: string, user_id: string, filters?: Filters) {
const account_id = await this.Ctx.Account.generateAccountId(token_id, org_id,
user_id);
await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistoryWithFilt
ers", "TOKEN", { account_id });
return await
this.Ctx.Account.getAccountTransactionHistoryWithFilters(account_id, org_id,
user_id.toLowerCase(), filters);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
[
            {
                "transaction_id":
"otransaction~64c5a4830949eae1424600f3d4a438c6f603a7c3ea31a68e374b899803999e22
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:37:28.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REJECT_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~a4537ef34a955b023b7c205b9abf06a6c79e4fdd761fb24f41b8eb34126b66c0
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:32.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "APPROVE_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~6237a759422bd9fb112742e8cd7e6450df5a74a32236d9b1005571afed8904a4
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:18.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "transaction_id":
"otransaction~06b35071415d74aa1a7c18449149c937d886cae76a832c44cf8d98e84586e76e
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:35:46.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            }
 ]
```

`getSubTransactionsById`
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin`, `Token Auditor`, or the `AccountOwner` who called the transaction.

```
  @Validator(yup.string())
  public async getSubTransactionsById(transaction_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getSubTransactionsById",
"TOKEN", { transaction_id });
    return await this.Ctx.Account.getSubTransactionsById(transaction_id);
  }
```

Parameters:

- `transaction_id: string` – The ID of the bulk transfer transaction.

Returns:

- An array of account subtransaction objects in JSON format for a specified bulk transfer transaction ID.

Example:

```
ochain invoke GetAccountSubTransactionHistory
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'

[
    {
        "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
        "transaction_type": "DEBIT",
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c81e728d9d4c2f636f067f89cc14862c",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "balance": 80,
        "onhold_balance": 0
    },
    {
        "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
        "transaction_type": "DEBIT",
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c4ca4238a0b923820dcc509a6f75849b",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "balance": 85,
        "onhold_balance": 0
    }
]
```

`getSubTransactionsByIdWithFilters`
This method returns an array of account subtransaction history details for a specified transaction.

```
  @Validator(yup.string(), yup.object().nullable())
  public async getSubTransactionsByIdWithFilters(transaction_id: string,
filters?: SubTransactionFilters) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getSubTransactionsByIdWithFilters",
"TOKEN", { transaction_id });
    return await
this.Ctx.Account.getSubTransactionsByIdWithFilters(transaction_id, filters);
  }
```

Parameters:

- `transaction_id: string` – The ID of the transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Returns:

- An array of account subtransaction objects in JSON format for a specified bulk transfer transaction ID.

Example:

```
ochain invoke GetAccountSubTransactionHistoryWithFilters
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
'{"PageSize":10,"Bookmark":"1"}'
```

```
[
  {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c81e728d9d4c2f636f067f89cc14862c",
    "transacted_amount": 5,
    "timestamp": "2022-04-20T15:52:21.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
    "transaction_type": "DEBIT",
    "balance": 80,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c4ca4238a0b923820dcc509a6f75849b",
    "transacted_amount": 5,
    "timestamp": "2022-04-20T15:52:21.000Z",
    "token_id": "tokenId",
    "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
    "transaction_type": "DEBIT",
    "balance": 85,
    "onhold_balance": 0
  }
]
```

`getTransactionById`
This method returns the history of a `Transaction` asset. This method can be called only by a `Token Admin` or `Token Auditor`, by an `Org Admin` or `Org Auditor` of the specified organization, or by a transaction participant (sender, recipient, or notary).

```
@Validator(yup.string())
    public async getTransactionById(transaction_id: string) {
        return await this.Ctx.Transaction.getTransactionById(transaction_id);
    }
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Returns:

- On success, an JSON array of the history for the transaction.

Return Value Example:

```
{
    "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
    "history": [
        {
            "trxId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
            "timeStamp": 1629180264,
            "value": {
                "assetType": "otransaction",
                "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
                "token_id": "digiCurr101",
                "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                "to_account_id": "",
                "transaction_type": "BULKTRANSFER",
                "amount": 20,
                "timestamp": "2021-08-17T06:04:24.000Z",
                "number_of_sub_transactions": 2,
                "holding_id": ""
            }
        }
    ],
    "sub_transactions": [
        {
            "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
            "history": [
                {
                    "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
                    "timeStamp": 1629180264,
                    "value": {
                        "assetType": "otransaction",
                        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
                        "token_id": "digiCurr101",
                        "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                        "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                        "transaction_type": "TRANSFER",
                        "amount": 10,
                        "timestamp": "2021-08-17T06:04:24.000Z",
                        "number_of_sub_transactions": 0,
                        "holding_id": ""
                    }
                }
            ]
        },
        {
            "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
```


Copyright © 2025, Oracle and/or its affiliates.

```
~c81e728d9d4c2f636f067f89cc14862c",
            "history": [
                {
                    "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
                    "timeStamp": 1629180264,
                    "value": {
                        "assetType": "otransaction",
                        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c81e728d9d4c2f636f067f89cc14862c",
                        "token_id": "digiCurr101",
                        "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                        "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
                        "transaction_type": "TRANSFER",
                        "amount": 10,
                        "timestamp": "2021-08-17T06:04:24.000Z",
                        "number_of_sub_transactions": 0,
                        "holding_id": ""
                    }
                }
            ]
        }
    ]
}
```

deleteHistoricalTransactions
This method deletes older transactions from the state database.

```
@Validator(yup.date())
    public async deleteHistoricalTransactions(time_to_expiration: Date) {
        await
this.Ctx.Auth.checkAuthorization('TRANSACTION.deleteTransactions', 'TOKEN');
        return await
this.Ctx.Transaction.deleteTransactions(time_to_expiration);
    }
```

Parameters:

- `time_to_expiration Date` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted.

Return Value Example:

```
"payload": {
    "msg": "Successfuly deleted transaction older than date: Thu Aug 19 2021
11:19:36 GMT+0000 (Coordinated Universal Time).",
    "transactions": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~ec3366dd48b4ce2838f820f2f138648e6e55a07226713e59b411ff31b0d21058
"
    ]
}
```

Methods for Token Behavior Management - Mintable Behavior

- Base
- Digital Assets

Base

`issueTokens`
This method mints tokens, which are then owned by the caller of the method. The caller must have an account and the minter role. The number of tokens that can be minted is limited by the `max_mint_quantity` property of `mintable` behavior in the specification file. If the `max_mint_quantity` property is not specified, an unlimited number of tokens can be minted. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. This method can be called only by the `AccountOwner` of the account with the minter role.

```
@Validator(yup.string(), yup.number().positive())
public async issueTokens(token_id: string, quantity: number) {
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Token.mint(quantity, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to mint.

Returns:

- On success, a message with account details.

Return Value Example:

```
{
    "msg": "Successfully minted 1000 tokens to Account Id: \
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: user1)  ",
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getTotalMintedTokens`
This method returns the total number of minted tokens for a specified token. This method can be called only by a `Token Admin` or any `Org Admin` of the chaincode.

```
@Validator(yup.string())
 public async getTotalMintedTokens(token_id: string) {
     const token_asset = await this.getTokenObject(token_id);
     await this.Ctx.Auth.checkAuthorization('TOKEN.getTotalMintedTokens',
'TOKEN');
     const totalMintedTokens = await
this.Ctx.Token.getTotalMintedTokens(token_asset);
     return {
         msg: `Total minted token for Token Id: ${token_id} is $
{totalMintedTokens} tokens.`,
         quantity: totalMintedTokens
     };
 }
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON string indicating the total number of tokens.

Return Value Example:

```
{"msg":"Total minted token for Token Id: digiCurr101 is 100
tokens.","quantity":100}
```

`getNetTokens`
This method returns the total net number of tokens available in the system for a specified token. The net token total is the amount of tokens remaining after tokens are burned. In equation form: net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens. This method can be called only by a `Token Admin` or any `Org Admin` of the chaincode.

```
@Validator(yup.string())
public async getNetTokens(token_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Auth.checkAuthorization('TOKEN.getNetTokens', 'TOKEN');
    const netTokens = await this.Ctx.Token.getNetTokens(token_asset);
    return {
        msg: `Net supply of token for Token Id: ${token_id} is ${netTokens}
tokens.`,
        quantity: netTokens
    };
}
```

Parameters:

- `token_id: string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a JSON string indicating the net number of tokens.

Return Value Example:

```
{"msg":"Net supply of token for Token Id: digiCurr101 is 0
tokens.","quantity":0}
```

Digital Assets

`requestMint`
This method can be called by a minter to send a request to the minter notary to create a specified amount of tokens.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.number().positive(), yup.date(), yup.object().nullable())
public async requestMint( token_id: string, operation_id: string,
notary_org_id: string, notary_user_id: string, quantity: number,
time_to_expiration: Date, info_details?: InfoDetails) {

const token_asset = await this.getTokenObject(token_id);
const notary_account_id = await this.Ctx.Account.generateAccountId(token_id,
notary_org_id, notary_user_id);
return await this.Ctx.Token.hold(operation_id, null, notary_account_id,
quantity, time_to_expiration, token_asset, HoldOperationType.MINT,
info_details);

}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.
- `notary_org_id: string` – The membership service provider (MSP) ID of the minter notary who will process the request.
- `notary_user_id: string` – The user name or email ID of the minter notary who will process the request.
- `quantity: number` – The amount of tokens to mint.
- `time_to_expiration` – The time after which the minting request expires and is no longer valid.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
msg:
"AccountId
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin) has successfully submitted request to mint
100 tokens",
}
```

approveMint
This method can be called by a minter notary to approve a minting request.

```
@Validator(yup.string(), yup.string())
public async approveMint(token_id: string, operation_id: string) {
const token_asset = await this.getTokenObject(token_id);
return await this.Ctx.Token.executeHold(operation_id, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.

Return Value Example:

```
{
msg:
"Successfully minted 100 tokens to Account Id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin)"
}
```

rejectMint
This method can be called by a minter notary to reject a minting request.

```
@Validator(yup.string(), yup.string())
public async rejectMint(token_id: string, operation_id: string) {
const token_asset = await this.getTokenObject(token_id);
return await this.Ctx.Token.releaseHold(operation_id, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.

Return Value Example:

```
{
 msg: "Successfully rejected mint request with Operation Id 'operation1' to
```


Copyright © 2025, Oracle and/or its affiliates.

```
mint 100 tokens of token id token"
}
```

issueTokens
This method mints tokens, which are then owned by the caller of the method.

```
@Validator(yup.string(), yup.number().positive(), yup.object().nullable())
public async issueTokens(token_id: string, quantity: number, info_details?:
InfoDetails) {
const token_asset = await this.getTokenObject(token_id);
return await this.Ctx.Token.mint(quantity, token_asset, info_details);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to mint.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"Successfully minted 100 tokens to Account Id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin)"
}
```

getTotalMintedTokens
This method returns the total number of minted tokens for a specified token. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin` or `Org Auditor`.

```
@Validator(yup.string())
 public async getTotalMintedTokens(token_id: string) {
     const token_asset = await this.getTokenObject(token_id);
     await this.Ctx.Auth.checkAuthorization('TOKEN.getTotalMintedTokens',
'TOKEN');
     const totalMintedTokens = await
this.Ctx.Token.getTotalMintedTokens(token_asset);
     return {
         msg: `Total minted token for Token Id: ${token_id} is $
{totalMintedTokens} tokens.`,
         quantity: totalMintedTokens
```


Copyright © 2025, Oracle and/or its affiliates.

```
    };
}
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON string indicating the total number of tokens.

Return Value Example:

```
{"msg":"Total minted token for Token Id: digiCurr101 is 100
tokens.","quantity":100}
```

`getNetTokens`
This method returns the total net number of tokens available in the system for a specified token. The net token total is the amount of tokens remaining after tokens are burned. In equation form: net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin` or `Org Auditor`.

```
@Validator(yup.string())
public async getNetTokens(token_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Auth.checkAuthorization('TOKEN.getNetTokens', 'TOKEN');
    const netTokens = await this.Ctx.Token.getNetTokens(token_asset);
    return {
        msg: `Net supply of token for Token Id: ${token_id} is ${netTokens}
tokens.`,
        quantity: netTokens
    };
}
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON string indicating the net number of tokens.

Return Value Example:

```
{"msg":"Net supply of token for Token Id: digiCurr101 is 0
tokens.","quantity":0}
```

Methods for Token Behavior Management - Transferable Behavior

- Base

Copyright © 2025, Oracle and/or its affiliates.

- Digital Assets

Base

`transferTokens`
This method transfers tokens from the caller to a specified account. The caller of the method must have an account. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. This method can be called only by the `AccountOwner` of the account.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.number().positive())
public async transferTokens(token_id: string, to_org_id: string, to_user_id:
string, quantity: number) {
   const token_asset = await this.getTokenObject(token_id);
   const to_account_id = await this.Ctx.Account.generateAccountId(token_id,
to_org_id, to_user_id);
   return await this.Ctx.Token.transfer(to_account_id, quantity, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `to_org_id: string` – The membership service provider (MSP) ID of the receiver (payee) in the current organization.
- `to_user_id: string` – The user name or email ID of the receiver.
- `quantity: number` – The number of tokens to transfer.

Returns:

- On success, a message with details for both payer and payee accounts.

Return Value Example:

```
{
    "msg": "Successfully transferred 400 tokens from account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: user1) to account id:
oaccount~digicur~682bb71de419602af74e3f226345ef308445ca51010737900c112435f6761
52df (Org-Id: Org1MSP, User-Id: user2) ",
}
```

`bulkTransferTokens`
This method transfers tokens in bulk from the caller account to the accounts that are specified in the `flow` object. The quantities must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.The caller of this method must have an account already created. This method can be called only by the `AccountOwner` of the account.

```
@Validator(yup.string(), yup.array().of(yup.object()))
public async bulkTransferTokens(token_id: string, flow: object[]) {
     const token_asset = await this.getTokenObject(token_id);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Token.bulkTransfer(flow, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `flow : object[]` – An array of JSON objects that specify receivers and quantities.
  - `to_orgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
  - `userId: string` – The user name or email ID of the receiver.
  - `quantity: number` – The number of tokens to transfer.

  You specify the `flow` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code:

  ```
  [
    { "to_org_id": "Org1MSP", "to_user_id": "user1", "quantity": 10 },
    { "to_org_id": "Org1MSP", "to_user_id": "user2", "quantity": 10 }
  ]
  ```

  CLI / Postman:

  ```
  "[{ \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user1\", \"quantity\":
  10 }, { \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user2\",
  \"quantity\": 10 }]"
  ```

Returns:

- A message indicating success.

Return Value Example:

```
{
    "msg": "Successfully transferred 20 tokens from Account Id
'oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df' (Org-Id: Org1MSP, User-Id: admin).",
    "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
    "sub_transactions": [
        {
            "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
            "amount": 10
        },
        {
            "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "amount": 10
        }
    ]
}
```

Digital Assets

`transferTokens`
This method transfers tokens from the caller to a specified account.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.number().positive(),
yup.object().nullable())
public async transferTokens(token_id: string, to_org_id: string, to_user_id:
string, quantity: number, info_details?: InfoDetails) {
const token_asset = await this.getTokenObject(token_id);
const to_account_id = await this.Ctx.Account.generateAccountId(token_id,
to_org_id, to_user_id);
return await this.Ctx.Token.transfer(to_account_id, quantity, token_asset,
info_details);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `to_org_id: string` – The membership service provider (MSP) ID of the receiver (payee) in the current organization.
- `to_user_id: string` – The user name or email ID of the receiver.
- `quantity: number` – The number of tokens to transfer.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
 msg: "Successfully transferred 100 tokens from account id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin) to account id:
oaccount~7yuijg39b4e1e4136dd86a806020c97a930909325340481b8fdhjklliugbv699
(Org-Id: Org1MSP, User-Id: user)",
}
```

`bulkTransferTokens`
This method transfers tokens in bulk from the caller account to the accounts that are specified in the `flow` object. The quantities must be within the decimal values specified by the `decimal`

Copyright © 2025, Oracle and/or its affiliates.

parameter of the `divisible` behavior in the specification file.The caller of this method must have an account already created. This method can be called only by the `AccountOwner` of the account.

```
@Validator(yup.string(), yup.array().of(yup.object()))
public async bulkTransferTokens(token_id: string, flow: object[]) {
     const token_asset = await this.getTokenObject(token_id);
     return await this.Ctx.Token.bulkTransfer(flow, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `flow : object[]` – An array of JSON objects that specify receivers and quantities.
  - `to_orgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
  - `userId: string` – The user name or email ID of the receiver.
  - `quantity: number` – The number of tokens to transfer.

  You specify the `flow` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code:

  ```
  [
    { "to_org_id": "Org1MSP", "to_user_id": "user1", "quantity": 10 },
    { "to_org_id": "Org1MSP", "to_user_id": "user2", "quantity": 10 }
  ]
  ```

  CLI / Postman:

  ```
  "[{ \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user1\", \"quantity\":
  10 }, { \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user2\",
  \"quantity\": 10 }]"
  ```

Returns:

- A message indicating success.

Return Value Example:

```
{
    "msg": "Successfully transferred 20 tokens from Account Id
'oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df' (Org-Id: Org1MSP, User-Id: admin).",
    "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
    "sub_transactions": [
        {
            "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
```


Copyright © 2025, Oracle and/or its affiliates.

```
622f",
            "amount": 10
        },
        {
            "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
            "amount": 10
        }
    ]
}
```

Methods for Token Behavior Management - Holdable Behavior

- Base
- Digital Assets

Base

holdTokens
This method creates a hold on behalf of the owner of the tokens with the `to_account_id` account. A notary account is specified, which is responsible to either complete or release the hold. When the hold is created, the specified token balance from the payer is put on hold. A held balance cannot be transferred until the hold is either completed or released. The caller of this method must have an account already created. This method can be called only by the `AccountOwner` of the account.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string(), yup.string(), yup.number().positive(), yup.date())
public async holdTokens(
    token_id: string,
    operation_id: string,
    to_org_id: string,
    to_user_id: string,
    notary_org_id: string,
    notary_user_id: string,
    quantity: number,
    time_to_expiration: Date
) {
    const token_asset = await this.getTokenObject(token_id);
    const to_account_id = await this.Ctx.Account.generateAccountId(token_id,
to_org_id, to_user_id);
    const notary_account_id = await
this.Ctx.Account.generateAccountId(token_id, notary_org_id, notary_user_id);
    return await this.Ctx.Token.hold(operation_id, to_account_id,
notary_account_id, quantity, time_to_expiration, token_asset);
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_org_id: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `to_user_id: string` – The user name or email ID of the receiver.
- `notary_org_id: string` – The membership service provider (MSP) ID of the notary in the current organization.
- `notary_user_id: string` – The user name or email ID of the notary.
- `quantity: number` – The number of tokens to put on hold.
- `time_to_expiration` – The time when the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12:46:06Z`.

Returns:

- On success, a message with the caller's account and hold details.

Return Value Example:

```
{
  "msg":"AccountId
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP , User-Id: admin) is   successfully holding 10 tokens"
}
```

`executeHoldTokens`
This method completes a hold on a token. A quantity of tokens previously held by a token owner is transferred to a receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can be called only by the `AccountOwner` ID with the `notary` role for the specified operation ID. The hold can only be completed by the notary.

```
@Validator(yup.string(), yup.string(), yup.number().positive())
public async executeHoldTokens(token_id: string, operation_id: string,
quantity: number) {
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Token.executeHold(operation_id, quantity,
token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity: number` – The number of on-hold tokens to transfer.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message with the caller's account ID and the quantity of the transaction.

Return Value Example:

```
{
 "msg":"Account Id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin) is successfully executed '10' tokens
from Operation Id 'opr_121'."
}
```

`releaseHoldTokens`
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `AccountOwner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
@Validator(yup.string(), yup.string())
public async releaseHoldTokens(token_id: string, operation_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Token.releaseHold(operation_id, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a message indicating that the hold was released.

Return Value Example:

```
{
 "msg":"Successfully released '10' tokens from Operation Id 'opr_121' to
Account Id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: user1)."
}
```

`getOnHoldIds`
This method returns a list of all of the holding IDs for a specified account. This method can be called by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getOnHoldIds(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getOnHoldIds", "TOKEN",
{ account_id });
    return await this.Ctx.Account.getOnHoldIds(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON list of holding IDs.

Return Value Example:

```
{"msg":"Holding Ids are: ohold~digicur~digiCurr101~opr_121","holding_ids":
["ohold~digicur~digiCurr101~opr_121"]}
```

`getOnHoldDetailsWithOperationId`
This method returns the on-hold transaction details for a specified operation ID and token. This method can be called by anyone.

```
@Validator(yup.string(), yup.string())
public async getOnHoldDetailsWithOperationId(token_id: string, operation_id:
string) {
    return await this.Ctx.Hold.getOnHoldDetailsWithOperationId(token_id,
operation_id);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON hold object that includes the following properties:
- `holding_id` – The holding ID of the transaction.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `from_account_id` – The account ID of the current owner of the on-hold tokens.
- `to_account_id` – The account ID of the receiver.
- `notary_account_id` – The account ID of the notary.
- `token_id: string` – The ID of the saved token.
- `quantity` – The amount of tokens that are on hold for the holding ID.

Copyright © 2025, Oracle and/or its affiliates.

- `time_to_expiration` – The duration until the hold expires.

Return Value Example:

```
{
    "assetType": "ohold",
    "holding_id": "ohold~digicur~digiCurr101~opr_121",
    "operation_id": "opr_121",
    "token_name": "digicur",
    "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
    "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "notary_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
    "token_id": "digiCurr101",
    "quantity": 10,
    "time_to_expiration": "2022-08-01T18:30:00.000Z"
}
```

`getOnHoldBalanceWithOperationId`
This method returns the on-hold balance for a specified operation ID and token. This method can be called by anyone.

```
@Validator(yup.string(), yup.string())
public async getOnHoldBalanceWithOperationId(token_id: string, operation_id:
string) {
    return await this.Ctx.Hold.getOnHoldBalanceWithOperationId(token_id,
operation_id);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON string indicating the holding balance.

Return Value Example:

```
{
    "msg": "Current Holding Balance of Operation 'opr_121' for token
'digiCurr101' is: 10",
    "holding_balance": 10
}
```


Copyright © 2025, Oracle and/or its affiliates.

Digital Assets

`holdTokens`
This method creates a hold on behalf of the owner of the tokens with the `to_account_id` account.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string(), yup.string(), yup.number().positive(), yup.date(),
yup.object().nullable())
  public async holdTokens( token_id: string, operation_id: string, to_org_id:
string, to_user_id: string, notary_org_id: string, notary_user_id: string,
quantity: number, time_to_expiration: Date, info_details?: InfoDetails) {
    const token_asset = await this.getTokenObject(token_id);
    const to_account_id = await this.Ctx.Account.generateAccountId(token_id,
to_org_id, to_user_id);
    const notary_account_id = await
this.Ctx.Account.generateAccountId(token_id, notary_org_id, notary_user_id);
    return await this.Ctx.Token.hold(operation_id, to_account_id,
notary_account_id, quantity, time_to_expiration, token_asset,
HoldOperationType.TRANSFER, info_details);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_org_id: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `to_user_id: string` – The user name or email ID of the receiver.
- `notary_org_id: string` – The membership service provider (MSP) ID of the notary in the current organization.
- `notary_user_id: string` – The user name or email ID of the notary.
- `quantity: number` – The number of tokens to put on hold.
- `time_to_expiration` – The time when the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12:46:06Z`.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
msg:
"AccountId
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin) is successfully holding 100 tokens",
}
```

`executeHoldTokens`
This method completes a hold on a token. A quantity of tokens previously held by a token owner is transferred to a receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can be called only by the `AccountOwner` ID with the `notary` role for the specified operation ID. The hold can only be completed by the notary.

```
@Validator(yup.string(), yup.string(), yup.number().positive())
public async executeHoldTokens(token_id: string, operation_id: string,
quantity: number) {
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Token.executeHold(operation_id, quantity,
token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity: number` – The number of on-hold tokens to transfer.

Returns:

- On success, a message with the caller's account ID and the quantity of the transaction.

Return Value Example:

```
{
 "msg":"Account Id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin) is successfully executed '10' tokens
from Operation Id 'opr_121'."
}
```

`releaseHoldTokens`
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `AccountOwner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
@Validator(yup.string(), yup.string())
public async releaseHoldTokens(token_id: string, operation_id: string) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Token.releaseHold(operation_id, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a message indicating that the hold was released.

Return Value Example:

```
{
 "msg":"Successfully released '10' tokens from Operation Id 'opr_121' to
Account Id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: user1)."
}
```

`getOnHoldIds`
This method returns a list of all of the holding IDs for a specified account. This method can be called by a `Token Admin` or `Token Auditor` of the chaincode, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getOnHoldIds(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getOnHoldIds", "TOKEN",
{ account_id });
    return await this.Ctx.Account.getOnHoldIds(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON list of holding IDs.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"Holding Ids are: ohold~digicur~digiCurr101~opr_121","holding_ids":
["ohold~digicur~digiCurr101~opr_121"]}
```

`getOnHoldDetailsWithOperationId`
This method returns the on-hold transaction details for a specified operation ID and token. This method can be called by a `Token Admin` or `Token Auditor` of the chaincode, or by a transaction participant (sender, recipient, notary).

```
@Validator(yup.string(), yup.string())
public async getOnHoldDetailsWithOperationId(token_id: string, operation_id:
string) {
    return await this.Ctx.Hold.getOnHoldDetailsWithOperationId(token_id,
operation_id);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON hold object that includes the following properties:
- `holding_id` – The holding ID of the transaction.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `from_account_id` – The account ID of the current owner of the on-hold tokens.
- `to_account_id` – The account ID of the receiver.
- `notary_account_id` – The account ID of the notary.
- `token_id: string` – The ID of the saved token.
- `quantity` – The amount of tokens that are on hold for the holding ID.
- `time_to_expiration` – The duration until the hold expires.

Return Value Example:

```
{
    "assetType": "ohold",
    "holding_id": "ohold~digicur~digiCurr101~opr_121",
    "operation_id": "opr_121",
    "token_name": "digicur",
    "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
    "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "notary_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
    "token_id": "digiCurr101",
    "quantity": 10,
    "time_to_expiration": "2022-08-01T18:30:00.000Z"
}
```

`getOnHoldBalanceWithOperationId`
This method returns the on-hold balance for a specified operation ID and token. This method can be called by a `Token Admin` or `Token Auditor` of the chaincode, or by a transaction participant (sender, recipient, notary).

```
@Validator(yup.string(), yup.string())
public async getOnHoldBalanceWithOperationId(token_id: string, operation_id:
string) {
    return await this.Ctx.Hold.getOnHoldBalanceWithOperationId(token_id,
operation_id);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON string indicating the holding balance.

Return Value Example:

```
{
    "msg": "Current Holding Balance of Operation 'opr_121' for token
'digiCurr101' is: 10",
    "holding_balance": 10
}
```

Methods for Token Behavior Management - Burnable Behavior

- Base
- Digital Assets

Base

`burnTokens`
This method deactivates, or burns, tokens from the transaction caller's account. The caller of this method must have an account and the burner role. The quantity must be within the

Copyright © 2025, Oracle and/or its affiliates.

decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. This method can be called by the `AccountOwner` of the account with the burner role.

```
@Validator(yup.string(), yup.number().positive())
public async burnTokens(token_id: string, quantity: number) {
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Token.burn(quantity, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to burn.

Returns:

- On success, a success message with the quantity of tokens burned and the account ID.

Return Value Example:

```
{
    "msg": "Successfully burned 10 tokens from account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin)"
}
```

Digital Assets

`requestBurn`
This method can be called by a burner to send a request to the burner notary to destroy a specified amount of tokens, which must be less than or equal to their available balance. When a burn request starts, the specified amount is immediately deducted from the available balance and added to the `onhold_burn_balance` field. If the request is approved, the tokens are burned. If the request is rejected, the tokens are returned from the `onhold_burn_balance` field to the available balance.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.number().positive(), yup.date(), yup.object().nullable())

public async requestBurn( token_id: string, operation_id: string,
notary_org_id: string, notary_user_id: string, quantity: number,
time_to_expiration: Date, info_details?: InfoDetails ) {

const token_asset = await this.getTokenObject(token_id);
const notary_account_id = await this.Ctx.Account.generateAccountId(token_id,
notary_org_id, notary_user_id);
return await this.Ctx.Token.hold(operation_id, null, notary_account_id,
quantity, time_to_expiration, token_asset, HoldOperationType.BURN, null,
description);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.
- `notary_org_id: string` – The membership service provider (MSP) ID of the burner notary who will process the request.
- `notary_user_id: string` – The user name or email ID of the burner notary who will process the request.
- `quantity: number` – The amount of tokens to burn.
- `time_to_expiration` – The time after which the burning request expires and is no longer valid.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"AccountId
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin) has successfully submitted request to mint
100 tokens",
}
```

approveBurn
This method can be called by a burner notary to approve a burning request.

```
@Validator(yup.string(), yup.string())
public async approveBurn(token_id: string, operation_id: string) {
const token_asset = await this.getTokenObject(token_id);
return await this.Ctx.Token.executeHold(operation_id, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.

Return Value Example:

```
{
msg:
"Successfully burned 100 tokens from account id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
```


Copyright © 2025, Oracle and/or its affiliates.

```
(Org-Id: Org1MSP, User-Id: admin)"
}
```

rejectBurn
This method can be called by a burner notary to reject a burning request.

```
@Validator(yup.string(), yup.string())
public async rejectBurn(token_id: string, operation_id: string) {
const token_asset = await this.getTokenObject(token_id);
return await this.Ctx.Token.releaseHold(operation_id, token_asset);
}
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.

Return Value Example:

```
{
 msg: "Successfully rejected burn request with Operation Id 'operation1' to
burn 100 tokens of token id token",
}
```

getAccountOnHoldBurnBalance
This method returns the on-hold burn balance for a specified user. This method can be called only by the `Token Admin`, `Token Auditor`, `Org Admin`, `Org Auditor`, or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async getAccountOnHoldBurnBalance(token_id: string, org_id: string,
user_id: string) {
  const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
  await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountOnHoldBurnBalance",
"TOKEN", { account_id });
  return await this.Ctx.Account.getAccountOnHoldBurnBalance(account_id);
}
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Return Value Example:

```
{
  "msg": "Total On Hold Burning Balance is: 10",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "onhold_burn_balance": 10
}
```

burnTokens
This method deactivates, or burns, tokens from the transaction caller's account.

```
@Validator(yup.string(), yup.number().positive(), yup.object().nullable())

public async burnTokens(token_id: string, quantity: number, info_details?:
InfoDetails) {
const token_asset = await this.getTokenObject(token_id);
return await this.Ctx.Token.burn(quantity, token_asset, info_details);
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to burn.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"Successfully burned 100 tokens from account id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin)"
}
```

Custom Methods

You can use the token SDK methods to write custom methods for your business application.

To avoid double-spending, do not combine multiple async functions that operate on the same key-value pairs in the state database. Instead, use the `bulkTransferTokens` method to make multiple transfers in one method.

The following example shows how to use token SDK methods in custom methods. When the `buyTicket` method is called, it transfers 20 tokens from the caller's account to the seller's account, and returns the transaction message of the transfer.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string())
public async buyTicket(token_id: string, seller_org_id: string,
```


Copyright © 2025, Oracle and/or its affiliates.

```
seller_user_id: string) {
    const token = await this.getTokenObject(token_id);

    /**
    * The following method this.Ctx.Account.generateAccountId(token_id,
seller_org_id, seller_user_id) generates account id of the seller.
    */
    const seller_account_id = await
this.Ctx.Account.generateAccountId(token_id, seller_org_id, seller_user_id);

    /**
    * The following method this.Ctx.Token.transfer(seller_account_id, 20,
token) transfers the quantity 20 from caller's
    * account & to seller's account.
    */
    const transaction = await this.Ctx.Token.transfer(seller_account_id, 20,
token);

    return transaction;
}
```

If you use more than one token SDK method in a custom method, do not use methods that will affect the same key-value pairs in the state database. The following example shows the incorrect way to make multiple transfers:

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string())
public async sendTokens(token_id: string, user1_org_id: string,
user1_user_id: string, user2_org_id: string, user2_user_id: string) {
    const token = await this.getTokenObject(token_id);
    const user1_account_id = await Account.generateAccountId(token_id,
user1_org_id, user1_user_id);
    const user2_account_id = await Account.generateAccountId(token_id,
user2_org_id, user2_user_id);
    await token.transfer(user1_account_id, 20);
    await token.transfer(user2_account_id, 30);
}
```

Instead, use the `bulkTransferTokens` method to transfer to multiple accounts from the caller's account, as shown in the following code snippet.

```
bulkTransferTokens(token_id: string, flow: object[])
```

> **Note**
>
> If you use more than one token SDK method in a custom method that might affect the same key-value pairs in the state database, enable the MVCC optimization for token chaincodes. For more information, see MVCC Optimization.

Token SDK Methods

- Access Control Management

Copyright © 2025, Oracle and/or its affiliates.

- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Holdable Behavior
  - Burnable Behavior

Methods for Access Control Management

The token SDK provides an access control function. Some methods can be called only by a `Token Admin`, `Org Admin`, or `AccountOwner` of the token. You can use this feature to ensure that operations are carried out only by the intended users. Any unauthorized access results in an error. To use the access control function, import the `Authorization` class from the `../lib/auth` module.

```
import { Authorization } from '../lib/auth';
```

- Base
- Digital Assets

Base

`addAdmin`
This method adds a user as a `Token Admin` of the token chaincode.

```
Ctx.Admin.addAdmin(org_id: string, user_id: string)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a promise message with a JSON object that lists details for the user added as a `Token Admin` of the token chaincode. On error, a rejection with an error message.

Return Value Example:

```
{
    "msg": "Successfully added Admin (Org_Id: Org1MSP, User_Id: user1)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

removeAdmin
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.Admin.removeAdmin(org_id: string, user_id: string)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a promise message with a JSON object that lists details for the user who is no longer a `Token Admin` of the token chaincode. On error, a rejection with an error message.

Return Value Example:

```
{
    "msg": "Successfully removed Admin (Org_Id: Org1MSP, User_Id: user1)"
}
```

isUserTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`.

```
Ctx.Auth.isUserTokenAdmin()
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- A Boolean response and an error message if an error is encountered.

getAllAdmins
This method returns a list of all users who are a `Token Admin` of the token chaincode.

```
Ctx.Admin.getAllAdmins()
```

Parameters:

- none

Returns:

- On success, a promise with a JSON object that lists details for all users who are a `Token Admin` of the token chaincode. On error, a rejection with an error message.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "admins": [
        {
            "orgId": "Org1MSP",
            "userId": "admin"
        }
    ]
}
```

checkAuthorization
Use this method to add an access control check to an operation. Certain token methods can be run only by a `Token Admin` or `AccountOwner` of the token or by the `MultipleAccountOwner` for users with multiple accounts. The access control mapping is described in the `../lib/constant.ts` file. You can modify access control by editing the `../lib/constant.ts` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
export const TOKENACCESS = {
  ADMIN: {
    isUserTokenAdmin: ["Admin", "OrgAdmin"],
    addTokenAdmin: ["Admin"],
    removeTokenAdmin: ["Admin"],
    getAllAdmins: ["Admin", "OrgAdmin"],
    addOrgAdmin: ["Admin", "OrgAdminForOrgId"],
    removeOrgAdmin: ["Admin", "OrgAdminForOrgId"],
    getOrgAdmins: ["Admin", "OrgAdmin"],
  },
  TOKEN: {
    save: ["Admin"],
    getAllTokens: ["Admin", "OrgAdmin"],
    get: ["Admin", "OrgAdmin"],
    update: ["Admin"],
    getDecimals: ["Admin", "OrgAdmin"],
    getTokensByName: ["Admin", "OrgAdmin"],
    addRoleMember: ["Admin", "OrgAdminRoleCheck"],
    removeRoleMember: ["Admin", "OrgAdminRoleCheck"],
    isInRole: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    getTotalMintedTokens: ["Admin", "OrgAdmin"],
    getNetTokens: ["Admin", "OrgAdmin"],
    getTokenHistory: ["Admin", "OrgAdmin"],
  },
  ROLE: {
    getAccountsByRole: ["Admin"],
    getOrgAccountsByRole: ["Admin", "OrgAdminForOrgId"],
    getUsersByRole: ["Admin"],
    getOrgUsersByRole: ["Admin", "OrgAdminForOrgId"],
  },
  TRANSACTION: {
    deleteTransactions: ["Admin"],
  },ACCOUNT: {
    createAccount: ["Admin", "OrgAdminForOrgId"],
```


Copyright © 2025, Oracle and/or its affiliates.

```
    associateToken: ["Admin", "OrgAdminForAccountId"],
    getAllAccounts: ["Admin"],
    getAllOrgAccounts: ["Admin", "OrgAdminForOrgId"],
    getAccountsByUser: ["Admin", "OrgAdminForOrgId", "MultipleAccountOwner"],
    getAccount: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    history: ["Admin", "AccountOwner"],
    getAccountTransactionHistory: ["Admin", "OrgAdminForAccountId",
"AccountOwner"],
    getAccountTransactionHistoryWithFilters: ["Admin",
"OrgAdminForAccountId", "AccountOwner"],
    getSubTransactionsById: ["Admin", "TransactionInvoker"],
    getSubTransactionsByIdWithFilters: ["Admin", "TransactionInvoker"],
    getAccountBalance: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    getAccountOnHoldBalance: ["Admin", "OrgAdminForAccountId",
"AccountOwner"],
    getOnHoldIds: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    getConversionHistory: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
  },
  ACCOUNT_STATUS: {
    get: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    history: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    activateAccount: ["Admin", "OrgAdminForOrgId"],
    suspendAccount: ["Admin", "OrgAdminForOrgId"],
    deleteAccount: ["Admin", "OrgAdminForOrgId"],
  },
  TOKEN_CONVERSION: {
    initializeExchangePoolUser: ["Admin"],
    addConversionRate: ["Admin"],
    updateConversionRate: ["Admin"],
    getConversionRate: ["Admin", "OrgAdmin", "AnyAccountOwner"],
    getConversionRateHistory: ["Admin", "OrgAdmin", "AnyAccountOwner"],
    tokenConversion: ["Admin", "AnyAccountOwner"],
    getExchangePoolUser: ["Admin"],
  },
}
```

```
await this.Ctx.Auth.checkAuthorization(<parameters>);
```

Parameters:

- `classFuncName: string` – The map value between the class and methods as described in the `../lib/constant.ts` file.
- `...args` – A variable argument where `args[0]` takes the constant `'TOKEN'` and `args[1]` takes the `account_id` to add an access control check for an `AccountOwner`. To add an access control check for a `MultipleAccountOwner`, `args[1]` takes the `org_id` and `args[2]` takes the `user_id`.

Returns:

- On success, a promise. On error, a rejection with an error message.

Copyright © 2025, Oracle and/or its affiliates.

addOrgAdmin
This method adds a user as an `Org Admin` of the organization.

```
Ctx.Admin.addOrgAdmin(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

removeOrgAdmin
This method removes a user as an `Org Admin` of the organization.

```
Ctx.Admin.removeOrgAdmin(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
  "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

getOrgAdmins
This method returns a list of all users who are an `Org Admin` of an organization.

```
Ctx.Admin.getAllOrgAdmins()
```

Parameters:

- none

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, an array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin1"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin2"
        }
    ]
}
```

Digital Assets

`addAdmin`
This method adds a user as a `Token Admin` of the token chaincode.

```
Ctx.Admin.addAdmin(org_id: string, user_id: string)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a promise message with a JSON object that lists details for the user added as a `Token Admin` of the token chaincode. On error, a rejection with an error message.

Return Value Example:

```
{
    "msg": "Successfully added Admin (Org_Id: Org1MSP, User_Id: user1)"
}
```

`removeAdmin`
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.Admin.removeAdmin(org_id: string, user_id: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a promise message with a JSON object that lists details for the user who is no longer a `Token Admin` of the token chaincode. On error, a rejection with an error message.

Return Value Example:

```
{
    "msg": "Successfully removed Admin (Org_Id: Org1MSP, User_Id: user1)"
}
```

`isUserTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`.

```
Ctx.Auth.isUserTokenAdmin()
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- A Boolean response and an error message if an error is encountered.

`getAllAdmins`
This method returns a list of all users who are a `Token Admin` of the token chaincode.

```
Ctx.Admin.getAllAdmins()
```

Parameters:

- none

Returns:

- On success, a promise with a JSON object that lists details for all users who are a `Token Admin` of the token chaincode. On error, a rejection with an error message.

Return Value Example:

```
{
    "admins": [
        {
            "orgId": "Org1MSP",
            "userId": "admin"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    ]
}
```

checkAuthorization
Use this method to add an access control check to an operation. Certain token methods can be run only by a `Token Admin` or `AccountOwner` of the token or by the `MultipleAccountOwner` for users with multiple accounts. The access control mapping is described in the `../lib/constant.ts` file. You can modify access control by editing the `../lib/constant.ts` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
export const TOKENACCESS = {
  ADMIN: {
    isUserTokenAdmin: ["Admin", "OrgAdmin"],
    addTokenAdmin: ["Admin"],
    removeTokenAdmin: ["Admin"],
    getAllAdmins: ["Admin", "OrgAdmin"],
    addOrgAdmin: ["Admin", "OrgAdminForOrgId"],
    removeOrgAdmin: ["Admin", "OrgAdminForOrgId"],
    getOrgAdmins: ["Admin", "OrgAdmin"],
  },
  TOKEN: {
    save: ["Admin"],
    getAllTokens: ["Admin", "OrgAdmin"],
    get: ["Admin", "OrgAdmin"],
    update: ["Admin"],
    getDecimals: ["Admin", "OrgAdmin"],
    getTokensByName: ["Admin", "OrgAdmin"],
    addRoleMember: ["Admin", "OrgAdminRoleCheck"],
    removeRoleMember: ["Admin", "OrgAdminRoleCheck"],
    isInRole: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    getTotalMintedTokens: ["Admin", "OrgAdmin"],
    getNetTokens: ["Admin", "OrgAdmin"],
    getTokenHistory: ["Admin", "OrgAdmin"],
  },
  ROLE: {
    getAccountsByRole: ["Admin"],
    getOrgAccountsByRole: ["Admin", "OrgAdminForOrgId"],
    getUsersByRole: ["Admin"],
    getOrgUsersByRole: ["Admin", "OrgAdminForOrgId"],
  },
  TRANSACTION: {
    deleteTransactions: ["Admin"],
  },ACCOUNT: {
    createAccount: ["Admin", "OrgAdminForOrgId"],
    associateToken: ["Admin", "OrgAdminForAccountId"],
    getAllAccounts: ["Admin"],
    getAllOrgAccounts: ["Admin", "OrgAdminForOrgId"],
    getAccountsByUser: ["Admin", "OrgAdminForOrgId", "MultipleAccountOwner"],
    getAccount: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    history: ["Admin", "AccountOwner"],
    getAccountTransactionHistory: ["Admin", "OrgAdminForAccountId",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"AccountOwner"],
    getAccountTransactionHistoryWithFilters: ["Admin",
"OrgAdminForAccountId", "AccountOwner"],
    getSubTransactionsById: ["Admin", "TransactionInvoker"],
    getSubTransactionsByIdWithFilters: ["Admin", "TransactionInvoker"],
    getAccountBalance: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    getAccountOnHoldBalance: ["Admin", "OrgAdminForAccountId",
"AccountOwner"],
    getOnHoldIds: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    getConversionHistory: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
  },
  ACCOUNT_STATUS: {
    get: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    history: ["Admin", "OrgAdminForAccountId", "AccountOwner"],
    activateAccount: ["Admin", "OrgAdminForOrgId"],
    suspendAccount: ["Admin", "OrgAdminForOrgId"],
    deleteAccount: ["Admin", "OrgAdminForOrgId"],
  },
  TOKEN_CONVERSION: {
    initializeExchangePoolUser: ["Admin"],
    addConversionRate: ["Admin"],
    updateConversionRate: ["Admin"],
    getConversionRate: ["Admin", "OrgAdmin", "AnyAccountOwner"],
    getConversionRateHistory: ["Admin", "OrgAdmin", "AnyAccountOwner"],
    tokenConversion: ["Admin", "AnyAccountOwner"],
    getExchangePoolUser: ["Admin"],
  },
}


await this.Ctx.Auth.checkAuthorization(<parameters>);
```

Parameters:

- `classFuncName: string` – The map value between the class and methods as described in the `../lib/constant.ts` file.
- `...args` – A variable argument where `args[0]` takes the constant `'TOKEN'` and `args[1]` takes the `account_id` to add an access control check for an `AccountOwner`. To add an access control check for a `MultipleAccountOwner`, `args[1]` takes the `org_id` and `args[2]` takes the `user_id`.

Returns:

- On success, a promise. On error, a rejection with an error message.

`addOrgAdmin`
This method adds a user as an `Org Admin` of the organization.

```
Ctx.Admin.addOrgAdmin(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`removeOrgAdmin`
This method removes a user as an `Org Admin` of the organization.

```
Ctx.Admin.removeOrgAdmin(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
  "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

`getOrgAdmins`
This method returns a list of all users who are an `Org Admin` of an organization.

```
Ctx.Admin.getAllOrgAdmins()
```

Parameters:

- none

Returns:

- On success, an array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{
    "admins": [
        {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "org_id": "Org1MSP",
            "user_id": "orgadmin"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin1"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin2"
        }
    ]
}
```

addTokenAuditor
This method adds a user as a `Token Auditor` of the chaincode.

```
this.Ctx.Admin.addTokenAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"cd81f6c4c9e7c18ece357dbf5c139ef66ef2d6566be3b14de5f6d0a3fd4bb2f0",
        "payload": {
            "msg": "Successfully added Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20009",
        "blockNumber": 196
    }
}
```

removeTokenAuditor
This method removes a user as a `Token Auditor` of the chaincode.

```
this.Ctx.Admin.removeTokenAuditor(org_id, user_id)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"a886a6040fbc76374a3c78c89ab0ffc9f7b8391cc5239b169bf3b878cf40c67b",
        "payload": {
            "msg": "Successfully removed Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 219
    }
}
```

`getTokenAuditors`
This method returns all `Token Auditors` of the chaincode.

```
this.Ctx.Admin.getAllTokenAuditors()
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "payload": {
            "auditors": [
                {
                    "org_id": "CB",
                    "user_id": "auditor_user_cb"
                }
            ]
        },
        "encode": "JSON"
    }
}
```


Copyright © 2025, Oracle and/or its affiliates.

addOrgAuditor
This method adds a user as a `Org Auditor` of the chaincode.

```
this.Ctx.Admin.addOrgAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"44bbad35a1478cb714e32f7cfd551897868a203520aab9cea5771d3aadc1cf03",
        "payload": {
            "msg": "Successfully added Org Auditor (Org_Id: CB, User_Id: cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20009",
        "blockNumber": 198
    }
}
```

removeOrgAuditor
This method removes a user as a `Org Auditor` of the chaincode.

```
this.Ctx.Admin.removeOrgAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "error": "",
    "result": {
        "txid":
"c3bc720461004a53b37c68d4bb264858b88d980bc093a0a3ebb62a32974fb306",
        "payload": {
            "msg": "Successfully removed Org Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 221
    }
}
```

getOrgAuditors
This method returns all `Org Auditors` of the chaincode.

```
this.Ctx.Admin.getAllOrgAuditors()
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "payload": {
            "auditors": [
                {
                    "org_id": "FI1",
                    "user_id": "auditor_user_fi1"
                },
                {
                    "org_id": "FI2",
                    "user_id": "auditor_user_fi2"
                }
            ]
        },
        "encode": "JSON"
    }
}
```

Methods for Token Configuration Management

save
This method creates a token and saves its properties in the state database.

```
Ctx.Token.save(token: <Instance of Token Class>,extraMetadata?:any)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise message with token details. On error, a rejection with an error message.

Return Value Example:

```
{
   "assetType":"otoken",
   "token_id":"digiCurr101",
   "token_name":"digicur",
   "token_type":"fungible",
   "behaviors":[
      "divisible",
      "mintable",
      "transferable",
      "burnable",
      "roles"
   ],
   "roles":{
      "minter_role_name":"minter"
   },
   "mintable":{
      "max_mint_quantity":1000
   },
   "divisible":{
      "decimal":2
   },
   "currency_name":"DOLLAR",
   "token_to_currency_ratio":1
}
```

`update`
This method updates token properties. After a token asset is created, you update only the `token_desc` value and its custom properties.

```
Ctx.Token.update(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise message with token details. On error, a rejection with an error message.

Return Value Example:

```
{
   "assetType":"otoken",
   "token_id":"digiCurr101",
```


Copyright © 2025, Oracle and/or its affiliates.

```
   "token_name":"digicur",
   "token_desc":"Digital Currency equiv of dollar",
   "token_type":"fungible",
   "behaviors":[
      "divisible",
      "mintable",
      "transferable",
      "burnable",
      "roles"
   ],
   "roles":{
      "minter_role_name":"minter"
   },
   "mintable":{
      "max_mint_quantity":1000
   },
   "divisible":{
      "decimal":2
   },
   "currency_name":"DOLLAR",
   "token_to_currency_ratio":1
}
```

`getTokenDecimals`
This method returns the number of decimal places available for a fractional token. If the `divisible` behavior is not specified, then the default value is 0.

```
Ctx.Token.getTokenDecimals(token_id: string)
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, the decimal places of the token, in the number data type. On error, it returns with an error message.

Return Value Example:

```
1
```

`get`
This method returns a token object if it is present in the state database.

```
Ctx.Token.get(token_id: string)
```

Parameters:

- `token_id: string` – The ID of the token to return.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise with the JSON representation of the token. On error, a rejection with an error message.

Return Value Example:

```
{
    "assetType": "otoken",
    "token_id": "token1",
    "token_name": "account",
    "token_desc": "Token 1",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "holdable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner",
        "notary_role_name": "notary"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "divisible": {
        "decimal": 1
    },
    "token_to_currency_ratio": 2,
    "currency_representation": "EURO"
}
```

`history`
This method returns history for the specified token.

```
Ctx.Token.history(tokenId)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a promise with an array of the account history details for the specified token. On error, a rejection with an error message.

Return Value Example:

```
[
    {
        "trxId":
"0d75f09446a60088afb948c6aca046e261fddcd43df416076201cdc5565f1a35",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "timeStamp": "2023-09-01T16:48:41.000Z",
            "value": {
                "assetType": "otoken",
                "token_id": "token",
                "token_name": "fiatmoneytok",
                "token_desc": "updatedDesc",
                "token_standard": "ttf+",
                "token_type": "fungible",
                "token_unit": "fractional",
                "behaviors": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter"
                },
                "mintable": {
                    "max_mint_quantity": 1000
                },
                "divisible": {
                    "decimal": 2
                }
            }
        },
        {
            "trxId":
"3666344878b043b65d5b821cc79c042ba52aec467618800df5cf14eac69f72fa",
            "timeStamp": "2023-08-31T20:24:55.000Z",
            "value": {
                "assetType": "otoken",
                "token_id": "token",
                "token_name": "fiatmoneytok",
                "token_standard": "ttf+",
                "token_type": "fungible",
                "token_unit": "fractional",
                "behaviors": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter"
                },
                "mintable": {
                    "max_mint_quantity": 1000
                },
                "divisible": {
                    "decimal": 2
                }
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    }
]
```

getAllTokens
This method returns all the token assets saved in the state database. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Token.getAllTokens()
```

Parameters:

- none

Returns:

- On success, it returns a promise with all the token assets. On error, it returns an error message.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"98e0a0a115803d25b843d630e6b23c435a192a03eb0a301fc9375f05da49a8b2",
        "payload": [
            {
                "key": "token1",
                "valueJson": {
                    "assetType": "otoken",
                    "token_id": "token1",
                    "token_name": "vtok",
                    "token_type": "fungible",
                    "behaviours": [
                        "divisible",
                        "mintable",
                        "transferable",
                        "burnable",
                        "holdable",
                        "roles"
                    ],
                    "roles": {
                        "burner_role_name": "burner",
                        "notary_role_name": "notary"
                    },
                    "mintable": {
                        "max_mint_quantity": 0
                    },
                    "divisible": {
                        "decimal": 1
                    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
                }
            }
        ],
        "encode": "JSON"
    }
}
```

getTokensByName
This method returns all the token assets with the specified name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Token.getTokensByName(token_name: string)
```

Parameters:

- `token_name: string` – The name of the token, which corresponds to the `Token_name` property of the model. The value is the class name of the token.

Returns:

- It returns an array of all of the token assets of the specified name, in JSON format.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"98e0a0a115803d25b843d630e6b23c435a192a03eb0a301fc9375f05da49a8b2",
        "payload": [
            {
                "key": "token1",
                "valueJson": {
                    "assetType": "otoken",
                    "token_id": "token1",
                    "token_name": "vtok",
                    "token_type": "fungible",
                    "behaviours": [
                        "divisible",
                        "mintable",
                        "transferable",
                        "burnable",
                        "holdable",
                        "roles"
                    ],
                    "roles": {
                        "burner_role_name": "burner",
                        "notary_role_name": "notary"
                    },
                    "mintable": {
                        "max_mint_quantity": 0
                    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "divisible": {
                        "decimal": 1
                    }
                }
            }
        ],
        "encode": "JSON"
    }
}
```

isTokenType
This method indicates whether a token asset exists with the specified ID.

```
Ctx.Token.isTokenType(token_id: string)
```

Parameters:

- `token_id: string` – The ID of the token to check.

Returns:

- On success, a promise with `true` if a token asset exists with the specified ID. On error, a rejection with an error message.

Return Value Example:

```
true
```

getByRange
This method calls the fabric `getStateByRange` method internally. Even though any asset with the given ID is returned from the ledger, this method casts the asset into the caller Asset type.

```
<Token ClassCtx.Token.getByRange(start_token_id: string, end_token_id:
string, token_class_reference?: <Instance of Token Class> )
```

Parameters:

- `startId: string` – The starting key of the range. This key is included in the range.
- `endId: string` – The end key of the range. This key is excluded from the range.
- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise with an array of `<Token Class>`. On error, a rejection with an error message.

Example:

```
@validator(yup.string(), yup.string())
public async getDigiCurrGetByRange(start_token_id: string, end_token_id:
string) {
   return await this.Ctx.Token.getByRange(start_token_id, end_token_id,
```


Copyright © 2025, Oracle and/or its affiliates.

```
DigiCurr);
}
```

Return Value Example:

```
[
    {
        "assetType": "otoken",
        "token_id": "token1",
        "token_name": "digicur",
        "token_desc": "Token 1",
        "token_type": "fungible",
        "behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "holdable",
            "roles"
        ],
        "roles": {
            "minter_role_name": "minter",
            "burner_role_name": "burner",
            "notary_role_name": "notary"
        },
        "mintable": {
            "max_mint_quantity": 20000
        },
        "divisible": {
            "decimal": 0
        },
        "token_to_currency_ratio": 1.5,
        "currency_representation": "USD"
    },
    {
        "assetType": "otoken",
        "token_id": "token2",
        "token_name": "digicur",
        "token_desc": "Token2",
        "token_type": "fungible",
        "behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "holdable",
            "roles"
        ],
        "roles": {
            "minter_role_name": "minter",
            "burner_role_name": "burner",
            "notary_role_name": "notary"
        },
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "mintable": {
            "max_mint_quantity": 20000
        },
        "divisible": {
            "decimal": 0
        },
        "token_to_currency_ratio": 1,
        "currency_representation": "EURO"
    }
]
```

Methods for Account Management

`getCallerAccountId`
This method returns the account ID of the caller.

```
Ctx.Account.getCallerAccountId(token_id: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a promise with the caller account ID. On error, a rejection with an error message.

Return Value Example:

```
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f622f
```

`generateAccountId`
This method returns an account ID, which is an alphanumeric set of characters, prefixed with `oaccount~<`*`token asset name`*`>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization and the unique token ID (`token_id`).

```
Ctx.Account.generateAccountId(token_id: string, org_id: string, user_id: string)
```

Parameters:

- `tokenId: string` – The ID of the token.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a promise with the generated account ID. On error, a rejection with an error message.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f
```

`createAccount`
This method creates an account for a specified user and token. Every user who has tokens at any point must have an account. Accounts track a user's balance, on-hold balance, and transaction history. An account ID is an alphanumeric set of characters, prefixed with `oaccount~<`*`token asset name`*`>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization. This method can be called only by the `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
this.Ctx.Account.createAccount(org_id: string, user_id: string, token_type:
string)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `token_type: string` – The type of the token, which must be `fungible`.

Returns:

- On success, the new account object in JSON format.

Return Value Example:

```
{
  "assetType": "oaccount",
  "bapAccountVersion": 0,
  "account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
  "user_id": "admin",
  "org_id": "Org1MSP",
  "token_type": "fungible",
  "token_id": "",
  "token_name": "",
  "balance": 0,
  "onhold_balance": 0
}
```

`associateTokenToAccount`
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the relevant organization.

```
async associateTokenToAccount(account_id: string, token_id: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `account_id: string` – The ID of the account.
- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON object of the updated account.

Return Value Example:

```
{
    "assetType": "oaccount",
    "bapAccountVersion": 0,
    "account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
    "user_id": "admin",
    "org_id": "Org1MSP",
    "token_type": "fungible",
    "token_id": "fungible",
    "token_name": "fiatmoneytok",
    "balance": 0,
    "onhold_balance": 0
}
```

`getAccountWithStatus`
This method returns account details for a specified account, including account status.

```
Ctx.Account.getAccountWithStatus(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise with the account details. On error, a rejection with an error message.

Return Value Example:

```
{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "account_id":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "user_id": "idcqa",
  "org_id": "appdev",
  "token_type": "fungible",
  "token_id": "t1",
  "token_name": "obptok",
  "balance": 0,
  "onhold_balance": 0
}
```


Copyright © 2025, Oracle and/or its affiliates.

getAccount
This method returns account details for a specified account.

```
Ctx.Account.getAccount(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise with the account details. On error, a rejection with an error message.

Return Value Example:

```
{
   "assetType":"oaccount",
   "bapAccountVersion": 0,

"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
   "user_id":"user1",
   "org_id":"Org1MSP",
   "token_id":"digiCurr101",
   "token_name":"digicur",
   "balance":0,
   "onhold_balance":0
}
```

history
This method returns an array of the account history details for a specified account.

```
Ctx.Account.history(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise with the array of account history details. On error, a rejection with an error message. The return value is the same as the `getAccountHistory` method.

Return Value Example:

```
[
    {

"trxId":"2gsdh17fff222467e5667be042e33ce18e804b3e065cca15de306f837e416d7c3e",
      "timeStamp":1629718288,
      "value":{
         "assetType":"oaccount",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
         "user_id":"user1",
         "org_id":"Org1MSP",
         "token_id":"digiCurr101",
         "token_name":"digicur",
         "balance":100,
         "onhold_balance":0,
         "bapAccountVersion": 1
   },
   {

"trxId":"9fd07fff222467e5667be042e33ce18e804b3e065cca15de306f837e416d7c3e",
      "timeStamp":1629718288,
      "value":{
         "assetType":"oaccount",

"account_id":"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d5
46d64f368642f622f",
         "user_id":"user1",
         "org_id":"Org1MSP",
         "token_id":"digiCurr101",
         "token_name":"digicur",
         "balance":0,
         "onhold_balance":0,
         "bapAccountVersion": 0
      }
   }
]
```

`getAccountOnHoldBalance`
This method returns the on-hold balance for a specified account.

```
Ctx.Account.getAccountOnHoldBalance(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise with a JSON object that shows the on-hold balance for the specified account. On error, a rejection with an error message.

Return Value Example:

```
{
   "holding_balance":0,
   "msg":"Total Holding Balance of Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id: Org1MSP, user_id: user1) is 0"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getAllAccounts`
This method returns a list of all accounts. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Account.getAllAccounts()
```

Parameters:

- none

Returns:

- On success, a promise with a JSON object that lists all accounts. On error, a rejection with an error message.

Return Value Example:

```
[
            {
                "key":
"oaccount~digicur~2e2ef3375ae347cbd7b4d3d7be5cece803f9c36a184aaf2b8d332c5d2dce
ad52",
                "valueJson": {
                    "assetType": "oaccount",
                    "account_id":
"oaccount~digicur~2e2ef3375ae347cbd7b4d3d7be5cece803f9c36a184aaf2b8d332c5d2dce
ad52",
                    "user_id": "admin",
                    "org_id": "Org1MSP",
                    "token_id": "digiCurr101",
                    "token_name": "digicur",
                    "bapAccountVersion": 0,
                    "balance": 0,
                    "onhold_balance": 0
                }
            },
            {
                "key":
"oaccount~digicur~30080c7e5ba94035af57fbbccbbb495e92515e4b2b3dbcd476eb1c0343e4
da65",
                "valueJson": {
                    "assetType": "oaccount",
                    "account_id":
"oaccount~digicur~30080c7e5ba94035af57fbbccbbb495e92515e4b2b3dbcd476eb1c0343e4
da65",
                    "bapAccountVersion": 0,
                    "user_id": "user1",
                    "org_id": "Org1MSP",
                    "token_id": "digiCurr101",
                    "token_name": "digicur",
                    "balance": 0,
                    "onhold_balance": 0
                }
            },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            {
                "key":
"oaccount~digicur~cbde438258cb01a82f71a9a9f8029243c40c6d836a505432120529c2b3c2
ff0c",
                "valueJson": {
                    "assetType": "oaccount",
                    "account_id":
"oaccount~digicur~cbde438258cb01a82f71a9a9f8029243c40c6d836a505432120529c2b3c2
ff0c",
                    "bapAccountVersion": 0,
                    "user_id": "user2",
                    "org_id": "Org1MSP",
                    "token_id": "digiCurr101",
                    "token_name": "digicur",
                    "balance": 0,
                    "onhold_balance": 0
                }
            },
            {
                "key":
"oaccount~digicur~ecbc3aefcc562d3049c988717940195b30297e95012b7824bbd33a57ca50
a626",
                "valueJson": {
                    "assetType": "oaccount",
                    "account_id":
"oaccount~digicur~ecbc3aefcc562d3049c988717940195b30297e95012b7824bbd33a57ca50
a626",
                    "bapAccountVersion": 0,
                    "user_id": "user3",
                    "org_id": "Org1MSP",
                    "token_id": "digiCurr101",
                    "token_name": "digicur",
                    "balance": 500,
                    "onhold_balance": 0
                }
            }
        ]
```

`getUserByAccountId`
This method returns the user details for a specified account.

```
Ctx.Account.getUserByAccountId(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise with a JSON object that includes three properties:
  - `user_id` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

  - `org_id` – The membership service provider (MSP) ID of the user in the current network organization.
  - `token_id` – The ID of the token.
- On error, a rejection with an error message.

Return Value Example:

```
{
   "token_id": "digiCurr101",
   "user_id": "user1",
   "org_id": "Org1MSP"
}
```

`getAccountBalance`
This method returns the account balance for a specified account.

```
Ctx.Account.getAccountBalance(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise message with a JSON object that includes two properties:
  - `msg` – A message showing the current balance.
  - `user_balance` – The numeric value of the current balance.
- On error, a rejection with an error message.

Return Value Example:

```
{
    "msg": "Current Balance is: 200",
    "user_balance": 200
}
```

`getAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization.

```
Ctx.Account.getAllOrgAccounts(org_id: string)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "user_id": "idcqa",
            "org_id": "appdev",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "balance": 0,
            "onhold_balance": 0
        }
    },
    {
        "key":
"oaccount~620fcf5deb5fd5a65c0b5b10fda129de0f629ccd232c5891c130e24a574df50a",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~620fcf5deb5fd5a65c0b5b10fda129de0f629ccd232c5891c130e24a574df50a",
            "user_id": "example_minter",
            "org_id": "appdev",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "balance": 0,
            "onhold_balance": 0
        }
    }
]
```

Methods for Role Management

`addRoleMember`
This method adds a role to a specified user and token.

```
Ctx.Token.addRoleMember(role: string, account_id: string, token: <Instance of
Token Class>)
```

Parameters:

- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.

Copyright © 2025, Oracle and/or its affiliates.

- `account_id: number` – The account ID to add the role to.
- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
    "msg":"Successfully added role minter to
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id :          Org1MSP, user_id : user1)"
}
```

`removeRoleMember`
This method removes a role from a specified user and token.

```
Ctx.Token.removeRoleMember(role: string, account_id: string, token: <Instance
of Token Class>)
```

Parameters:

- `role: string` – The name of the role to remove from to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `account_id: number` – The account ID to remove the role from.
- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
  "msg":"successfully removed member_id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) from role minter"
}
```

`getAccountsByRole`
This method returns a list of all accounts for a specified role and token.

```
Ctx.Role.getAccountsByRole(token_id: string, role: string)
```

Parameters:

- `token_id: string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

- `role: string` – The name of the role to search for.

Returns:

- On success, a promise with a JSON object that lists all accounts for the specified role and token. On error, a rejection with an error message.

Return Value Example:

```
{
    "accounts": [

"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",

"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f"
    ]
}
```

`getAccountsByUser`
This method returns a list of all account IDs for a specified user.

```
async getAccountsByUser(org_id: string, user_id: string)
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{"accounts":
["oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642
f622f"]}
```

`getUsersByRole`
This method returns a list of all users for a specified role and token.

```
Ctx.Role.getUsersByRole(token_id: string, role: string)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise with a JSON object that lists all users for the specified role and token. On error, a rejection with an error message.

Return Value Example:

```
{
   "users":[
      {
         "token_id":"digiCurr101",
         "user_id":"user1",
         "org_id":"Org1MSP"
      }
   ]
}
```

`isInRole`
This method indicates whether a user and token has a specified role.

```
Ctx.Token.isInRole(role: string, account_id: string, token: <Instance of
Token Class>)
```

Parameters:

- `role: string` – The name of the role to check.
- `account_id: number` – The account ID to check.
- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise with `true` if the user has the role, and `false` if the user does not have the role. On error, a rejection with an error message.

Return Value Example:

```
{"result":"true"}
```

`getOrgAccountsByRole`
This method returns information about all accounts that have a specified role in a specified organization.

```
Ctx.Role.getOrgAccountsByRole(token_id: string, role: string, org_id: string)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts with the specified role in the specified organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "accounts": [

"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",

"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850"
    ]
}
```

`getOrgUsersByRole`
This method returns information about all users that have a specified role in a specified organization.

```
Ctx.Role.getOrgUsersByRole(token_id: string, role: string, org_id: string)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all users with the specified role in the specified organization.

Return Value Example:

```
{
    "users": [
        {
            "token_id": "token",
            "user_id": "admin",
            "org_id": "Org1MSP"
        },
        {
            "token_id": "token",
            "user_id": "orgAdmin",
            "org_id": "Org1MSP"
        }
    ]
}
```

`roleCheck`
This method checks if the provided account ID is a member of any role.

```
Ctx.Token.roleCheck(account_id: string, token: <Instance of Token Class>)
```

Parameters:

- `account_id: string` – The account ID to check.

Copyright © 2025, Oracle and/or its affiliates.

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- If the account ID is part of any role, it returns `true`. Otherwise, it returns `false`.

Return Value Example:

```
{ result: true }
```

Methods for Transaction History Management

`getAccountTransactionHistory`
This method returns an array of the transaction history details for a specified account.

```
Ctx.Account.getAccountTransactionHistory(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- The return value is the same as the `getAccountTransactionHistory` method.
- On success, a promise with the array of account transaction objects:
  - `transaction_id` – The ID of the transaction.
  - `transacted_account` – The account with which the transaction took place.
  - `transaction_type` – The type of transaction.
  - `transacted_amount` – The amount of the transaction.
  - `timestamp` – The time of the transaction.
  - `balance` – The account balance at the time of the transaction.
  - `onhold_balance` – The on-hold balance at the time of the transaction.
  - `sub_transactions` – For bulk transfers only, a list of transactions that are part of a bulk transfer.
  - `holding_id` – A unique identifier returned by the `holdTokens` method.
  - `token_id` – The ID of the token.
- On error, a rejection with an error message.

Return Value Example:

```
[
   {

"transaction_id":"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e
63a9bbf1d8e0fc775",
      "transacted_amount":20,
      "timestamp":"2021-08-17T06:04:24.000Z",
      "balance":60,
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "onhold_balance":0,
      "token_id":"digiCurr101",
      "transaction_type":"BULKTRANSFER",
      "sub_transactions":[
         {

"transacted_account":"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca5
1010737900c1d85f0376152df",
            "transaction_type":"CREDIT",

"transaction_id":"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e
63a9bbf1d8e0fc775~c4ca4238a0b923820dcc509a6f75849b",
            "transacted_amount":10
         }
      ]
   },
   {

"transaction_id":"otransaction~757864d5369bd0539d044caeb3bb4898db310fd7aa740f4
5a9938771903d43da",
      "transacted_amount":50,
      "timestamp":"2021-08-17T06:02:44.000Z",
      "balance":50,
      "onhold_balance":0,
      "token_id":"digiCurr101",

"transacted_account":"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca5
1010737900c1d85f0376152df",
      "transaction_type":"CREDIT"
   }
]
```

`getAccountTransactionHistoryWithFilters`
This method returns an array of the transaction history details for a specified account. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
await this.Ctx.Account.getAccountTransactionHistoryWithFilters(account_id:
string, filters?: Filters)
```

Parameters:

- `account_id: string` – The ID of the account.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Example:

```
ochain invoke getAccountTransactionHistoryWithFilters 'token1' 'appbuilder12'
'user_minter'
```


Copyright © 2025, Oracle and/or its affiliates.

```
'{"PageSize":10,"Bookmark":"1","StartTime":"2022-01-25T17:41:42Z","EndTime":"202
2-01-25T17:59:10Z"}'

[
    {
        "transaction_id":
"otransaction~672897b5a4fa78b421c000e4d6d4f71f3d46529bfbb5b4be10bf5471dc35ce89
",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:46:04.000Z",
        "token_id": "token1",
        "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
        "transaction_type": "DEBIT",
        "balance": 90,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~467bb67a33aaffca4487f33dcd46c9844efdb5421a2e7b6aa2d53152eb2c6d85
",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:45:47.000Z",
        "token_id": "token1",
        "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
        "transaction_type": "DEBIT",
        "balance": 95,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~c6d56ce54a9bbe24597d1d10448e39316dc6f16328bf3c5b0c8ef10e1dfeb397
",
        "transacted_amount": 100,
        "timestamp": "2022-04-20T15:44:26.000Z",
        "token_id": "token1",
        "transacted_account":
"oaccount~deb5fb0906c40506f6c2d00c573b774e01a53dd91499e651d92ac4778b6add6a",
        "transaction_type": "MINT",
        "balance": 100,
        "onhold_balance": 0
    }
]
```

`getSubTransactionHistory`
This method returns an array of the transaction history details for a specified transaction.

```
await this.Ctx.Account.getSubTransactionHistory(transaction_id)
```

Parameters:

- `transaction_id: string` – The ID of the bulk transfer transaction.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
ochain invoke GetAccountSubTransactionHistory
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'

[
    {
        "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
        "transaction_type": "DEBIT",
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c81e728d9d4c2f636f067f89cc14862c",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "balance": 80,
        "onhold_balance": 0
    },
    {
        "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
        "transaction_type": "DEBIT",
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c4ca4238a0b923820dcc509a6f75849b",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "balance": 85,
        "onhold_balance": 0
    }
]
```

`getSubTransactionHistoryWithFilters`
This method returns an array of the subtransaction history details for a specified transaction.

```
await this.Ctx.Account.getSubTransactionHistoryWithFilters(transaction_id:
string, filters?: SubTransactionFilters)
```

Parameters:

- `transaction_id: string` – The ID of the bulk transfer transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
ochain invoke GetAccountSubTransactionHistoryWithFilters
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
'{"PageSize":10,"Bookmark":"1"}'

[
    {
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c81e728d9d4c2f636f067f89cc14862c",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "transacted_account":
"oaccount~16c38d804413ebabf416360d374f76c973d4e71c74adfde73cc40c7c274883b8",
        "transaction_type": "DEBIT",
        "balance": 80,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~6e0f8fe4a6430322170b9c619b04b6c9f1c8d257923f611b866bdf69d7fe6cb8
~c4ca4238a0b923820dcc509a6f75849b",
        "transacted_amount": 5,
        "timestamp": "2022-04-20T15:52:21.000Z",
        "token_id": "token1",
        "transacted_account":
"oaccount~fbf95683b21bbc91a22205819ac1e2e9c90355d536821ed3fe22b7d23915c248",
        "transaction_type": "DEBIT",
        "balance": 85,
        "onhold_balance": 0
    }
]
```

`getTransactionById`
This method returns the history of a `Transaction` asset.

```
async getTransactionById(transaction_id: string)
```

Parameters:

- `transaction_id: string` – The ID of the transaction asset.

Returns:

- On success, the transaction asset history.

Return Value Example:

```
{
    "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
    "history": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
        {
            "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
            "timeStamp": 1629180264,
            "value": {
                "assetType": "otransaction",
                "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
",
                "token_id": "digiCurr101",
                "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                "to_account_id": "",
                "transaction_type": "BULKTRANSFER",
                "amount": 20,
                "timestamp": "2021-08-17T06:04:24.000Z",
                "number_of_sub_transactions": 2,
                "holding_id": ""
            }
        }
    ],
    "sub_transactions": [
        {
            "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
            "history": [
                {
                    "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
                    "timeStamp": 1629180264,
                    "value": {
                        "assetType": "otransaction",
                        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c4ca4238a0b923820dcc509a6f75849b",
                        "token_id": "digiCurr101",
                        "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                        "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                        "transaction_type": "TRANSFER",
                        "amount": 10,
                        "timestamp": "2021-08-17T06:04:24.000Z",
                        "number_of_sub_transactions": 0,
                        "holding_id": ""
                    }
                }
            ]
        },
        {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c81e728d9d4c2f636f067f89cc14862c",
            "history": [
                {
                    "trxId":
"68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775",
                    "timeStamp": 1629180264,
                    "value": {
                        "assetType": "otransaction",
                        "transaction_id":
"otransaction~68f46c90d0d8d6b93d827e6b9e0152b4845e6e42a61965e63a9bbf1d8e0fc775
~c81e728d9d4c2f636f067f89cc14862c",
                        "token_id": "digiCurr101",
                        "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
                        "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
                        "transaction_type": "TRANSFER",
                        "amount": 10,
                        "timestamp": "2021-08-17T06:04:24.000Z",
                        "number_of_sub_transactions": 0,
                        "holding_id": ""
                    }
                }
            ]
        }
    ]
}
```

`deleteHistoricalTransactions`
This method returns an array of the transaction history details for a specified account.

```
async deleteHistoricalTransactions(time_to_expiration: Date)
```

Parameters:

- `time_to_expiration: Date` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted..

Returns:

- The return value is the same as the `getAccountTransactionHistory` method.
- On success, a promise with the array of account transaction objects:
  - `transaction_id` – The ID of the transaction.
  - `transacted_account` – The account with which the transaction took place.
  - `transaction_type` – The type of transaction.
  - `transacted_amount` – The amount of the transaction.

Copyright © 2025, Oracle and/or its affiliates.

  - `timestamp` – The time of the transaction.
  - `balance` – The account balance at the time of the transaction.
  - `onhold_balance` – The on-hold balance at the time of the transaction.
  - `sub_transactions` – For bulk transfers only, a list of transactions that are part of a bulk transfer.
  - `holding_id` – A unique identifier returned by the `holdTokens` method.
  - `token_id` – The ID of the token.
- On error, a rejection with an error message.

Return Value Example:

```
"payload": {
            "msg": "Successfuly deleted transaction older than date: Thu Aug
19 2021 11:19:36 GMT+0000 (Coordinated Universal Time).",
            "transactions": [

"otransaction~ec3366dd48b4ce2838f820f2f138648e6e55a07226713e59b411ff31b0d21058
"
            ]
        }
```

Token Behavior Management

The token lifecycle management methods are based on the the standards of the Token Taxonomy Framework. To use the token lifecycle methods, import the `Token` class from the `../lib/token` module.

```
import { Token } from '../lib/token';
```

Methods for Token Behavior Management - Mintable Behavior

`mint`
This method mints a quantity of tokens, which are then owned by the caller of the method. The caller must have an account and the minter role. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.

```
Ctx.Token.mint(quantity: number, token: <Instance of Token Class>)
```

Parameters:

- `quantity: number` – The total number of tokens to mint.
- `token: <Instance of Token Class>` – The token asset to mint.

Returns:

- On success, a promise with a success message and `toAccount` details. On error, a rejection with an error message.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "msg":"Successfully minted 1000 tokens to Account Id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin)"
}
```

`getTotalMintedTokens`
This method returns the total number of tokens minted.

```
Ctx.Token.getTotalMintedTokens(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, the quantity of minted tokens, in the number data type. On error, it returns with an error message.

Return Value Example:

```
4000
```

`getNetTokens`
This method returns the net quantity of tokens that are available in the system. The net tokens are the amount of tokens remaining after tokens are burned. In equation form: net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens.

```
Ctx.Token.getNetTokens(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, the quantity of net tokens, in the number data type. On error, it returns with an error message.

Return Value Example:

```
2000
```


Copyright © 2025, Oracle and/or its affiliates.

`getMaxMintQuantity`
This method returns the maximum mintable quantity for a token. If the `max_mint_quantity` behavior is not specified, then the default value is 0, which allows any number of tokens to be minted.

```
Ctx.Token.getMaxMintQuantity(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, the maximum mintable quantity of the token, in the number data type. On error, it returns with an error message.

Return Value Example:

```
20000
```

Methods for Token Behavior Management - Transferable Behavior

`transfer`
This method transfers tokens from the transaction caller to the `to_account_id` account. The caller of this method must have an account and the quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.

```
Ctx.Token.transfer(to_account_id: string, quantity: number, token: <Instance
of Token Class>)
```

Parameters:

- `to_account_id: string` – The account ID to receive the tokens.
- `quantity: number` – The total number of tokens to transfer.

Returns:

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
 "msg":"Successfully transferred 50 tokens from account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin) to account id:
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`bulkTransfer`
This method transfers tokens in bulk from the caller account to the accounts that are specified in the `flow` object. The caller of this method must have an account already created.

```
Ctx.Token.bulkTransfer(flow: object[], token: <Instance of Token Class>)
```

Parameters:

- `flow: object[]` – An array of JSON objects specifying the receiver details and quantity. The transfer quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. For example:

```
[{
    "to_org_id": "Org1MSP",
    "to_user_id": "user1",
    "quantity": 10
}, {
    "to_org_id": "Org1MSP",
    "to_user_id": "user2",
    "quantity": 10
}]
```

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise with a success message and account information. On error, a rejection with an error message.

Return Value Example:

```
{
    "from_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "msg": "Successfully transferred 2 tokens from Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)",
    "sub_transactions": [
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e"
        },
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df"
        }
    ]
}
```

Methods for Token Behavior Management - Holdable Behavior

Copyright © 2025, Oracle and/or its affiliates.

`hold`
This method creates a hold on behalf of the owner of the tokens with the `to_account_id` account. A notary account is specified, which is responsible to either complete or release the hold. When the hold is created, the specified token balance from the payer is put on hold. A held balance cannot be transferred until the hold is either completed or released. The caller of this method must have an account already created.

```
Ctx.Token.hold(operation_id: string, to_account_id: string,
notary_account_id: string, quantity: number, time_to_expiration: Date, token:
<Instance of Token Class>)
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_account_id: string` – The ID of the account to receive the tokens.
- `notary__account_id: string` – The ID of the notary account.
- `quantity: number` – The total number of tokens to put on hold.
- `time_to_expiration: Date` – The duration until the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12`.
- `token: <Instance of Token Class>` – The token asset to hold.

Returns:

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
 "msg": "account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (org_id : Org1MSP, user_id : user1) is successfully holding 10 tokens",
}
```

`executeHold`
This method completes a hold on tokens, transferring the specified quantity of tokens previously on hold to the receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can be called only by the `AccountOwner` ID with the `notary` role for the specified operation ID. The hold can only be completed by the notary.

```
Ctx.Token.executeHold(operation_id: string, quantity: number, token:
<Instance of Token Class>)
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity: number` – The total number of tokens to complete a hold on.

Copyright © 2025, Oracle and/or its affiliates.

- `token: <Instance of Token Class>` – The token asset to complete a hold on.

Returns:

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
 "msg": "user with accountId:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (org_id : Org1MSP, user_id : user1) has successfully executed 5
tokens(digiCurr101) from the hold with Operation Id opr_121",
}
```

`releaseHold`
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `AccountOwner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
Ctx.Token.releaseHold(operation_id: string, token: <Instance of Token Class>)
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `token: <Instance of Token Class>` – The token asset to release a hold on.

Returns:

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
  "msg": "Successfully released 5 tokens from Operation Id opr_121 to Account
Id: oaccount~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f0376152df
(org_id : Org1MSP, user_id : user1)",
}
```

`getOnHoldIds`
This method returns a list of all of the holding IDs for a specified account.

```
Ctx.Account.getOnHoldIds(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise with a JSON object that lists all holding IDs for the specified account. On error, a rejection with an error message.

Return Value Example:

```
{
   "msg":"Holding Ids are: ohold~digicur~digiCurr101~opr_121",
   "holding_ids":[
      "ohold~digicur~digiCurr101~opr_121"
   ]
}
```

`getOnHoldDetailsWithOperationId`
This method returns the on-hold transaction details for a specified operation ID and token.

```
Ctx.Hold.getOnHoldDetailsWithOperationId(token_id: string, operation_id:
string)
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a hold object that includes the following properties:
  - `holding_id` – The holding ID of the transaction.
  - `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
  - `from_account_id` – The account ID of the current owner of the on-hold tokens.
  - `to_account_id` – The account ID of the receiver.
  - `notary_account_id` – The account ID of the notary.
  - `token_id: string` – The ID of the saved token.
  - `quantity` – The amount of tokens that are on hold for the holding ID.
  - `time_to_expiration` – The duration until the hold expires.
- On error, a rejection with an error message.

Return Value Example:

```
{
    "assetType": "ohold",
    "holding_id": "ohold~digicur~digiCurr101~opr_121",
    "operation_id": "opr_121",
    "token_name": "digicur",
    "from_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "to_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "notary_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
    "token_id": "digiCurr101",
    "quantity": 10,
    "time_to_expiration": "2022-08-01T18:30:00.000Z"
}
```

`getOnHoldBalanceWithOperationId`
This method returns the on-hold balance for a specified operation ID and token. This method can be called by anyone.

```
Ctx.Hold.getOnHoldBalanceWithOperationId(token_id: string, operation_id:
string)
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a promise object with the on-hold balance for the specified operation ID and token. On error, a rejection with an error message

Return Value Example:

```
{
    "msg": "Current Holding Balance of Operation 'op1' for token 'token1' is:
10",
    "holding_balance": 10
}
```

Methods for Token Behavior Management - Burnable Behavior

`burn`
This method deactivates, or burns, tokens from the transaction caller's account. The caller of this method must have an account and the burner role. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.

```
Ctx.Token.burn(quantity: number, token: <Instance of Token Class>)
```

Parameters:

- `quantity: number` – The total number of tokens to burn.
- `token: <Instance of Token Class>` – The token asset to burn.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise with a success message. On error, a rejection with an error message.

Return Value Example:

```
{
 "msg":"Successfully burned 10 tokens from account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin)"
}
```

TypeScript Methods for Token Conversion

Blockchain App Builder automatically generates methods that you can use to convert fungible tokens that use the Token Taxonomy Framework standard.

The token conversion methods include the concept of the **exchange pool**. The exchange pool account is funded by other token accounts. When you mint tokens, you can specify that a percentage of the minted tokens are transferred to the exchange pool account.

- Token Conversion Process
- Automatically Generated Token Conversion Methods
- Token Conversion SDK Methods

Token Conversion Process

A typical flow for converting tokens follows these steps:

1. Call the `initializeExchangePoolUser` method to initialize the exchange pool user.
2. Call the `createExchangePoolAccounts` method to create exchange pool accounts. Create an exchange pool account for every type of fungible token that you want to convert from or convert to.
3. Call the `addConversionRate` method to set the conversion rate for each pair of tokens that you want to convert between.
4. Fund the exchange pool token accounts in one of the following ways:
   - Transfer tokens to the exchange pool token accounts using the standard transfer methods.
   - Call the `mintWithFundingExchangePoolToken` method when minting tokens, which can transfer a percentage of minted tokens to an exchange pool account.
5. Call the `tokenConversion` method to convert between two fungible tokens. A single user can convert tokens between two of their token accounts, or a pair of users can directly convert tokens from one account to another.
6. The exchange pool user can view the exchange pool account balances and account transactions.
   - Call the `getAccount` method to view the balances of each of the exchange pool token accounts.
   - Call the `getAccountTransactionHistory` and `getAccountTransactionHistoryWithFilters` methods to view account transactions for each of the exchange pool token accounts.

Copyright © 2025, Oracle and/or its affiliates.

Automatically Generated Token Conversion Methods

Blockchain App Builder automatically generates methods to convert between different types of fungible tokens. Controller methods must have a `@Validator(...params)` decorator to be invokable.

- Base
- Digital Assets

Base

`initializeExchangePoolUser`
This method initializes the exchange pool user, which is a one-time activity. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async initializeExchangePoolUser(org_id: string, user_id: string){
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.initializeExchangePoolUser'
, 'TOKEN');
    return await this.Ctx.TokenConvertor.initializeExchangePoolUser(org_id,
user_id);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the exchange pool user.

Return Value Example:

```
{
    "assetType": "oconversion",
    "convertor_id":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "org_id": "Org1MSP",
    "user_id": "exchangepooluser"
}
```


Copyright © 2025, Oracle and/or its affiliates.

createExchangePoolAccounts
This method creates exchange pool token accounts for a given array of token IDs. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.array().of(yup.string()))
public async createExchangePoolAccounts(token_ids: string[]){
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.initializeExchangePoolUser'
, 'TOKEN');
    return await
this.Ctx.TokenConvertor.createExchangePoolAccounts(token_ids);
}
```

Parameters:

- `token_ids: string []` – An array of token IDs.

Returns:

- On success, a list of objects that includes details of the exchange pool accounts that were created.

Return Value Example:

```
[
  {
    "account_id":
"oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034",
    "token_id": "USD",
    "status": "created"
  },
  {
    "account_id":
"oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0",
    "token_id": "INR",
    "status": "created"
  }
]
```

addConversionRate
This method adds a conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.number())
public async addConversionRate(from_token_id:string , to_token_id:string,
token_conversion_rate: number) {
   await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.addConversionRate',
'TOKEN');
   return await
this.Ctx.TokenConvertor.addConversionToken(from_token_id,to_token_id,token_con
```


Copyright © 2025, Oracle and/or its affiliates.

```
version_rate);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: number` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 10
}
```

`getConversionRate`
This method gets the current conversion rate for a pair of tokens. This method can be called by the `Token Admin` of the chaincode, any `Org Admin`, and by any token account owner.

```
@Validator(yup.string(), yup.string())
public async getConversionRate(from_token_id:string , to_token_id:string) {
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.getConversionRate',
'TOKEN');
    const conversion_rate_id = await
this.Ctx.TokenConversionRate.getConversionRateId(from_token_id, to_token_id);
    return await this.Ctx.TokenConversionRate.get(conversion_rate_id);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 10
}
```

updateConversionRate
This method updates the current conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.number())
public async updateConversionRate(from_token_id:string , to_token_id:string,
token_conversion_rate: number) {
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.updateConversionRate',
'TOKEN');
    return await
this.Ctx.TokenConvertor.updateTokenConversionRate(from_token_id,to_token_id,to
ken_conversion_rate);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: number` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the updated conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 20
}
```


Copyright © 2025, Oracle and/or its affiliates.

`mintWithFundingExchangePool`
This method mints tokens in the caller's account based on the specified token ID and quantity. A percentage of tokens from the minted quantity is then transferred to the exchange pool token account.

```
@Validator(yup.string(), yup.number(), yup.number())
public async mintWithFundingExchangePool(token_id: string, token_quantity:
number, percentage_token_to_exchange_pool: number) {
    return await
this.Ctx.TokenConvertor.mintWithFundingExchangePool(token_id, token_quantity,
percentage_token_to_exchange_pool);
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `token_quantity: number` – The total number of tokens to mint.
- `percentage_token_to_exchange_pool: number` – The percentage of minted tokens to transfer to the exchange pool token account.

Returns:

- On success, a message that indicates that minting and funding the exchange pool were successful.

Return Value Example:

```
{
    "msg": "Successfully minted 100 tokens to Account Id:
oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb
(Org-Id: Org1MSP, User-Id: admin) and Successfully transfered 20 tokens to
exchange pool Account with Account Id:
oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034
(Org-Id: Org1MSP, User-Id: exchangepooluser) "
}
```

`tokenConversion`
This method converts tokens from the caller's account to the account specified by the `to_token_id`, `to_org_id` and `to_user_id` values. This method can be called by the `Token Admin` of the chaincode and by any token account owner. An exchange pool user cannot call this method.

```
@Validator(yup.string(),yup.string(),yup.string(),yup.string(),yup.number())
public async tokenConversion(from_token_id:string, to_token_id:string,
to_org_id:string, to_user_id:string, token_quantity:number){
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.tokenConversion', 'TOKEN');
    return await
this.Ctx.TokenConvertor.tokenConversion(from_token_id,to_token_id,to_org_id,to
_user_id,token_quantity);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `to_org_id: string` – The membership service provider (MSP) ID of the user in the current organization to receive the tokens.
- `to_user_id: string` – The user name or email ID of the user to receive the tokens.
- `token_quantity: number` – The total number of tokens to transfer.

Returns:

- On success, a message that indicates the token conversion was successful.

Return Value Example:

```
{
    "msg": "Succesfully converted 5 of tokens with tokenId: [USD] from
AccountId:
oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb
(Org-Id: Org1MSP, User-Id: admin) to 100 of tokens with tokenId: [INR] to
AccountId:
oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e
(Org-Id: Org1MSP, User-Id: user) as per the conversion rate of 20"
}
```

`getConversionHistory`
This method returns the token conversion history for a specified token account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async getConversionHistory(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getConversionHistory",
"TOKEN", { account_id });
    return await this.Ctx.Account.getTokenConversionHistory(account_id,
org_id, user_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object with conversion history details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
  {
    "transaction_id":
"otransaction~34edd19e03ec8bbbc77bc3372081410a824a5c10f9aa522b3a6390d7e8cb11cf
",
    "from_account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
    "to_account_id":
"oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e",
    "transacted_amount": 5,
    "converted_amount": 100,
    "conversion_rate": "20",
    "from_token_id": "USD",
    "to_token_id": "INR",
    "balance": 75,
    "onhold_balance": 0,
    "timestamp": "2022-11-30T11:03:20.000Z",
    "transaction_type": "TOKEN_CONVERSION_DEBIT"
  }
]
```

`getConversionRateHistory`
This method returns the token conversion rate history for a pair of tokens. This method can be called by the `Token Admin` of the chaincode, any `Org Admin`, and by any token account owner.

```
@Validator(yup.string(), yup.string())
public async getConversionRateHistory(from_token_id:string ,
to_token_id:string) {
    const conversion_rate_id = await
this.Ctx.TokenConversionRate.getConversionRateId(from_token_id,to_token_id);
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.getConversionRateHistory',
'TOKEN');
    return await this.Ctx.TokenConversionRate.history(conversion_rate_id);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from, for the purpose of calculating the conversion rate.
- `to_token_id: string` – The ID of the token to convert to, for the purpose of calculating the conversion rate.

Returns:

- On success, a JSON object with conversion rate history details.

Return Value Example:

```
 [
  {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "trxId":
"0b1ba7bc2620e1438b6580365e5c0ab852247ccfa5a3eb2157d3baca02c0e521",
    "timeStamp": "2022-11-30T10:23:38.000Z",
    "value": {
      "assetType": "oconversionRate",
      "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
      "from_token_id": "USD",
      "to_token_id": "INR",
      "conversion_rate": 20
    }
  },
  {
    "trxId":
"36fc40ddb3d8308ee7e156af700da131d78d941fe390fc57985b7589e7035d5c",
    "timeStamp": "2022-11-30T10:13:18.000Z",
    "value": {
      "assetType": "oconversionRate",
      "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
      "from_token_id": "USD",
      "to_token_id": "INR",
      "conversion_rate": 10
    }
  }
]
```

`getExchangePoolUser`
This method returns the `org_id` and `user_id` values for the exchange pool user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator()
public async getExchangePoolUser() {
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.getExchangePoolUser',
'TOKEN');
    return await this.Ctx.TokenConvertor.getExchangePoolUser();
}
```

Parameters:

- none

Returns:

- On success, a message with information about the exchange pool user.

Return Value Example:

```
{
   "assetType": "oconversion",
   "convertor_id":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
   "org_id": "Org1MSP",
   "user_id": "exchangepooluser"
}
```

Digital Assets

`initializeExchangePoolUser`
This method initializes the exchange pool user, which is a one-time activity. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async initializeExchangePoolUser(org_id: string, user_id: string){
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.initializeExchangePoolUser'
, 'TOKEN');
    return await this.Ctx.TokenConvertor.initializeExchangePoolUser(org_id,
user_id);
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the exchange pool user.

Return Value Example:

```
{
    "assetType": "oconversion",
    "convertor_id":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "org_id": "Org1MSP",
    "user_id": "exchangepooluser"
}
```

`createExchangePoolAccounts`
This method creates exchange pool token accounts for a given array of token IDs. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.array().of(yup.string()))
public async createExchangePoolAccounts(token_ids: string[]){
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.initializeExchangePoolUser'
, 'TOKEN');
    return await
this.Ctx.TokenConvertor.createExchangePoolAccounts(token_ids);
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `token_ids: string []` – An array of token IDs.

Returns:

- On success, a list of objects that includes details of the exchange pool accounts that were created.

Return Value Example:

```
[
  {
    "account_id":
"oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034",
    "token_id": "USD",
    "status": "created"
  },
  {
    "account_id":
"oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0",
    "token_id": "INR",
    "status": "created"
  }
]
```

`addConversionRate`
This method adds a conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.number())
public async addConversionRate(from_token_id:string , to_token_id:string,
token_conversion_rate: number) {
   await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.addConversionRate',
'TOKEN');
   return await
this.Ctx.TokenConvertor.addConversionToken(from_token_id,to_token_id,token_con
version_rate);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: number` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the conversion rate object.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 10
}
```

`getConversionRate`
This method gets the current conversion rate for a pair of tokens. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, and by any token account owner.

```
@Validator(yup.string(), yup.string())
public async getConversionRate(from_token_id:string , to_token_id:string) {
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.getConversionRate',
'TOKEN');
    const conversion_rate_id = await
this.Ctx.TokenConversionRate.getConversionRateId(from_token_id, to_token_id);
    return await this.Ctx.TokenConversionRate.get(conversion_rate_id);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 10
}
```


Copyright © 2025, Oracle and/or its affiliates.

updateConversionRate
This method updates the current conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.number())
public async updateConversionRate(from_token_id:string , to_token_id:string,
token_conversion_rate: number) {
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.updateConversionRate',
'TOKEN');
    return await
this.Ctx.TokenConvertor.updateTokenConversionRate(from_token_id,to_token_id,to
ken_conversion_rate);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: number` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the updated conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 20
}
```

mintWithFundingExchangePool
This method mints tokens in the caller's account based on the specified token ID and quantity. A percentage of tokens from the minted quantity is then transferred to the exchange pool token account.

```
@Validator(yup.string(), yup.number(), yup.number())
public async mintWithFundingExchangePool(token_id: string, token_quantity:
number, percentage_token_to_exchange_pool: number) {
    return await
this.Ctx.TokenConvertor.mintWithFundingExchangePool(token_id, token_quantity,
percentage_token_to_exchange_pool);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id: string` – The ID of the token to mint.
- `token_quantity: number` – The total number of tokens to mint.
- `percentage_token_to_exchange_pool: number` – The percentage of minted tokens to transfer to the exchange pool token account.

Returns:

- On success, a message that indicates that minting and funding the exchange pool were successful.

Return Value Example:

```
{
    "msg": "Successfully minted 100 tokens to Account Id:
oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb
(Org-Id: Org1MSP, User-Id: admin) and Successfully transfered 20 tokens to
exchange pool Account with Account Id:
oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034
(Org-Id: Org1MSP, User-Id: exchangepooluser) "
}
```

`tokenConversion`
This method converts tokens from the caller's account to the account specified by the `to_token_id`, `to_org_id` and `to_user_id` values. This method can be called by the `Token Admin` of the chaincode and by any token account owner. An exchange pool user cannot call this method.

```
@Validator(yup.string(),yup.string(),yup.string(),yup.string(),yup.number())
public async tokenConversion(from_token_id:string, to_token_id:string,
to_org_id:string, to_user_id:string, token_quantity:number){
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.tokenConversion', 'TOKEN');
    return await
this.Ctx.TokenConvertor.tokenConversion(from_token_id,to_token_id,to_org_id,to
_user_id,token_quantity);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `to_org_id: string` – The membership service provider (MSP) ID of the user in the current organization to receive the tokens.
- `to_user_id: string` – The user name or email ID of the user to receive the tokens.
- `token_quantity: number` – The total number of tokens to transfer.

Returns:

- On success, a message that indicates the token conversion was successful.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Succesfully converted 5 of tokens with tokenId: [USD] from
AccountId:
oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb
(Org-Id: Org1MSP, User-Id: admin) to 100 of tokens with tokenId: [INR] to
AccountId:
oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e
(Org-Id: Org1MSP, User-Id: user) as per the conversion rate of 20"
}
```

`getConversionHistory`
This method returns the token conversion history for a specified token account. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the token account owner.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async getConversionHistory(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getConversionHistory",
"TOKEN", { account_id });
    return await this.Ctx.Account.getTokenConversionHistory(account_id,
org_id, user_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object with conversion history details.

Return Value Example:

```
[
  {
    "transaction_id":
"otransaction~34edd19e03ec8bbbc77bc3372081410a824a5c10f9aa522b3a6390d7e8cb11cf
",
    "from_account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
    "to_account_id":
"oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e",
    "transacted_amount": 5,
    "converted_amount": 100,
    "conversion_rate": "20",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "from_token_id": "USD",
    "to_token_id": "INR",
    "balance": 75,
    "onhold_balance": 0,
    "timestamp": "2022-11-30T11:03:20.000Z",
    "transaction_type": "TOKEN_CONVERSION_DEBIT"
  }
]
```

getConversionRateHistory
This method returns the token conversion rate history for a pair of tokens. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, and by any token account owner.

```
@Validator(yup.string(), yup.string())
public async getConversionRateHistory(from_token_id:string ,
to_token_id:string) {
    const conversion_rate_id = await
this.Ctx.TokenConversionRate.getConversionRateId(from_token_id,to_token_id);
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.getConversionRateHistory',
'TOKEN');
    return await this.Ctx.TokenConversionRate.history(conversion_rate_id);
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from, for the purpose of calculating the conversion rate.
- `to_token_id: string` – The ID of the token to convert to, for the purpose of calculating the conversion rate.

Returns:

- On success, a JSON object with conversion rate history details.

Return Value Example:

```
 [
  {
    "trxId":
"0b1ba7bc2620e1438b6580365e5c0ab852247ccfa5a3eb2157d3baca02c0e521",
    "timeStamp": "2022-11-30T10:23:38.000Z",
    "value": {
      "assetType": "oconversionRate",
      "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
      "from_token_id": "USD",
      "to_token_id": "INR",
      "conversion_rate": 20
    }
  },
```


Copyright © 2025, Oracle and/or its affiliates.

```
  {
    "trxId":
"36fc40ddb3d8308ee7e156af700da131d78d941fe390fc57985b7589e7035d5c",
    "timeStamp": "2022-11-30T10:13:18.000Z",
    "value": {
      "assetType": "oconversionRate",
      "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
      "from_token_id": "USD",
      "to_token_id": "INR",
      "conversion_rate": 10
    }
  }
]
```

`getExchangePoolUser`
This method returns the `org_id` and `user_id` values for the exchange pool user. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode.

```
@Validator()
public async getExchangePoolUser() {
    await
this.Ctx.Auth.checkAuthorization('TOKEN_CONVERSION.getExchangePoolUser',
'TOKEN');
    return await this.Ctx.TokenConvertor.getExchangePoolUser();
}
```

Parameters:

- none

Returns:

- On success, a message with information about the exchange pool user.

Return Value Example:

```
{
   "assetType": "oconversion",
   "convertor_id":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
   "org_id": "Org1MSP",
   "user_id": "exchangepooluser"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Token Conversion SDK Methods

`initializeExchangePoolUser`
This method initializes the exchange pool user, which is a one-time activity. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.initializeExchangePoolUser(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the exchange pool user.

Return Value Example:

```
{
    "assetType": "oconversion",
    "convertor_id":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "org_id": "Org1MSP",
    "user_id": "exchangepooluser"
}
```

`createExchangePoolAccounts`
This method creates exchange pool token accounts for a given array of token IDs. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.createExchangePoolAccounts(token_ids: string[])
```

Parameters:

- `token_ids: string []` – An array of token IDs.

Returns:

- On success, a list of objects that includes details of the exchange pool accounts that were created.

Return Value Example:

```
[
  {
    "account_id":
"oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034",
    "token_id": "USD",
    "status": "created"
  },
```


Copyright © 2025, Oracle and/or its affiliates.

```
  {
    "account_id":
"oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0",
    "token_id": "INR",
    "status": "created"
  }
]
```

addConversionToken
This method adds tokens with a new conversion rate for a specified token. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.addConversionToken(fromTokenId: string, toTokenId: string,
tokenConversionRate: number)
```

Parameters:

- `fromTokenId: string` – The ID of the token to convert from.
- `toTokenId: string` – The ID of the token to convert to.
- `tokenConversionRate: number` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 10
}
```

get
This method gets the current conversion rate for a pair of tokens. This method can be called by the `Token Admin` of the chaincode and by any token account owner.

```
Ctx.TokenConversionRate.get(id: string)
```

Parameters:

- `id: string` – The ID of the token conversion rate object.

Returns:

- On success, a JSON representation of the conversion rate object.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 10
}
```

`updateTokenConversionRate`
This method updates the current conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.updateTokenConversionRate(fromTokenId: string, toTokenId:
string, tokenConversionRate: number)
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: number` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the updated conversion rate object.

Return Value Example:

```
{
  "assetType": "oconversionRate",
  "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
  "from_token_id": "USD",
  "to_token_id": "INR",
  "conversion_rate": 20
}
```

`mintWithFundingExchangePool`
This method mints tokens in the caller's account based on the specified token ID and quantity. A percentage of tokens from the minted quantity is then transferred to the exchange pool token account.

```
Ctx.TokenConvertor.mintWithFundingExchangePool(tokenId: string,
tokenQuantity: number, percentageTokenToExchangePool: number)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id: string` – The ID of the token to mint.
- `token_quantity: number` – The total number of tokens to mint.
- `percentage_token_to_exchange_pool: number` – The percentage of minted tokens to transfer to the exchange pool token account.

Returns:

- On success, a message that indicates that minting and funding the exchange pool were successful.

Return Value Example:

```
{
    "msg": "Successfully minted 100 tokens to Account Id:
oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb
(Org-Id: Org1MSP, User-Id: admin) and Successfully transfered 20 tokens to
exchange pool Account with Account Id:
oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034
(Org-Id: Org1MSP, User-Id: exchangepooluser) "
}
```

`tokenConversion`
This method converts tokens from the caller's account to the account specified by the `to_token_id`, `to_org_id`, and `to_user_id` values. This method can be called by the `Token Admin` of the chaincode and by any token account owner. An exchange pool user cannot call this method.

```
Ctx.TokenConvertor.tokenConversion(fromTokenId: string, toTokenId: string,
toOrgId: string, toUserId: string, tokenQuantity: number)
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `to_org_id: string` – The membership service provider (MSP) ID of the user in the current organization to receive the tokens.
- `to_user_id: string` – The user name or email ID of the user to receive the tokens.

Returns:

- On success, a message that indicates the token conversion was successful.

Return Value Example:

```
{
    "msg": "Succesfully converted 5 of tokens with tokenId: [USD] from
AccountId:
oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb
(Org-Id: Org1MSP, User-Id: admin) to 100 of tokens with tokenId: [INR] to
AccountId:
oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e
```


Copyright © 2025, Oracle and/or its affiliates.

```
(Org-Id: Org1MSP, User-Id: user) as per the conversion rate of 20"
}
```

getTokenConversionHistory
This method returns the token conversion history for a specified token account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, and by the token account owner.

```
Ctx.Account.getTokenConversionHistory(account_id: string, org_id: string,
user_id: string)
```

Parameters:

- `account_id: string` – The ID of the fungible token account.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object with conversion history details.

Return Value Example:

```
[
  {
    "transaction_id":
"otransaction~34edd19e03ec8bbbc77bc3372081410a824a5c10f9aa522b3a6390d7e8cb11cf
",
    "from_account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
    "to_account_id":
"oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e",
    "transacted_amount": 5,
    "converted_amount": 100,
    "conversion_rate": "20",
    "from_token_id": "USD",
    "to_token_id": "INR",
    "balance": 75,
    "onhold_balance": 0,
    "timestamp": "2022-11-30T11:03:20.000Z",
    "transaction_type": "TOKEN_CONVERSION_DEBIT"
  }
]
```

history
This method returns the token conversion rate history for a pair of tokens. This method can be called by the `Token Admin` of the chaincode, any `Org Admin`, and by any token account owner.

```
Ctx.TokenConversionRate.history(conversion_rate_id: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `conversion_rate_id: string` – The ID of the conversion rate object.

Returns:

- On success, a JSON object with conversion rate history details.

Return Value Example:

```
 [
  {
    "trxId":
"0b1ba7bc2620e1438b6580365e5c0ab852247ccfa5a3eb2157d3baca02c0e521",
    "timeStamp": "2022-11-30T10:23:38.000Z",
    "value": {
      "assetType": "oconversionRate",
      "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
      "from_token_id": "USD",
      "to_token_id": "INR",
      "conversion_rate": 20
    }
  },
  {
    "trxId":
"36fc40ddb3d8308ee7e156af700da131d78d941fe390fc57985b7589e7035d5c",
    "timeStamp": "2022-11-30T10:13:18.000Z",
    "value": {
      "assetType": "oconversionRate",
      "conversion_rate_id":
"oconversionRate~91c7eeb0614e7a50b1d5ecad559fcbc80b94034648bf405c9491dacf8d578
73b",
      "from_token_id": "USD",
      "to_token_id": "INR",
      "conversion_rate": 10
    }
  }
]
```

`getExchangePoolUser`
This method returns the `OrgId` and `UserId` values for the exchange pool user. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.getExchangePoolUser()
```

Parameters:

- none

Returns:

- On success, a message with information about the exchange pool user.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
   "assetType": "oconversion",
   "convertor_id":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
   "org_id": "Org1MSP",
   "user_id": "exchangepooluser"
}
```

TypeScript Methods for Token Account Status

Blockchain App Builder automatically generates methods that you can use to manage account status for fungible tokens that use the Token Taxonomy Framework standard.

You can use the following methods to put token user accounts in the active, suspended, or deleted states.

When an account is suspended, the account user cannot complete any write operations, which include minting, burning, transferring, and holding tokens. Additionally, other users cannot transfer tokens to or hold tokens in a suspended account. A suspended account can still complete read operations.

An account with a non-zero token balance cannot be deleted. You must transfer or burn all tokens in an account before you can delete the account. After an account is in the deleted state, the account state cannot be changed back to active or suspended.

- Automatically Generated Account Status Methods
- Account Status SDK Methods

Automatically Generated Account Status Methods

Blockchain App Builder automatically generates methods to manage token account status. Controller methods must have a `@Validator(...params)` decorator to be invokable.

- Base
- Digital Assets

Base

`getAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner. This method also supports data migration for existing chaincode that is upgraded to a newer version.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async getAccountStatus(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.get", "TOKEN",
{ account_id });
    try {
      return await this.Ctx.AccountStatus.getAccountStatus(account_id);
    } catch (err) {
      return await this.Ctx.AccountStatus.getDefaultAccountStatus(account_id);
    }
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the token account status.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`getAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountStatusHistory(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Account.getAccount(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.history", "TOKEN",
{ account_id });
    const status_id = await
this.Ctx.AccountStatus.generateAccountStatusId(account_id);
    let account_status_history: any;
    try {
      account_status_history = await
this.Ctx.AccountStatus.history(status_id);
    } catch (err) {
      return [];
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    return account_status_history;
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the account status history.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```


Copyright © 2025, Oracle and/or its affiliates.

activateAccount
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. Deleted accounts cannot be activated.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async activateAccount(token_id: string, org_id: string, user_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.activateAccount",
"TOKEN", { org_id });
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    return await this.Ctx.Account.activateAccount(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

suspendAccount
This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async suspendAccount(token_id: string, org_id: string, user_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.suspendAccount",
"TOKEN", { org_id });
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Account.suspendAccount(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

`deleteAccount`
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance and the on-hold balance must be zero.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async deleteAccount(token_id: string, org_id: string, user_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.deleteAccount",
"TOKEN", { org_id });
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    return await this.Ctx.Account.deleteAccount(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Digital Assets

`getAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the token account owner. This method also supports data migration for existing chaincode that is upgraded to a newer version.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async getAccountStatus(token_id: string, org_id: string, user_id:
string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.get", "TOKEN",
{ account_id });
    try {
      return await this.Ctx.AccountStatus.getAccountStatus(account_id);
    } catch (err) {
      return await this.Ctx.AccountStatus.getDefaultAccountStatus(account_id);
    }
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the token account status.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`getAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the token account owner.

```
  @Validator(yup.string(), yup.string(), yup.string())
  public async getAccountStatusHistory(token_id: string, org_id: string,
user_id: string) {
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    await this.Ctx.Account.getAccount(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.history", "TOKEN",
{ account_id });
    const status_id = await
this.Ctx.AccountStatus.generateAccountStatusId(account_id);
    let account_status_history: any;
    try {
      account_status_history = await
this.Ctx.AccountStatus.history(status_id);
    } catch (err) {
      return [];
    }
    return account_status_history;
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the account status history.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

`activateAccount`
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. Deleted accounts cannot be activated.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async activateAccount(token_id: string, org_id: string, user_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.activateAccount",
"TOKEN", { org_id });
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    return await this.Ctx.Account.activateAccount(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

`suspendAccount`
This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async suspendAccount(token_id: string, org_id: string, user_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.suspendAccount",
"TOKEN", { org_id });
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    return await this.Ctx.Account.suspendAccount(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```


Copyright © 2025, Oracle and/or its affiliates.

deleteAccount
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance and the on-hold balance must be zero.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async deleteAccount(token_id: string, org_id: string, user_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.deleteAccount",
"TOKEN", { org_id });
    const account_id = await this.Ctx.Account.generateAccountId(token_id,
org_id, user_id);
    return await this.Ctx.Account.deleteAccount(account_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Account Status SDK Methods

getAccountStatus
This method gets the current status of the token account.

```
Ctx.AccountStatus.getAccountStatus(account_id: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`saveAccountStatus`
This method saves the status object (if a status object is not present) for the token account, and sets the status to the specified value.

```
Ctx.AccountStatus.saveAccountStatus(account_id: string, status: AccountStatus)
```

Parameters:

- `account_id: string` – The ID of the token account.
- `status: AccountStatus` – The status to set for the specified account.
  `AccountStatus` is an `enum` type which must be `active`, `suspended`, or `deleted`.

Returns:

- On success, a JSON representation of the account status object.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`getAccountStatusHistory`
This method gets the history of the account status.

```
Ctx.AccountStatus.history(status_id: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `status_id: string` – The ID of the account status object.

Returns:

- On success, a JSON representation of the account status history.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

`activateAccount`
This method activates a token account.

```
Ctx.Account.activateAccount(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

`suspendAccount`
This method suspends a token account.

```
Ctx.Account.suspendAccount(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

`deleteAccount`
This method deletes a token account.

```
Ctx.Account.deleteAccount(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Scaffolded Go Project for Token Taxonomy Framework

Blockchain App Builder takes the input from your token specification file and generates a fully-functional scaffolded chaincode project.

The project automatically generates token lifecycle classes and functions, including CRUD and non-CRUD methods. Validation of arguments, marshalling/unmarshalling, and transparent persistence capability are all supported automatically.

For information on the scaffolded project and methods that are not directly related to tokens, see Scaffolded Go Chaincode Project.

Reference:

- Model
- Controller
  - Automatically Generated Token Methods
  - Custom Methods
- Token SDK Methods

Model

Transparent Persistence Capability, or simplified ORM, is captured in the `OchainModel` class.

```
package src
type Digicur struct {
    AssetType              string                `json:"AssetType"
final:"otoken"`
    Token_id               string                `json:"Token_id" id:"true"
mandatory:"true" validate:"regexp=^[A-Za-z0-9][A-Za-z0-9_-]*$,max=16"`
    Token_name             string                `json:"Token_name"
final:"digicur"`
    Token_desc             string                `json:"Token_desc"
validate:"max=256"`
    Token_type             string                `json:"Token_type"
final:"fungible" validate:"regexp=^fungible$"`
    Behavior               []string              `json:"Behavior"
final:"[\"divisible\",\"mintable\",\"transferable\",\"burnable\",\"holdable\",
\"roles\"]"`
    Roles                  map[string]interface{} `json:"Roles"
final:"{\"minter_role_name\":\"minter\",\"burner_role_name\":\"burner\",\"nota
ry_role_name\":\"notary\"}"`
    Mintable               map[string]interface{} `json:"Mintable"
```


Copyright © 2025, Oracle and/or its affiliates.

```
final:"{\"Max_mint_quantity\":20000}"`
    Divisible               map[string]interface{} `json:"Divisible"
final:"{\"Decimal\":1}"`
    Token_to_currency_ratio int
`json:"Token_to_currency_ratio" validate:"int"`
    Currency_representation string
`json:"Currency_representation" validate:"string"`
    Metadata                interface{}            `json:"Metadata,omitempty"`
}
```

Controller

There is only one main controller.

```
type Controller struct {
    Ctx trxcontext.TrxContext
}
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable. The other methods are hidden.

You can use the token SDK methods to write custom methods for your business application.

If you use more than one token SDK method in a custom method, do not use methods that will affect the same key-value pairs in the state database.

Instead, use the `BulkTransferTokens` method to transfer to multiple accounts from the caller's account, as shown in the following code snippet.

```
BulkTransferTokens(token_id string, flow: []map[string]interface{})
```

> **Note**
>
> If you use more than one token SDK method in a custom method that might affect the same key-value pairs in the state database, enable the MVCC optimization for token chaincodes. For more information, see MVCC Optimization.

Automatically Generated Token Methods

Blockchain App Builder automatically generates methods to support tokens and token life cycles. You can use these methods to initialize tokens, manage roles and accounts, and complete other token lifecycle tasks without any additional coding. Controller methods must be public to be invokable. Public method names begin with an upper case character. Method names that begin with a lower case character are private.

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior

Copyright © 2025, Oracle and/or its affiliates.

– Transferable Behavior
– Holdable Behavior
– Burnable Behavior

Methods for Access Control Management

- Base
- Digital Assets

Base

`AddTokenAdmin`
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenAdmin(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.AddAdmin", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.AddTokenAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{
    "msg": "Successfully added Token Admin (Org_Id: Org1MSP, User_Id: user1)"
}
```

`RemoveTokenAdmin`
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) RemoveTokenAdmin(org_id string, user_id string)
(interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Admin.RemoveAdmin", "TOKEN")
```


Copyright © 2025, Oracle and/or its affiliates.

```
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Admin.RemoveAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfuly removed Admin (Org_Id Org1MSP User_Id user1)"}
```

IsTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. A `Token Admin` or `Org Admin` can call this function on any other user in the blockchain network. Other users can call this method only on their own accounts.

```
func (t *Controller) IsTokenAdmin(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.IsTokenAdmin",
"TOKEN", map[string]string{"org_id": org_id, "user_id": user_id})
      if err != nil || !auth {
            return false, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Auth.IsUserTokenAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

Return Value Example:

```
{"result":false}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetAllTokenAdmins`
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by the `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) GetAllTokenAdmins() (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Admin.GetAllAdmins", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Admin.GetAllAdmins()
}
```

Parameters:

- none

Returns:

- On success, a JSON list of admins that includes `OrgId` and `UserId` objects.

Return Value Example:

```
{"admins":[{"OrgId":"Org1MSP","UserId":"admin"},
{"OrgId":"Org1MSP","UserId":"user2"}]}
```

`AddOrgAdmin`
This method adds a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) AddOrgAdmin(org_id string, user_id string) (interface{},
error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.AddOrgAdmin",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.AddOrgAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

RemoveOrgAdmin
This method removes a user as an `Org Admin` of an organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) RemoveOrgAdmin(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.RemoveOrgAdmin",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.RemoveOrgAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

GetOrgAdmins
This method returns a list of all users who are an `Org Admin` of an organization. This method can be called only by a `Token Admin` of the chaincode or by any `Org Admin`.

```
func (t *Controller) GetOrgAdmins() (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.GetOrgAdmins",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.GetAllOrgAdmins()
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- none

Returns:

- On success, a JSON list that includes `OrgId` and `UserId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin1"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin2"
        }
    ]
}
```

Digital Assets

`AddTokenAdmin`
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenAdmin(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.AddAdmin", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.AddTokenAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Successfully added Token Admin (Org_Id: Org1MSP, User_Id: user1)"
}
```

RemoveTokenAdmin
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) RemoveTokenAdmin(org_id string, user_id string)
(interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Admin.RemoveAdmin", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Admin.RemoveAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfuly removed Admin (Org_Id Org1MSP User_Id user1)"}
```

IsTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. This function can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) IsTokenAdmin(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.IsTokenAdmin",
"TOKEN", map[string]string{"org_id": org_id, "user_id": user_id})
      if err != nil || !auth {
            return false, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Auth.IsUserTokenAdmin(org_id, user_id)
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

Return Value Example:

```
{"result":false}
```

`GetAllTokenAdmins`
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` or `Token Auditor`.

```
func (t *Controller) GetAllTokenAdmins() (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Admin.GetAllAdmins", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Admin.GetAllAdmins()
}
```

Parameters:

- none

Returns:

- On success, a JSON list of admins that includes `OrgId` and `UserId` objects.

Return Value Example:

```
{"admins":[{"OrgId":"Org1MSP","UserId":"admin"},
{"OrgId":"Org1MSP","UserId":"user2"}]}
```

`AddOrgAdmin`
This method adds a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) AddOrgAdmin(org_id string, user_id string) (interface{},
error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.AddOrgAdmin",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.AddOrgAdmin(org_id, user_id)
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`RemoveOrgAdmin`
This method removes a user as an `Org Admin` of an organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) RemoveOrgAdmin(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.RemoveOrgAdmin",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.RemoveOrgAdmin(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetOrgAdmins
This method returns a list of all users who are an `Org Admin` of an organization. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetOrgAdmins() (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Admin.GetOrgAdmins",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Admin.GetAllOrgAdmins()
}
```

Parameters:

- none

Returns:

- On success, a JSON list that includes `OrgId` and `UserId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin1"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin2"
        }
    ]
}
```

AddTokenAuditor
This method adds a user as a `Token Auditor` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenAuditor(org_id string, user_id string)
(interface{}, error)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a message that includes details of the user who was added as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"f0888dd52f39dfa669275cc8f35d0b47b37b8407d384493d16970fcbb377f937",
        "payload": {
            "msg": "Successfully added Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 268
    }
}
```

`RemoveTokenAuditor`
This method removes a user as a `Token Auditor` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) RemoveTokenAuditor(org_id string, user_id string)
(interface{}, error)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"2f01f48eceaf2dff175f98b96a5bdd22c949f48fc5683ce86d6141cc4892aee3",
        "payload": {
            "msg": "Successfully removed Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 270
    }
}
```

GetTokenAuditors
This method returns all `Token Auditors` of the chaincode. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode.

```
func (t *Controller) GetTokenAuditors() (interface{}, error)
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"18eaa81b04d43db64f28287bab1cf6609e2a1d8ff84852ff73849ddb9a9dfba1",
        "payload": {
            "auditors": [
                {
                    "OrgId": "CB",
                    "UserId": "auditor_user_cb"
                }
            ]
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 269
    }
}
```

AddOrgAuditor
This method adds a user as a `Org Auditor` of the chaincode. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) AddOrgAuditor(org_id string, user_id string)
(interface{}, error)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Org Auditor` of the chaincode.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"3d5ec46003c68c6208d43c82894bd6da5c0b763339cc5212e09b71d39d0d80e2",
        "payload": {
            "msg": "Successfully added Org Auditor (Org_Id: CB, User_Id: cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 271
    }
}
```

`RemoveOrgAuditor`
This method removes a user as a `Org Auditor` of the chaincode. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) RemoveOrgAuditor(org_id string, user_id string)
(interface{}, error)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"713a120641afbc4dfaeac73b82c9fd51df6fcfd7d4d9a82553d3c487bf11f530",
        "payload": {
            "msg": "Successfully removed Org Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 273
    }
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetOrgAuditors
This method returns all `Org Auditors` of the chaincode. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetOrgAuditors() (interface{}, error)
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"07656bf434616d7a3d7fe4fb81dc80c8cc608991648adfd9f6f2f2b9f6ddf468",
        "payload": {
            "auditors": [
                {
                    "OrgId": "CB",
                    "UserId": "cb"
                },
                {
                    "OrgId": "CB",
                    "UserId": "issuer_user_cb"
                }
            ]
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 272
    }
}
```

Methods for Token Configuration Management

- Base
- Digital Assets

Base

Init
This method is called when the chaincode is deployed. Every `Token Admin` is identified by the `user_id` and `org_id` information in the mandatory `adminList` parameter. The `user_id` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `org_id` is the membership service provider (MSP) ID of the user in the current network organization.

Copyright © 2025, Oracle and/or its affiliates.

Any `Token Admin` user can add and remove other `Token Admin` users by calling the `AddTokenAdmin` and `RemoveTokenAdmin` methods.

```
func (t *Controller) Init(adminList []admin.TokenAdminAsset) (interface{},
error) {
    list, err := t.Ctx.Admin.InitAdmin(adminList)
    if err != nil {
        return nil, fmt.Errorf("initializing admin list failed %s",
err.Error())
    }
    return list, nil
}
```

Parameters:

- `adminList array` – An array of `{user_id, org_id}` information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

Parameter example, macOS and Linux CLI:

```
'[{"user_id":"userid", "org_id":"OrgMSPId"}]'
```

Parameter example, Microsoft Windows CLI:

```
"[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"
```

Parameter example, Oracle Blockchain Platform console:

```
["[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"]
```

`Initialize<Token Name>Token`
This method creates a token and initializes the token properties. The asset and its properties are saved in the state database. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) InitializeDigicurToken(asset Digicur) (interface{},
error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.Save", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.Save(&asset)
}
```

Parameters:

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

Copyright © 2025, Oracle and/or its affiliates.

You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

```
"{\"token_id\":\"USD\",\"token_desc\":\"token_desc
value\",\"Currency_Name\":\"Currency_Name value\"}"
```

Returns:

- On success, a JSON representation of the token asset that was created.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"roles"],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

`Update<Token Name>Token`
This method updates token properties. After a token asset is created, only the `token_desc` property and custom properties can be updated. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) UpdateDigicurToken(asset Digicur) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.Update", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.Update(&asset)
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

  ```
  "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
  value\",\"Currency_Name\":\"Currency_Name value\"}"
  ```

Returns:

- On success, an updated JSON representation of the token asset.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "Digital Currency equiv of dollar",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"roles"],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

`GetTokenDecimals`
This method returns the number of decimal places that were configured for a fractional token. If the `divisible` behavior was not specified for the token, then the default value is 0. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) GetTokenDecimals(token_id string) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTokenDecimals",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    tokenDecimal, err := t.Ctx.Token.GetTokenDecimals(token_id)
```


Copyright © 2025, Oracle and/or its affiliates.

```
    if err != nil {
        return nil, fmt.Errorf("Error in GetTokenDecimals %s", err.Error())
    }
    response := make(map[string]interface{})
    response["msg"] = fmt.Sprintf("Token Id: %s has %d decimal places.",
token_id, tokenDecimal)
    return response, nil
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON string showing the number of token decimal places.

Return Value Example:

```
{"msg":"Token Id: digiCurr101 has 1 decimal places."}
```

`GetTokenById`
This method returns a token object if it is present in the state database. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) GetTokenById(token_id string) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.Get", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    tokenAsset, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return tokenAsset.Interface(), err
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token asset.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "Digital Currency equiv of dollar",
    "Token_type": "fungible",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "Behavior": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

GetTokenHistory
This method returns the token history for a specified token ID. Any user can call this method.

```
func (t *Controller) GetTokenHistory(token_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTokenHistory",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Token.History(token_id)
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token history.

Return Value Example:

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2023-09-01T16:46:33Z",
        "TxId":
"12333b8a4f63aa9b3a34072efcbd7df546c6d1e7d82a7a9596e899383656d6f7",
        "Value": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "Currency_name1": "",
                "Divisible": {
                    "Decimal": 2
                },
                "Mintable": {
                    "Max_mint_quantity": 1000
                },
                "Roles": {
                    "minter_role_name": "minter"
                },
                "Token_desc": "updated description",
                "Token_id": "token",
                "Token_name": "fiatmoneytok",
                "Token_to_currency_ratio": 0,
                "Token_type": "fungible",
                "Token_unit": "fractional"
            }
        },
        {
            "IsDelete": "false",
            "Timestamp": "2023-09-01T16:04:25Z",
            "TxId":
"99702e2dad7554a5ee4716a0d01d3e394cbce39bea8bade265d8911f30ebad0b",
            "Value": {
                "AssetType": "otoken",
                "Behavior": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "Currency_name1": "",
                "Divisible": {
                    "Decimal": 2
                },
                "Mintable": {
                    "Max_mint_quantity": 1000
                },
                "Roles": {
                    "minter_role_name": "minter"
                },
                "Token_desc": "",
                "Token_id": "token",
                "Token_name": "fiatmoneytok",
                "Token_to_currency_ratio": 0,
                "Token_type": "fungible",
                "Token_unit": "fractional"
            }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
]
```

GetAllTokens
This method returns all tokens that are stored in the state database. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) GetAllTokens() (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetAllTokens", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Token.GetAllTokens()
}
```

Parameters:

- none

Returns:

- On success, a JSON object that represents all token assets.

Return Value Example:

```
"payload": [
    {
        "key": "t1",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "Currency_name value",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "Token_desc value",
            "Token_id": "t1",
            "Token_name": "obptok",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "Token_to_currency_ratio": 2,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    }
]
```

GetTokensByName
This method returns all token objects with a specified name. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetTokensByName(token_name string) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTokensByName",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Token.GetTokensByName(token_name)
}
```

Parameters:

- `token_name string` – The name of the tokens to retrieve. The name corresponds to the `Token_name` property in the specification file. The value is the class name of the token.

Returns:

- On success, a JSON object of all token assets that match the name.

Return Value Example:

```
"payload": [
    {
        "key": "t1",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "Currency_name value",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "Token_desc value",
            "Token_id": "t1",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 999,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    },
    {
        "key": "obp2",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "",
            "Token_id": "obp2",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 0,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    }
]
```

Digital Assets

`Init`

This method is called when the chaincode is deployed. Every `Token Admin` is identified by the `user_id` and `org_id` information in the mandatory `adminList` parameter. The `user_id` is the user name or email ID of the instance owner or the user who is logged in to the instance. The

Copyright © 2025, Oracle and/or its affiliates.

`org_id` is the membership service provider (MSP) ID of the user in the current network organization.

Any `Token Admin` user can add and remove other `Token Admin` users by calling the `AddTokenAdmin` and `RemoveTokenAdmin` methods.

```
func (t *Controller) Init(adminList []admin.TokenAdminAsset) (interface{},
error) {
    list, err := t.Ctx.Admin.InitAdmin(adminList)
    if err != nil {
        return nil, fmt.Errorf("initializing admin list failed %s",
err.Error())
    }
    return list, nil
}
```

Parameters:

- `adminList array` – An array of {`user_id, org_id`} information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

Parameter example, macOS and Linux CLI:

```
'[{"user_id":"userid", "org_id":"OrgMSPId"}]'
```

Parameter example, Microsoft Windows CLI:

```
"[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"
```

Parameter example, Oracle Blockchain Platform console:

```
["[{\"user_id\":\"userid\", \"org_id\":\"OrgMSPId\"}]"]
```

`Initialize<Token Name>Token`
This method creates a token and initializes the token properties. The asset and its properties are saved in the state database. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) InitializeDigicurToken(asset Digicur) (interface{},
error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.Save", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.Save(&asset)
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

  ```
  "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
  value\",\"Currency_Name\":\"Currency_Name value\"}"
  ```

Returns:

- On success, a JSON representation of the token asset that was created.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"roles"],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

`Update<Token Name>Token`
This method updates token properties. After a token asset is created, only the `token_desc` property and custom properties can be updated. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) UpdateDigicurToken(asset Digicur) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.Update", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.Update(&asset)
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `asset <Token Class>` – The token asset is passed as the parameter to this method. The properties of the token asset can vary, and are described in the token specification file. Do not include parameters that are marked as read-only in the specification file.

  You specify the `asset` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: Use the GUI to pass individual parameters that correspond to the token class fields.

  CLI / Postman: Pass a serialized JSON string that includes the token specification fields, as shown in the following example.

  ```
  "{\"token_id\":\"USD\",\"token_desc\":\"token_desc
  value\",\"Currency_Name\":\"Currency_Name value\"}"
  ```

Returns:

- On success, an updated JSON representation of the token asset.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "Digital Currency equiv of dollar",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"roles"],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

`GetTokenDecimals`

This method returns the number of decimal places that were configured for a fractional token. If the `divisible` behavior was not specified for the token, then the default value is 0. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetTokenDecimals(token_id string) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTokenDecimals",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    tokenDecimal, err := t.Ctx.Token.GetTokenDecimals(token_id)
    if err != nil {
        return nil, fmt.Errorf("Error in GetTokenDecimals %s", err.Error())
    }
    response := make(map[string]interface{})
    response["msg"] = fmt.Sprintf("Token Id: %s has %d decimal places.",
token_id, tokenDecimal)
    return response, nil
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON string showing the number of token decimal places.

Return Value Example:

```
{"msg":"Token Id: digiCurr101 has 1 decimal places."}
```

`GetTokenById`
This method returns a token object if it is present in the state database. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetTokenById(token_id string) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.Get", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    tokenAsset, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return tokenAsset.Interface(), err
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token asset.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "Digital Currency equiv of dollar",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "Token_type": "fungible",
    "Behavior": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

`GetTokenHistory`
This method returns the token history for a specified token ID. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetTokenHistory(token_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTokenHistory",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Token.History(token_id)
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object that represents the token history.

Return Value Example:

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2023-09-01T16:46:33Z",
        "TxId":
"12333b8a4f63aa9b3a34072efcbd7df546c6d1e7d82a7a9596e899383656d6f7",
        "Value": {
            "AssetType": "otoken",
            "Behavior": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "Currency_name1": "",
                "Divisible": {
                    "Decimal": 2
                },
                "Mintable": {
                    "Max_mint_quantity": 1000
                },
                "Roles": {
                    "minter_role_name": "minter"
                },
                "Token_desc": "updated description",
                "Token_id": "token",
                "Token_name": "fiatmoneytok",
                "Token_to_currency_ratio": 0,
                "Token_type": "fungible",
                "Token_unit": "fractional"
            }
        },
        {
            "IsDelete": "false",
            "Timestamp": "2023-09-01T16:04:25Z",
            "TxId":
"99702e2dad7554a5ee4716a0d01d3e394cbce39bea8bade265d8911f30ebad0b",
            "Value": {
                "AssetType": "otoken",
                "Behavior": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "Currency_name1": "",
                "Divisible": {
                    "Decimal": 2
                },
                "Mintable": {
                    "Max_mint_quantity": 1000
                },
                "Roles": {
                    "minter_role_name": "minter"
                },
                "Token_desc": "",
                "Token_id": "token",
                "Token_name": "fiatmoneytok",
                "Token_to_currency_ratio": 0,
                "Token_type": "fungible",
                "Token_unit": "fractional"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    }
]
```

`GetAllTokens`
This method returns all tokens that are stored in the state database. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetAllTokens() (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetAllTokens", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Token.GetAllTokens()
}
```

Parameters:

- none

Returns:

- On success, a JSON object that represents all token assets.

Return Value Example:

```
"payload": [
    {
        "key": "t1",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "Currency_name value",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "Token_desc value",
            "Token_id": "t1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "Token_name": "obptok",
            "Token_to_currency_ratio": 2,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    }
]
```

GetTokensByName
This method returns all token objects with a specified name. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetTokensByName(token_name string) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTokensByName",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Token.GetTokensByName(token_name)
}
```

Parameters:

- `token_name string` – The name of the tokens to retrieve. The name corresponds to the `Token_name` property in the specification file. The value is the class name of the token.

Returns:

- On success, a JSON object of all token assets that match the name.

Return Value Example:

```
"payload": [
    {
        "key": "t1",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "Currency_name value",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
```


Copyright © 2025, Oracle and/or its affiliates.

```
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "Token_desc value",
            "Token_id": "t1",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 999,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    },
    {
        "key": "obp2",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "",
            "Token_id": "obp2",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 0,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    }
]
```


Methods for Account Management

- Base

Copyright © 2025, Oracle and/or its affiliates.

- Digital Assets

Base

`CreateAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track balances, on-hold balances, and transation history. An account ID is an alphanumeric set of characters, prefixed with `oaccount~<token asset name>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization.

```
unc (t *Controller) CreateAccount(org_id string, user_id string, token_type
string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.CreateAccount",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.CreateAccount(org_id, user_id, token_type)
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `token_type: string` – The type of the token, which must be `fungible`.

Returns:

- On success, a JSON object of the account that was created. The `BapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
   "AssetType":"oaccount",

"AccountId":"oaccount~a73085a385bc96c4a45aa2dff032e7dede82c0664dee5f396b7c5854
eeafd4bd",
   "BapAccountVersion": 0,
   "UserId":"user1",
   "OrgId":"Org1MSP",
   "AccountType":"fungible",
   "TokenId":"",
   "TokenName":"",
   "Balance":0,
   "BalanceOnHold":0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
   "OnHoldBurnBalance":0
}
```

AssociateTokenToAccount
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the relevant organization.

```
func (t *Controller) AssociateTokenToAccount(account_id string, token_id
string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.AssociateToken",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.AssociateToken(account_id, token_id)
}
```

Parameters:

- `account_id string` – The ID of the account.
- `token_id string` – The ID of the token.

Returns:

- On success, a JSON object of the updated account. The `BapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
"AssetType":"oaccount",
"AccountId":"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bd
bebf48eb",
"BapAccountVersion": 0,
"UserId":"admin",
"OrgId":"Org1MSP",
"AccountType":"fungible",
"TokenId":"token1",
"TokenName":"loyaltok",
"Balance":0,
"BalanceOnHold":0,
"OnHoldBurnBalance":0
}
```

GetAccount
This method returns account details for a specified user and token. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
func (t *Controller) GetAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAccount",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetAccountWithStatus(account_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `AccountId` – The ID of the user account.
- `UserId` – The user name or email ID of the user.
- `OrgId` – The membership service provider (MSP) ID of the user in the current organization.
- `TokenId` – The ID of the token.
- `Balance` – The current balance of the account.
- `BalanceOnHold` – The current on-hold balance of the account.
- `OnHoldBurnBalance` – The current on-hold balance awaiting burning approval.
- `BapAccountVersion` – An account object parameter for internal use.
- `Status` – The current status of the user account.

Return Value Example:

```
{
  "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "AssetType": "oaccount",
  "Balance": 95,
  "BalanceOnHold": 0,
  "OnHoldBurnBalance": 0,
  "BapAccountVersion": 8,
  "OrgId": "appdev",
  "Status": "active",
  "TokenId": "obp1",
  "TokenName": "obptok",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "TokenType": "fungible",
  "UserId": "idcqa"
}
```

GetAccountHistory
This method returns account history details for a specified user and token. This method can be called only by a `Token Admin` of the chaincode or the `AccountOwner` of the account.

```
func (t *Controller) GetAccountHistory(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.History", "TOKEN",
map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.History(account_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, an array of JSON account objects that includes the following properties:
- `TxId` – The transaction ID of the transaction as returned by the ledger.
- `Timestamp` – The time of the transaction.
- `IsDelete` – A Boolean value that indicates whether the record is deleted.
- `Value` – A JSON string of the account object. The `BapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
[
  {
      "IsDelete": "false",
      "Timestamp": "2023-08-28T19:31:15Z",
      "TxId":
"adde470a63860ec1013bd5c5987e8a506a48942a91b0f39fc8e561374042bd27",
      "Value": {
          "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "AssetType": "oaccount",
            "Balance": 100,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 1,
            "OrgId": "Org1MSP",
            "TokenId": "t1",
            "TokenName": "obptok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-08-28T19:30:23Z",
        "TxId":
"8fbeda2ba60ba175091faae5ae369247775f2cba45c4d6d1ead6f0b05be84743",
        "Value": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "t1",
            "TokenName": "obptok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-08-28T19:29:54Z",
        "TxId":
"19bb296ae71709e91b097ba5d9ebd7f7522095880382fbf5913334a46a6026aa",
        "Value": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "",
            "TokenName": "",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

GetAccountOnHoldBalance
This method returns the current on-hold balance for a specified account and token. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
func (t *Controller) GetAccountOnHoldBalance(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountOnHoldBalance", "TOKEN",
map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      response, err := t.Ctx.Account.GetAccountOnHoldBalance(account_id)
      return response, err
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current on-hold balance.

Return Value Example:

```
{
    "holding_balance": 0,
    "msg": "Total Holding Balance of Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id: Org1MSP, user_id: user1) is 0"
}
```

GetAllAccounts
This method returns a list of all accounts. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAllAccounts() (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAllAccounts",
"TOKEN")
    if err != nil && !auth {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Account.GetAllAccounts()
}
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 100,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 1,
            "OrgId": "appdev",
            "TokenId": "t1",
            "TokenName": "obptok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    }
]
```

`GetUserByAccountId`
This method returns user details (`org_id` and `user_id`) for a specified account. This method can be called by any user of the chaincode.

```
func (t *Controller) GetUserByAccountId(account_id string) (interface{},
error) {
    return t.Ctx.Account.GetUserByAccountById(account_id)
}
```

Parameters:

- `account_id string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`org_id`, `token_id`, and `user_id`).

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"org_id":"Org1MSP","token_id":"digiCurr101","user_id":"user1"}
```

GetAccountBalance
This method returns the current balance for a specified account and token. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
func (t *Controller) GetAccountBalance(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAccountBalance",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      response, err := t.Ctx.Account.GetAccountBalance(account_id)
      return response, err
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current account balance.

Return Value Example:

```
{"msg":"Current Balance of
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f is 0","user_balance":0}
```

GetAllOrgAccounts
This method returns a list of all token accounts that belong to a specified organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) GetAllOrgAccounts(org_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAllOrgAccounts",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetAllOrgAccounts(org_id)
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "appdev",
            "TokenId": "token",
            "TokenName": "fiatmoneytok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "key":
"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850",
        "valueJson": {
            "AccountId":
"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "appdev",
            "TokenId": "token",
            "TokenName": "fiatmoneytok",
            "TokenType": "fungible",
            "UserId": "example_minter"
        }
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

Digital Assets

`CreateAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track balances, on-hold balances, and transation history. An account ID is an alphanumeric set of characters, prefixed with `oaccount~<token asset name>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization.

```
func (t *Controller) CreateAccount(org_id string, user_id string, token_type
string, daily_limits ...account.AccountDailyLimits) (interface{}, error) {
auth, err := t.Ctx.Auth.CheckAuthorization("Account.CreateAccount", "TOKEN",
map[string]string{"org_id": org_id})
if err != nil && !auth {
return nil, fmt.Errorf("error in authorizing the caller %s", err.Error())
}
return t.Ctx.Account.CreateAccount(org_id, user_id, token_type,
daily_limits...)
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `token_type: string` – The type of the token, which must be `fungible`.
- `daily_limits: JSON` – An object specifying the maximum amount of tokens that can be used in transactions daily (`max_daily_amount`) and the maximum number of transactions that can be completed daily (`max_daily_transactions`).

  You specify the `daily_limits` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "max_daily_amount": 1000, "max_daily_transactions": 100 }`

  CLI / Postman: `"{\"max_daily_amount\":1000,\"max_daily_transactions\":100}"`

Returns:

- On success, a JSON object of the account that was created. The `BapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
   "AssetType":"oaccount",

"AccountId":"oaccount~a73085a385bc96c4a45aa2dff032e7dede82c0664dee5f396b7c5854
eeafd4bd",
   "BapAccountVersion": 0,
   "UserId":"user1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
   "OrgId":"Org1MSP",
   "AccountType":"fungible",
   "TokenId":"",
   "TokenName":"",
   "Balance":0,
   "BalanceOnHold":0,
   "OnHoldBurnBalance":0,
   "max_daily_amount": 1000,
   "max_daily_transactions": 100
}
```

`AssociateTokenToAccount`
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the relevant organization.

```
func (t *Controller) AssociateTokenToAccount(account_id string, token_id
string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.AssociateToken",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.AssociateToken(account_id, token_id)
}
```

Parameters:

- `account_id string` – The ID of the account.
- `token_id string` – The ID of the token.

Returns:

- On success, a JSON object of the updated account. The `BapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
"AssetType":"oaccount",
"AccountId":"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bd
bebf48eb",
"BapAccountVersion": 0,
"UserId":"admin",
"OrgId":"Org1MSP",
"AccountType":"fungible",
"TokenId":"token1",
"TokenName":"loyaltok",
"Balance":0,
"BalanceOnHold":0,
"OnHoldBurnBalance":0
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetAccount`
This method returns account details for a specified user and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
func (t *Controller) GetAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAccount",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetAccountWithStatus(account_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `AccountId` – The ID of the user account.
- `UserId` – The user name or email ID of the user.
- `OrgId` – The membership service provider (MSP) ID of the user in the current organization.
- `TokenId` – The ID of the token.
- `Balance` – The current balance of the account.
- `BalanceOnHold` – The current on-hold balance of the account.
- `OnHoldBurnBalance` – The current on-hold balance awaiting burn approval.
- `BapAccountVersion` – An account object parameter for internal use.
- `Status` – The current status of the user account.

Return Value Example:

```
{
  "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "AssetType": "oaccount",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "Balance": 95,
  "BalanceOnHold": 0,
  "OnHoldBurnBalance": 0,
  "BapAccountVersion": 8,
  "OrgId": "appdev",
  "Status": "active",
  "TokenId": "obp1",
  "TokenName": "obptok",
  "TokenType": "fungible",
  "UserId": "idcqa"
}
```

GetAccountHistory

This method returns account history details for a specified user and token. This method can be called only by a `Token Admin` of the chaincode or the `AccountOwner` of the account.

```
func (t *Controller) GetAccountHistory(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.History", "TOKEN",
map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.History(account_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, an array of JSON account objects that includes the following properties:
- `TxId` – The transaction ID of the transaction as returned by the ledger.
- `Timestamp` – The time of the transaction.
- `IsDelete` – A Boolean value that indicates whether the record is deleted.
- `Value` – A JSON string of the account object. The `BapAccountVersion` parameter is defined in the account object for internal use.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
[
  {
      "IsDelete": "false",
      "Timestamp": "2023-08-28T19:31:15Z",
      "TxId":
"adde470a63860ec1013bd5c5987e8a506a48942a91b0f39fc8e561374042bd27",
      "Value": {
          "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
          "AssetType": "oaccount",
          "Balance": 100,
          "BalanceOnHold": 0,
          "OnHoldBurnBalance": 0,
          "BapAccountVersion": 1,
          "OrgId": "Org1MSP",
          "TokenId": "t1",
          "TokenName": "obptok",
          "TokenType": "fungible",
          "UserId": "idcqa"
      }
  },
  {
      "IsDelete": "false",
      "Timestamp": "2023-08-28T19:30:23Z",
      "TxId":
"8fbeda2ba60ba175091faae5ae369247775f2cba45c4d6d1ead6f0b05be84743",
      "Value": {
          "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
          "AssetType": "oaccount",
          "Balance": 0,
          "BalanceOnHold": 0,
          "OnHoldBurnBalance": 0,
          "BapAccountVersion": 0,
          "OrgId": "Org1MSP",
          "TokenId": "t1",
          "TokenName": "obptok",
          "TokenType": "fungible",
          "UserId": "idcqa"
      }
  },
  {
      "IsDelete": "false",
      "Timestamp": "2023-08-28T19:29:54Z",
      "TxId":
"19bb296ae71709e91b097ba5d9ebd7f7522095880382fbf5913334a46a6026aa",
      "Value": {
          "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
          "AssetType": "oaccount",
          "Balance": 0,
          "BalanceOnHold": 0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "",
            "TokenName": "",
            "TokenType": "fungible",
            "UserId": "idcqa"
      }
  }
]
```

`GetAccountOnHoldBalance`
This method returns the current on-hold balance for a specified account and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the specified account owner.

```
func (t *Controller) GetAccountOnHoldBalance(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountOnHoldBalance", "TOKEN",
map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      response, err := t.Ctx.Account.GetAccountOnHoldBalance(account_id)
      return response, err
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the current on-hold balance.

Return Value Example:

```
{
    "holding_balance": 0,
    "msg": "Total Holding Balance of Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
```


Copyright © 2025, Oracle and/or its affiliates.

```
22f (org_id: Org1MSP, user_id: user1) is 0"
}
```

GetAllAccounts
This method returns a list of all accounts. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAllAccounts() (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAllAccounts",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.Account.GetAllAccounts()
}
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 100,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 1,
            "OrgId": "appdev",
            "TokenId": "t1",
            "TokenName": "obptok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

GetUserByAccountId
This method returns user details (`org_id` and `user_id`) for a specified account. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
func (t *Controller) GetUserByAccountId(account_id string) (interface{},
error) {
    return t.Ctx.Account.GetUserByAccountById(account_id)
}
```

Parameters:

- `account_id string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`org_id`, `token_id`, and `user_id`).

Return Value Example:

```
{"org_id":"Org1MSP","token_id":"digiCurr101","user_id":"user1"}
```

GetAccountBalance
This method returns the current balance for a specified account and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the specified account owner.

```
func (t *Controller) GetAccountBalance(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAccountBalance",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      response, err := t.Ctx.Account.GetAccountBalance(account_id)
      return response, err
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON representation of the current account balance.

Return Value Example:

```
{"msg":"Current Balance of
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f is 0","user_balance":0}
```

`GetAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
func (t *Controller) GetAllOrgAccounts(org_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAllOrgAccounts",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetAllOrgAccounts(org_id)
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "appdev",
            "TokenId": "token",
            "TokenName": "fiatmoneytok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "key":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850",
        "valueJson": {
            "AccountId":
"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "appdev",
            "TokenId": "token",
            "TokenName": "fiatmoneytok",
            "TokenType": "fungible",
            "UserId": "example_minter"
        }
    }
]
```

Methods for Role Management

- Base
- Digital Assets

Base

`AddRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also has the specified role.

```
func (t *Controller) AddRole(token_id string, user_role string, org_id
string, user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.AddRoleMember",
"TOKEN", map[string]string{"org_id": org_id, "token_id": token_id, "role":
user_role})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
```


Copyright © 2025, Oracle and/or its affiliates.

```
      return t.Ctx.Token.AddRoleMember(user_role, account_id,
tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
 {"msg":"Successfully added role minter to
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"}
```

`RemoveRole`
This method removes a role from a specified user and token. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization who also has the specified role.

```
func (t *Controller) RemoveRole(token_id string, user_role string, org_id
string, user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.RemoveRoleMember",
"TOKEN", map[string]string{"org_id": org_id, "token_id": token_id, "role":
user_role})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Token.RemoveRoleMember(user_role, account_id,
tokenAssetValue.Interface())
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
{"msg":"successfully removed member_id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) from role minter"}
```

`GetAccountsByRole`
This method returns a list of all account IDs for a specified role and token. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetAccountsByRole(token_id string, user_role string)
(interface{}, error) {
    auth, err:= t.Ctx.Auth.CheckAuthorization("Role.GetAccountsByRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Role.GetAccountsByRole(token_id, user_role)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
    "accounts": [

"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4"
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAccountsByUser
This method returns a list of all account IDs for a specified organization ID and user ID. This method can be called only by a `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the `Account Owner` specified in the parameters.

```
func (t *Controller) GetAccountsByUser(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAccountsByUser",
"TOKEN", map[string]string{"org_id": org_id, "user_id": user_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetAccountsByUser(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
    "accounts": [

"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4"
    ]
}
```

GetUsersByRole
This method returns a list of all users for a specified role and token. This method can be called only by a `Token Admin` of the chaincode or by the `Account Owner` specified in the parameters.

```
func (t *Controller) GetUsersByRole(token_id string, user_role string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Role.GetUsersByRole",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Role.GetUsersByRole(token_id, user_role)
}
```

Parameters:

- `token_id string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

- `user_role string` – The name of the role to search for.

Returns:

- On success, a JSON array of the user objects (`org_id` and `user_id`).

Return Value Example:

```
{"Users":[{"org_id":"Org1MSP","token_id":"digiCurr101","user_id":"user1"}]}
```

`IsInRole`
This method returns a Boolean value to indicate if a user and token has a specified role. This method can be called only by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
func (t *Controller) IsInRole(token_id string, org_id string, user_id string,
user_role string) (interface{}, error) {
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.IsInRole", "TOKEN",
map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      result, err := t.Ctx.Token.IsInRole(user_role, account_id,
tokenAssetValue.Interface())
      if err != nil {
            return nil, fmt.Errorf("error in IsInRole %s", err.Error())
      }
      response := make(map[string]interface{})
      response["result"] = result
      return response, nil
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.
- `user_role string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"result":false}
```

GetOrgAccountsByRole
This method returns information about all accounts that have a specified role in a specified organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) GetOrgAccountsByRole(token_id string, user_role string,
org_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Role.GetOrgAccountsByRole",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Role.GetOrgAccountsByRole(token_id, user_role, org_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts with the specified role in the specified organization.

Return Value Example:

```
{
    "accounts": [

"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",

"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850"
    ]
}
```

GetOrgUsersByRole
This method returns information about all users that have a specified role in a specified organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
func (t *Controller) GetOrgUsersByRole(token_id string, user_role string,
org_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Role.GetOrgUsersByRole",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
```


Copyright © 2025, Oracle and/or its affiliates.

```
err.Error())
      }
      return t.Ctx.Role.GetOrgUsersByRole(token_id, user_role, org_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all users with the specified role in the specified organization.

Return Value Example:

```
{
    "Users": [
        {
            "org_id": "Org1MSP",
            "token_id": "token",
            "user_id": "admin"
        },
        {
            "org_id": "Org1MSP",
            "token_id": "token",
            "user_id": "orgAdmin"
        }
    ]
}
```

Digital Assets

`AddRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also has the specified role.

```
func (t *Controller) AddRole(token_id string, user_role string, org_id
string, user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.AddRoleMember",
"TOKEN", map[string]string{"org_id": org_id, "token_id": token_id, "role":
user_role})
```


Copyright © 2025, Oracle and/or its affiliates.

```
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Token.AddRoleMember(user_role, account_id,
tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
 {"msg":"Successfully added role minter to
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"}
```

`RemoveRole`
This method removes a role from a specified user and token. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization who also has the specified role.

```
func (t *Controller) RemoveRole(token_id string, user_role string, org_id
string, user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.RemoveRoleMember",
"TOKEN", map[string]string{"org_id": org_id, "token_id": token_id, "role":
user_role})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
```


Copyright © 2025, Oracle and/or its affiliates.

```
      return t.Ctx.Token.RemoveRoleMember(user_role, account_id,
tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
{"msg":"successfully removed member_id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) from role minter"}
```

`GetAccountsByRole`
This method returns a list of all account IDs for a specified role and token. This method can be called only by a `Token Admin` or `Token Auditor`.

```
func (t *Controller) GetAccountsByRole(token_id string, user_role string)
(interface{}, error) {
    auth, err:= t.Ctx.Auth.CheckAuthorization("Role.GetAccountsByRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Role.GetAccountsByRole(token_id, user_role)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "accounts": [

"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4"
    ]
}
```

GetAccountsByUser
This method returns a list of all account IDs for a specified organization ID and user ID. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the `Account Owner` specified in the parameters.

```
func (t *Controller) GetAccountsByUser(org_id string, user_id string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetAccountsByUser",
"TOKEN", map[string]string{"org_id": org_id, "user_id": user_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetAccountsByUser(org_id, user_id)
}
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
    "accounts": [

"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4"
    ]
}
```

GetUsersByRole
This method returns a list of all users for a specified role and token. This method can be called only by a `Token Admin` or `Token Auditor`.

```
func (t *Controller) GetUsersByRole(token_id string, user_role string)
(interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Role.GetUsersByRole",
"TOKEN")
```


Copyright © 2025, Oracle and/or its affiliates.

```
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Role.GetUsersByRole(token_id, user_role)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `user_role string` – The name of the role to search for.

Returns:

- On success, a JSON array of the user objects (`org_id` and `user_id`).

Return Value Example:

```
{"Users":[{"org_id":"Org1MSP","token_id":"digiCurr101","user_id":"user1"}]}
```

IsInRole
This method returns a Boolean value to indicate if a user and token has a specified role. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the specified account owner.

```
func (t *Controller) IsInRole(token_id string, org_id string, user_id string,
user_role string) (interface{}, error) {
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Token.IsInRole", "TOKEN",
map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      result, err := t.Ctx.Token.IsInRole(user_role, account_id,
tokenAssetValue.Interface())
      if err != nil {
            return nil, fmt.Errorf("error in IsInRole %s", err.Error())
      }
      response := make(map[string]interface{})
      response["result"] = result
      return response, nil
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.
- `user_role string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Return Value Example:

```
{"result":false}
```

`GetOrgAccountsByRole`
This method returns information about all accounts that have a specified role in a specified organization. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetOrgAccountsByRole(token_id string, user_role string,
org_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Role.GetOrgAccountsByRole",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Role.GetOrgAccountsByRole(token_id, user_role, org_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts with the specified role in the specified organization.

Return Value Example:

```
{
    "accounts": [

"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",

"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850"
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetOrgUsersByRole`
This method returns information about all users that have a specified role in a specified organization. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
func (t *Controller) GetOrgUsersByRole(token_id string, user_role string,
org_id string) (interface{}, error) {
      auth, err := t.Ctx.Auth.CheckAuthorization("Role.GetOrgUsersByRole",
"TOKEN", map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Role.GetOrgUsersByRole(token_id, user_role, org_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all users with the specified role in the specified organization.

Return Value Example:

```
{
    "Users": [
        {
            "org_id": "Org1MSP",
            "token_id": "token",
            "user_id": "admin"
        },
        {
            "org_id": "Org1MSP",
            "token_id": "token",
            "user_id": "orgAdmin"
        }
    ]
}
```

Methods for Transaction History Management

- Base
- Digital Assets

Copyright © 2025, Oracle and/or its affiliates.

Base

`GetAccountTransactionHistory`
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account.

```
func (t *Controller) GetAccountTransactionHistory(token_id string, org_id
string, user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountTransactionHistory",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }

      transactionArray, err :=
t.Ctx.Account.GetAccountTransactionHistory(account_id, org_id, user_id)
      return transactionArray, err
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, an array of JSON account transaction objects that includes the following properties:
- `balance` – The account balance.
- `holding_id` – The ID of a holding account.
- `onhold_balance` – The on-hold balance.
- `timestamp` – The time of the transaction.
- `token_id` – The ID of the token.
- `transacted_account` – The account with which the transaction took place.
- `transacted_amount` – The amount of the transaction.
- `transaction_id` – The ID of the transaction.
- `transaction_type` – The type of transaction.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[{
    "balance": 199,
    "onhold_balance": 0,
    "timestamp": "2021-08-16T17:42:32.905+05:30",
    "token_id": "digiCurr101",
    "transacted_account":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "transacted_amount": 1,
    "transaction_id":
"otransaction~c8a9fa001aba6e0d8391b034655889df47eb5103713840b999a4ab41f5e57b38
",
    "transaction_type": "DEBIT"
}, {
    "balance": 200,
    "onhold_balance": 0,
    "timestamp": "2021-08-16T17:41:59.262+05:30",
    "token_id": "digiCurr101",
    "transacted_account":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "transacted_amount": 100,
    "transaction_id":
"otransaction~65a0bf8ae8108baa7495fbab91c205651c055e9f480f6808753287173026aa69
",
    "transaction_type": "MINT"
}]
```

`GetAccountTransactionHistoryWithFilters`
This method returns an array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `AccountOwner` of the account. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAccountTransactionHistoryWithFilters(token_id string,
org_id string, user_id string, filters ...account.AccountHistoryFilters)
(interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountTransactionHistoryWithFilters
", "TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }

      // sample format of filter: []string{"3", "",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"2022-01-16T15:16:36+00:00", "2022-01-17T15:16:36+00:00"}
      transactionArray, err :=
t.Ctx.Account.GetAccountTransactionHistoryWithFilters(account_id, org_id,
user_id, filters...)
      return transactionArray, err
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Example:

```
ochain invoke GetAccountTransactionHistoryWithFilters 'token1' 'appbuilder12'
'user_minter'
'{"PageSize":10,"Bookmark":"1","StartTime":"2022-01-25T17:41:42Z","EndTime":"202
2-01-25T17:59:10Z"}'
```

```
[
  {
    "transaction_id":
"otransaction~3f9c306b0ef6994885939c1a6eb5f063b06617ecb932d4a043f323ba53d55f9f
",
    "transacted_amount": 200,
    "timestamp": "2022-02-15T18:27:13.000Z",
    "token_id": "token1",
    "transacted_account":
"oaccount~obptok~26e046c8ba8b98da2cdabb78113d67200581ea3d4eea5aa324
1abd3598e05d05",
    "transaction_type": "DEBIT",
    "balance": 9200,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~f1d37c3abd5c85c0a399f246d8eb68257c49ab4fe4cdfd3501908583c51c421e
",
    "transacted_amount": 200,
    "timestamp": "2022-02-15T18:27:02.000Z",
    "token_id": "token1",
    "transaction_type": "BULKTRANSFER",
    "number_of_sub_transactions": 2,
    "balance": 9600,
    "onhold_balance": 0
  },
```


Copyright © 2025, Oracle and/or its affiliates.

```
  {
    "transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
",
    "transacted_amount": 200,
    "timestamp": "2022-02-15T18:26:57.000Z",
    "token_id": "token1",
    "transaction_type": "BULKTRANSFER",
    "number_of_sub_transactions": 2,
    "balance": 9800,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~07331a1f7be99d6750973674a783da9ec9ca17df23747cdf52d388865d93f9a"
,
    "transacted_amount": 10000,
    "timestamp": "2022-02-15T18:26:30.000Z",
    "token_id": "token1",
    "transacted_account":
"oaccount~obptok~88b62f329f20fffc6fc9231cb51019a5e9550c78b657123d140897
62397d2b55",
    "transaction_type": "MINT",
    "balance": 10000,
    "onhold_balance": 0
  }
]
```

GetSubTransactionsById
This method returns an array of subtransaction history details for a specified transaction.

```
func (t *Controller) GetSubTransactionsById(transaction_id string)
(interface{}, error) {
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetSubTransactionsById", "TOKEN",
map[string]string{"transaction_id": transaction_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetSubTransactionsById(transaction_id)
}
```

Parameters:

- `transaction_id string` – The ID of the transaction.

Example:

```
ochain invoke GetAccountSubTransactionHistory
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
```

```
[
  {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c4ca4238a0b923820dcc509a6f75849b",
    "transacted_amount": 100,
    "timestamp": "2022-02-15T18:26:57.000Z",
    "token_id": "token1",
    "transacted_account":
"oaccount~obptok~6600eb38d365552b76f41d4186acece104f31eae331a440f963e6fa75b62f
f21",
    "transaction_type": "DEBIT",
    "balance": 9900,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c81e728d9d4c2f636f067f89cc14862c",
    "transacted_amount": 100,
    "timestamp": "2022-02-15T18:26:57.000Z",
    "token_id": "token1",
    "transacted_account":
"oaccount~obptok~26e046c8ba8b98da2cdabb78113d67200581ea3d4eea5aa3241abd3598e05
d05",
    "transaction_type": "DEBIT",
    "balance": 9800,
    "onhold_balance": 0
  }
]
```

`GetSubTransactionsByIdWithFilters`
This method returns an array of subtransaction history details for a specified transaction.

```
func (t *Controller) GetSubTransactionsByIdWithFilters(transaction_id string,
filters ...account.SubTransactionFilters) (interface{}, error) {
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetSubTransactionsByIdWithFilters",
"TOKEN", map[string]string{"transaction_id": transaction_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetSubTransactionsByIdWithFilters(transaction_id,
filters...)
}
```

Parameters:

- `transaction_id string` – The ID of the transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
ochain invoke GetAccountSubTransactionHistoryWithFilters
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
'{"PageSize":10,"Bookmark":"1"}'
```


```
[
{
"transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c4ca4238a0b923820dcc509a6f75849b",
"transacted_amount": 100,
"timestamp": "2022-02-15T18:26:57.000Z",
"token_id": "token1",
"transacted_account":
"oaccount~obptok~6600eb38d365552b76f41d4186acece104f31eae331a440f963e6fa75b62f
f21",
"transaction_type": "DEBIT",
"balance": 9900,
"onhold_balance": 0
},
{
"transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c81e728d9d4c2f636f067f89cc14862c",
"transacted_amount": 100,
"timestamp": "2022-02-15T18:26:57.000Z",
"token_id": "token1",
"transacted_account":
"oaccount~obptok~26e046c8ba8b98da2cdabb78113d67200581ea3d4eea5aa3241abd3598e05
d05",
"transaction_type": "DEBIT",
"balance": 9800,
"onhold_balance": 0
}
]
```


`GetTransactionById`

This method returns the history of a `Transaction` asset.

```
func (t *Controller) GetTransactionById(transaction_id string) (interface{},
error) {
    return t.Ctx.Transaction.GetTransactionById(transaction_id)
}
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Returns:

- On success, an JSON array of the history for the transaction.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "history": [
        {
            "IsDelete": "false",
            "Timestamp": "2021-08-16 20:19:05.028 +0530 IST",
            "TxId":
"67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220",
            "Value": {
                "Amount": 3,
                "AssetType": "otransaction",
                "FromAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "HoldingId": "ohold~digicur~digiCurr101~op2",
                "NumberOfSubTransactions": 0,
                "Timestamp": "2021-08-16T20:19:05+05:30",
                "ToAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "TokenId": "digiCurr101",
                "TransactionId":
"otransaction~67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220
",
                "TransactionType": "RELEASEHOLD"
            }
        }
    ],
    "transaction_id":
"otransaction~67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220
"
}
```

`DeleteHistoricalTransactions`
This method deletes older transactions from the state database.

```
func (t *Controller) DeleteHistoricalTransactions(timestamp string)
(interface{}, error) {
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Transaction.DeleteHistoricalTransactions",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Transaction.DeleteHistoricalTransactions(timestamp)
}
```

Parameters:

- `timestamp string` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted.

Copyright © 2025, Oracle and/or its affiliates.

Digital Assets

GetAccountTransactionHistoryWithFiltersFromRichHistDB

```
func (t *Controller)
GetAccountTransactionHistoryWithFiltersFromRichHistDB(token_id string, org_id
string, user_id string, custom_endPoint string, bearer_token string,
filters ...account.AccountHistoryFilters) (interface{}, error) {
account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id, user_id)
if err != nil {
return nil, err
}
auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountTransactionHistoryWithFilters
", "TOKEN", map[string]string{"account_id": account_id})
if err != nil && !auth {
return nil, fmt.Errorf("error in authorizing the caller %s", err.Error())
}
// sample format of filter: []string{"3", "", "2022-01-16T15:16:36+00:00",
"2022-01-17T15:16:36+00:00"}
transactionArray, err :=
t.Ctx.Account.GetAccountTransactionHistoryWithFiltersFromRichHistDB(account_id
, org_id, user_id, custom_endPoint, bearer_token, filters...)
return transactionArray, err
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

GetAccountTransactionHistory
This method returns an array of account transaction history details for a specified user and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the specified account owner.

```
func (t *Controller) GetAccountTransactionHistory(token_id string, org_id
string, user_id string) (interface{}, error) {
account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id, user_id)
if err != nil {
return nil, err
}
```


Copyright © 2025, Oracle and/or its affiliates.

```
auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountTransactionHistory",
"TOKEN", map[string]string{"account_id": account_id})
if err != nil && !auth {
return nil, fmt.Errorf("error in authorizing the caller %s", err.Error())
}



transactionArray, err :=
t.Ctx.Account.GetAccountTransactionHistory(account_id, org_id, user_id)
return transactionArray, err
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
[
            {
                "transaction_id":
"otransaction~64c5a4830949eae1424600f3d4a438c6f603a7c3ea31a68e374b899803999e22
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:37:28.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REJECT_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~a4537ef34a955b023b7c205b9abf06a6c79e4fdd761fb24f41b8eb34126b66c0
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:32.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "transaction_type": "APPROVE_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~6237a759422bd9fb112742e8cd7e6450df5a74a32236d9b1005571afed8904a4
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:18.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~06b35071415d74aa1a7c18449149c937d886cae76a832c44cf8d98e84586e76e
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:35:46.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            }
 ]
```

`GetAccountTransactionHistoryWithFilters`

This method returns a filtered array of account transaction history details for a specified user and token. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the specified account owner.

```
func (t *Controller) GetAccountTransactionHistoryWithFilters(token_id string,
filters ...account.AccountHistoryFilters) (interface{}, error) {
org_id, err := t.Ctx.Model.GetTransientMapKeyAsString(constants.OrgIdCC)
if err != nil {
return nil, err
}
user_id, err := t.Ctx.Model.GetTransientMapKeyAsString(constants.UserIdCC)
if err != nil {
```


Copyright © 2025, Oracle and/or its affiliates.

```
return nil, err
}
account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id, user_id)
if err != nil {
return nil, err
}
auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountTransactionHistoryWithFilters
", "TOKEN", map[string]string{"account_id": account_id})
if err != nil && !auth {
return nil, fmt.Errorf("error in authorizing the caller %s", err.Error())
}



// sample format of filter: []string{"3", "", "2022-01-16T15:16:36+00:00",
"2022-01-17T15:16:36+00:00"}
transactionArray, err :=
t.Ctx.Account.GetReconciledTransactionHistory(account_id, org_id, user_id,
filters...)
return transactionArray, err
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
            {
                "transaction_id":
"otransaction~64c5a4830949eae1424600f3d4a438c6f603a7c3ea31a68e374b899803999e22
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:37:28.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REJECT_MINT",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~a4537ef34a955b023b7c205b9abf06a6c79e4fdd761fb24f41b8eb34126b66c0
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:32.000Z",
                "balance": 550,
                "onhold_balance": 10,
                "token_id": "USD",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "APPROVE_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~6237a759422bd9fb112742e8cd7e6450df5a74a32236d9b1005571afed8904a4
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:36:18.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            },
            {
                "transaction_id":
"otransaction~06b35071415d74aa1a7c18449149c937d886cae76a832c44cf8d98e84586e76e
",
                "transacted_amount": 10,
                "timestamp": "2024-12-11T13:35:46.000Z",
                "balance": 540,
                "onhold_balance": 10,
                "token_id": "USD",
                "category": "category value",
                "description": "description value",
                "transacted_account":
"oaccount~9d9806fa92aa0c4fdb34eaffac6e830181b5d47e64fbce752195e83024125ca0",
                "transaction_type": "REQUEST_MINT",
                "transacted_org_id": "CB",
                "transacted_user_id'": "creator_user_cb"
            }
 ]
```


Copyright © 2025, Oracle and/or its affiliates.

GetSubTransactionsById
This method returns an array of subtransaction history details for a specified transaction. This method can be called only by a `Token Admin`, `Token Auditor`, or the `AccountOwner` who called the transaction.

```
func (t *Controller) GetSubTransactionsById(transaction_id string)
(interface{}, error) {
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetSubTransactionsById", "TOKEN",
map[string]string{"transaction_id": transaction_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetSubTransactionsById(transaction_id)
}
```

Parameters:

- `transaction_id string` – The ID of the transaction.

Example:

```
ochain invoke GetAccountSubTransactionHistory
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'

[
  {
    "transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c4ca4238a0b923820dcc509a6f75849b",
    "transacted_amount": 100,
    "timestamp": "2022-02-15T18:26:57.000Z",
    "token_id": "token1",
    "transacted_account":
"oaccount~obptok~6600eb38d365552b76f41d4186acece104f31eae331a440f963e6fa75b62f
f21",
    "transaction_type": "DEBIT",
    "balance": 9900,
    "onhold_balance": 0
  },
  {
    "transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c81e728d9d4c2f636f067f89cc14862c",
    "transacted_amount": 100,
    "timestamp": "2022-02-15T18:26:57.000Z",
    "token_id": "token1",
    "transacted_account":
"oaccount~obptok~26e046c8ba8b98da2cdabb78113d67200581ea3d4eea5aa3241abd3598e05
d05",
    "transaction_type": "DEBIT",
    "balance": 9800,
    "onhold_balance": 0
```


Copyright © 2025, Oracle and/or its affiliates.

```
  }
]
```

GetSubTransactionsByIdWithFilters
This method returns an array of subtransaction history details for a specified transaction.

```
func (t *Controller) GetSubTransactionsByIdWithFilters(transaction_id string,
filters ...account.SubTransactionFilters) (interface{}, error) {
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetSubTransactionsByIdWithFilters",
"TOKEN", map[string]string{"transaction_id": transaction_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetSubTransactionsByIdWithFilters(transaction_id,
filters...)
}
```

Parameters:

- `transaction_id string` – The ID of the transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Example:

```
ochain invoke GetAccountSubTransactionHistoryWithFilters
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
'{"PageSize":10,"Bookmark":"1"}'
```

```
[
{
"transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c4ca4238a0b923820dcc509a6f75849b",
"transacted_amount": 100,
"timestamp": "2022-02-15T18:26:57.000Z",
"token_id": "token1",
"transacted_account":
"oaccount~obptok~6600eb38d365552b76f41d4186acece104f31eae331a440f963e6fa75b62f
f21",
"transaction_type": "DEBIT",
"balance": 9900,
"onhold_balance": 0
},
{
"transaction_id":
"otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b
~c81e728d9d4c2f636f067f89cc14862c",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"transacted_amount": 100,
"timestamp": "2022-02-15T18:26:57.000Z",
"token_id": "token1",
"transacted_account":
"oaccount~obptok~26e046c8ba8b98da2cdabb78113d67200581ea3d4eea5aa3241abd3598e05
d05",
"transaction_type": "DEBIT",
"balance": 9800,
"onhold_balance": 0
}
]
```

GetTransactionById

This method returns the history of a `Transaction` asset. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by a transaction participant (sender, recipient, notary).

```
func (t *Controller) GetTransactionById(transaction_id string) (interface{},
error) {
    return t.Ctx.Transaction.GetTransactionById(transaction_id)
}
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Returns:

- On success, an JSON array of the history for the transaction.

Return Value Example:

```
{
    "history": [
        {
            "IsDelete": "false",
            "Timestamp": "2021-08-16 20:19:05.028 +0530 IST",
            "TxId":
"67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220",
            "Value": {
                "Amount": 3,
                "AssetType": "otransaction",
                "FromAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "HoldingId": "ohold~digicur~digiCurr101~op2",
                "NumberOfSubTransactions": 0,
                "Timestamp": "2021-08-16T20:19:05+05:30",
                "ToAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "TokenId": "digiCurr101",
                "TransactionId":
"otransaction~67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
                "TransactionType": "RELEASEHOLD"
            }
        }
    ],
    "transaction_id":
"otransaction~67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220
"
}
```

`DeleteHistoricalTransactions`
This method deletes older transactions from the state database.

```
func (t *Controller) DeleteHistoricalTransactions(timestamp string)
(interface{}, error) {
      auth, err :=
t.Ctx.Auth.CheckAuthorization("Transaction.DeleteHistoricalTransactions",
"TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Transaction.DeleteHistoricalTransactions(timestamp)
}
```

Parameters:

- `timestamp string` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted.

Methods for Token Behavior Management - Mintable Behavior

- Base
- Digital Assets

Base

`IssueTokens`
This method mints tokens, which are then owned by the caller of the method. The caller must have an account and the minter role. The number of tokens that can be minted is limited by the `max_mint_quantity` property of `mintable` behavior in the specification file. If the `max_mint_quantity` property is not specified, an unlimited number of tokens can be minted. The quantity must be within the decimal values specified by the `decimal` parameter of the

Copyright © 2025, Oracle and/or its affiliates.

`divisible` behavior in the specification file. This method can be called only by the `AccountOwner` of the account with the minter role.

```
func (t *Controller) IssueTokens(token_id string, quantity float64)
(interface{}, error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.Mint(quantity, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `quantity float64` – The number of tokens to mint.

Returns:

- On success, a message with account details.

Return Value Example:

```
{"msg":"Successfully minted 100 tokens to account
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"}
```

`GetTotalMintedTokens`
This method returns the total number of minted tokens for a specified token. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) GetTotalMintedTokens(token_id string) (interface{},
error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTotalMintedTokens",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.GetTotalMintedTokens(tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON string indicating the total number of tokens.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"total minted amount for token with id digiCurr101 is
1000","quantity":1000}
```

GetNetTokens
This method returns the total net number of tokens available in the system for a specified token. The net token total is the amount of tokens remaining after tokens are burned. In equation form net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
func (t *Controller) GetNetTokens(token_id string) (interface{}, error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetNetTokens", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.GetNetTokens(tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON string indicating the net number of tokens.

Return Value Example:

```
{"msg":"net minted amount for token with id digiCurr101 is
1000","quantity":1000}
```

Digital Assets

RequestMint
This method can be called by a minter to send a request to the minter notary to create a specified amount of tokens.

```
func (t *Controller) RequestMint(token_id string, operation_id string,
notary_org_id string, notary_user_id string, quantity float64,
timeToExpiration string, info_details ...token.InfoDetails) (interface{},
error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
notary_account_id, err := t.Ctx.Account.GenerateAccountId(token_id,
```


Copyright © 2025, Oracle and/or its affiliates.

```
notary_org_id, notary_user_id)
if err != nil {
return nil, fmt.Errorf("error in getting notary account id from org_id: %s
and user_id: %s with token_id: %s, error %s ", notary_org_id, notary_user_id,
token_id, err.Error())
 }
return t.Ctx.Token.Hold(operation_id, "", notary_account_id, quantity,
timeToExpiration, tokenAssetValue.Interface(), constants.HoldMint,
info_details...)
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.
- `notary_org_id: string` – The membership service provider (MSP) ID of the minter notary who will process the request.
- `notary_user_id: string` – The user name or email ID of the minter notary who will process the request.
- `quantity: number` – The amount of tokens to mint.
- `TimeToExpiration` – The time after which the minting request expires and is no longer valid.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"AccountId
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(org_id: Org1MSP, user_id: admin) has successfully submitted request to mint
100 tokens",
}
```

`ApproveMint`
This method can be called by a minter notary to approve a minting request.

```
func (t *Controller) ApproveMint(token_id string, operation_id string)
(interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
```


Copyright © 2025, Oracle and/or its affiliates.

```
}
return t.Ctx.Token.ExecuteHold(operation_id, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.

Return Value Example:

```
{
msg:
"Successfully minted 100 tokens to Account Id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(org_id: Org1MSP, user_id: admin)"
}
```

`RejectMint`
This method can be called by a minter notary to reject a minting request.

```
func (t *Controller) RejectMint(token_id string, operation_id string)
(interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
return t.Ctx.Token.ReleaseHold(operation_id, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.

Return Value Example:

```
{
 msg: "Successfully rejected mint request with Operation Id 'operation1' to
mint 100 tokens of token id token"
}
```

`IssueTokens`
This method mints tokens, which are then owned by the caller of the method.

```
func (t *Controller) IssueTokens(token_id string, quantity float64,
info_details ...token.InfoDetails) (interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
 }
```


Copyright © 2025, Oracle and/or its affiliates.

```
return t.Ctx.Token.Mint(quantity, tokenAssetValue.Interface(),
info_details...)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to mint.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"Successfully minted 100 tokens to Account Id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin)"
}
```

`GetTotalMintedTokens`
This method returns the total number of minted tokens for a specified token. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetTotalMintedTokens(token_id string) (interface{},
error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetTotalMintedTokens",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.GetTotalMintedTokens(tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON string indicating the total number of tokens.

Return Value Example:

```
{"msg":"total minted amount for token with id digiCurr101 is
1000","quantity":1000}
```

`GetNetTokens`
This method returns the total net number of tokens available in the system for a specified token. The net token total is the amount of tokens remaining after tokens are burned. In equation form net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin`, or `Org Auditor`.

```
func (t *Controller) GetNetTokens(token_id string) (interface{}, error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    auth, err := t.Ctx.Auth.CheckAuthorization("Token.GetNetTokens", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    return t.Ctx.Token.GetNetTokens(tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.

Returns:

- On success, a JSON string indicating the net number of tokens.

Return Value Example:

```
{"msg":"net minted amount for token with id digiCurr101 is
1000","quantity":1000}
```

Methods for Token Behavior Management - Transferable Behavior

- Base
- Digital Assets

Copyright © 2025, Oracle and/or its affiliates.

Base

`TransferTokens`
This method transfers tokens from the caller to a specified account. The caller of the method must have an account. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. This method can be called only by the `AccountOwner` of the account.

```
func (t *Controller) TransferTokens(token_id string, to_org_id string,
to_user_id string, quantity float64) (interface{}, error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    to_account_id, err := t.Ctx.Account.GenerateAccountId(token_id,
to_org_id, to_user_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.Transfer(to_account_id, quantity,
tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `to_org_id string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `to_user_id string` – The user name or email ID of the receiver.
- `quantity float64` – The number of tokens to transfer.

Returns:

- On success, a message with details for both accounts.

Return Value Example:

```
{"msg":"successfully transferred 1 tokens from account
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) to account
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (org_id : Org1MSP, user_id : admin)"}
```

`BulkTransferTokens`
This method transfers tokens in bulk from the caller account to the accounts that are specified in the `flow` object. The quantities must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.The caller of this method must have an account already created. This method can be called only by the `AccountOwner` of the account.

```
func (t *Controller) BulkTransferTokens(token_id string,
flow[]map[string]interface{}) (interface{}, error) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.BulkTransfer(flow, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `flow[]map[string]interface{}` – An array of JSON objects that specify receiver details and quantities.
  - `to_org_id string` – The membership service provider (MSP) ID of the receiver in the current organization.
  - `to_user_id string` – The user name or email ID of the receiver.
  - `quantity float64` – The number of tokens to transfer.

  You specify the `flow` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code:

```
[
  { "to_org_id": "Org1MSP", "to_user_id": "user1", "quantity": 10 },
  { "to_org_id": "Org1MSP", "to_user_id": "user2", "quantity": 10 }
]
```

  CLI / Postman:

```
"[{ \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user1\", \"quantity\":
10 }, { \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user2\",
\"quantity\": 10 }]"
```

Returns:

- A message indicating success.

Return Value Example:

```
{
    "from_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "msg": "Successfully transferred 2 tokens from Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)",
    "sub_transactions": [
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        },
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df"
        }
    ]
}
```

Digital Assets

`TransferTokens`
This method transfers tokens from the caller to a specified account.

```
func (t *Controller) TransferTokens(token_id string, to_org_id string,
to_user_id string, quantity float64, info_details ...token.InfoDetails)
(interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
to_account_id, err := t.Ctx.Account.GenerateAccountId(token_id, to_org_id,
to_user_id)
if err != nil {
return nil, err
 }
return t.Ctx.Token.Transfer(to_account_id, quantity,
tokenAssetValue.Interface(), info_details...)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `to_org_id: string` – The membership service provider (MSP) ID of the receiver (payee) in the current organization.
- `to_user_id: string` – The user name or email ID of the receiver.
- `quantity: number` – The number of tokens to transfer.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
 msg: "Successfully transferred 100 tokens from account id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin) to account id:
oaccount~7yuijg39b4e1e4136dd86a806020c97a930909325340481b8fdhjklliugbv699
(Org-Id: Org1MSP, User-Id: user)",
}
```

`BulkTransferTokens`
This method transfers tokens in bulk from the caller account to the accounts that are specified in the `flow` object. The quantities must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.The caller of this method must have an account already created. This method can be called only by the `AccountOwner` of the account.

```
func (t *Controller) BulkTransferTokens(token_id string,
flow[]map[string]interface{}) (interface{}, error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.BulkTransfer(flow, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `flow[]map[string]interface{}` – An array of JSON objects that specify receiver details and quantities.
  - `to_org_id string` – The membership service provider (MSP) ID of the receiver in the current organization.
  - `to_user_id string` – The user name or email ID of the receiver.
  - `quantity float64` – The number of tokens to transfer.

  You specify the `flow` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code:

  ```
  [
    { "to_org_id": "Org1MSP", "to_user_id": "user1", "quantity": 10 },
    { "to_org_id": "Org1MSP", "to_user_id": "user2", "quantity": 10 }
  ]
  ```


Copyright © 2025, Oracle and/or its affiliates.

CLI / Postman:

```
"[{ \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user1\", \"quantity\":
10 }, { \"to_org_id\": \"Org1MSP\", \"to_user_id\": \"user2\",
\"quantity\": 10 }]"
```

Returns:

- A message indicating success.

Return Value Example:

```
{
    "from_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "msg": "Successfully transferred 2 tokens from Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)",
    "sub_transactions": [
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e"
        },
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df"
        }
    ]
}
```

Methods for Token Behavior Management - Holdable Behavior

- Base
- Digital Assets

Base

`HoldTokens`
This method creates a hold on behalf of the owner of the tokens with the `to_account_id` account. A notary account is specified, which is responsible to either complete or release the hold. When the hold is created, the specified token balance from the payer is put on hold. A held balance cannot be transferred until the hold is either completed or released. The caller of

Copyright © 2025, Oracle and/or its affiliates.

this method must have an account already created. This method can be called only by the `AccountOwner` of the account.

```
func (t *Controller) HoldTokens(token_id string, operation_id string,
to_org_id string, to_user_id string, notary_org_id string, notary_user_id
string, quantity float64, TimeToExpiration string) (interface{}, error) {
      tokenAssetValue, err := t.getTokenObject(token_id)
      if err != nil {
            return nil, err
      }
      notary_account_id, err := t.Ctx.Account.GenerateAccountId(token_id,
notary_org_id, notary_user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting notary account id from
org_id: %s and user_id: %s with token_id: %s, error %s ", notary_org_id,
notary_user_id, token_id, err.Error())
      }
      to_account_id, err := t.Ctx.Account.GenerateAccountId(token_id,
to_org_id, to_user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting to_account id from
org_id: %s and user_id: %s with token_id: %s, error %s ", to_org_id,
to_user_id, token_id, err.Error())
      }
      return t.Ctx.Token.Hold(operation_id, to_account_id, notary_account_id,
quantity, TimeToExpiration, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_org_id string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `to_user_id string` – The user name or email ID of the receiver.
- `notary_org_id string` – The membership service provider (MSP) ID of the notary in the current organization.
- `notary_user_id string` – The user name or email ID of the notary.
- `quantity float64` – The number of tokens to put on hold.
- `time_to_expiration` – The time when the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12:46:06Z`.

Returns:

- On success, a message with the caller's account and hold details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"AccountId
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) is successfully holding 2 tokens"}
```

`ExecuteHoldTokens`
This method completes a hold on a token. A quantity of tokens previously held by a token owner is transferred to a receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can be called only by the `AccountOwner` ID with the `notary` role. The hold can only be completed by the notary.

```
func (t *Controller) ExecuteHoldTokens(token_id string, operation_id string,
quantity float64) (interface{} error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.ExecuteHold(operation_id, quantity,
tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity float64` – The number of on-hold tokens to transfer.

Returns:

- On success, a message with the caller's account ID and the quantity of the transaction.

Return Value Example:

```
{"msg":"Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) has successfully executed '1'
tokens(digiCurr101) from the hold with Operation Id 'op1'"}
```

`ReleaseHoldTokens`
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `Account Owner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
func (t *Controller) ReleaseHoldTokens(token_id string, operation_id string)
(interface{} error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    return t.Ctx.Token.ReleaseHold(operation_id, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a message indicating that the hold was released.

Return Value Example:

```
{"msg":"Successfully released '3' tokens from Operation Id 'op2' to Account
Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"}
```

`GetOnHoldIds`
This method returns a list of all of the holding IDs for a specified account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or the `Account Owner` of the account.

```
func (t *Controller) GetOnHoldIds(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetOnHoldIds",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetOnHoldIDs(account_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON list of holding IDs. A holding ID is a concatenation of the `ohold` asset type, the name of the token, the token ID, and the operation ID.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"holding_ids":["ohold~loyaltok123~t1~op1"],"msg":"Holding Ids are:
[ohold~loyaltok123~t1~op1]"}
```

`GetOnHoldDetailsWithOperationId`
This method returns the on-hold transaction details for a specified operation ID and token. This method can be called by anyone.

```
func (t *Controller) GetOnHoldDetailsWithOperationId(token_id string,
operation_id string) (interface{} error) {
    return t.Ctx.Hold.GetOnHoldDetailsWithOperationId(token_id, operation_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON hold object that includes the following properties:
- `HoldingId` – The holding ID of the transaction.
- `OperationId` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `FromAccountId` – The account ID of the current owner of the on-hold tokens.
- `ToAccountId` – The account ID of the receiver.
- `NotaryAccountId` – The account ID of the notary.
- `TokenId` – The ID of the saved token.
- `Quantity` – The amount of tokens that are on hold for the holding ID.
- `TimeToExpiration` – The duration until the hold expires.

Return Value Example:

```
{
    "AssetType": "ohold",
    "HoldingId": "ohold~digicur~digiCurr101~op1",
    "OperationId": "op1",
    "TokenName": "digicur",
    "FromAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "ToAccountId":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
    "NotaryAccountId":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "TokenId": "digiCurr101",
    "Quantity": 2,
    "TimeToExpiration": "0"
}
```

GetOnHoldBalanceWithOperationId
This method returns the on-hold balance for a specified operation ID and token. This method can be called by anyone.

```
func (t *Controller) GetOnHoldBalanceWithOperationId(token_id string,
operation_id string) (interface{} error) {
    return t.Ctx.Hold.GetOnHoldBalanceWithOperationId(token_id, operation_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON string indicating the holding balance.

Return Value Example:

```
{
    "holding_balance": 10,
    "msg": "Current Holding Balance of OperationId opr_121 for token
digiCurr101 is : 10"
}
```

Digital Assets

HoldTokens
This method creates a hold on behalf of the owner of the tokens with the `to_account_id` account.

```
func (t *Controller) HoldTokens(token_id string, operation_id string,
to_org_id string, to_user_id string, notary_org_id string, notary_user_id
string, quantity float64, timeToExpiration string,
info_details ...token.InfoDetails) (interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
notary_account_id, err := t.Ctx.Account.GenerateAccountId(token_id,
notary_org_id, notary_user_id)
if err != nil {
return nil, fmt.Errorf("error in getting notary account id from org_id: %s
and user_id: %s with token_id: %s, error %s ", notary_org_id, notary_user_id,
token_id, err.Error())
```


Copyright © 2025, Oracle and/or its affiliates.

```
}
to_account_id, err := t.Ctx.Account.GenerateAccountId(token_id, to_org_id,
to_user_id)
if err != nil {
return nil, fmt.Errorf("error in getting to_account id from org_id: %s and
user_id: %s with token_id: %s, error %s ", to_org_id, to_user_id, token_id,
err.Error())
 }
return t.Ctx.Token.Hold(operation_id, to_account_id, notary_account_id,
quantity, timeToExpiration, tokenAssetValue.Interface(),
constants.HoldTransfer, info_details...)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_org_id: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `to_user_id: string` – The user name or email ID of the receiver.
- `notary_org_id: string` – The membership service provider (MSP) ID of the notary in the current organization.
- `notary_user_id: string` – The user name or email ID of the notary.
- `quantity: number` – The number of tokens to put on hold.
- `time_to_expiration` – The time when the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12:46:06Z`.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"AccountId
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin) is successfully holding 100 tokens",
}
```

`ExecuteHoldTokens`

This method completes a hold on a token. A quantity of tokens previously held by a token owner is transferred to a receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can

Copyright © 2025, Oracle and/or its affiliates.

be called only by the `AccountOwner` ID with the `notary` role. The hold can only be completed by the notary.

```
func (t *Controller) ExecuteHoldTokens(token_id string, operation_id string,
quantity float64) (interface{} error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.ExecuteHold(operation_id, quantity,
tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity float64` – The number of on-hold tokens to transfer.

Returns:

- On success, a message with the caller's account ID and the quantity of the transaction.

Return Value Example:

```
{"msg":"Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) has successfully executed '1'
tokens(digiCurr101) from the hold with Operation Id 'op1'"}
```

`ReleaseHoldTokens`
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `Account Owner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
func (t *Controller) ReleaseHoldTokens(token_id string, operation_id string)
(interface{} error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
    }
    return t.Ctx.Token.ReleaseHold(operation_id, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message indicating that the hold was released.

Return Value Example:

```
{"msg":"Successfully released '3' tokens from Operation Id 'op2' to Account
Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"}
```

`GetOnHoldIds`
This method returns a list of all of the holding IDs for a specified account. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the specified account owner.

```
func (t *Controller) GetOnHoldIds(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, err
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("Account.GetOnHoldIds",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.GetOnHoldIDs(account_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON list of holding IDs. A holding ID is a concatenation of the `ohold` asset type, the name of the token, the token ID, and the operation ID.

Return Value Example:

```
{"holding_ids":["ohold~loyaltok123~t1~op1"],"msg":"Holding Ids are:
[ohold~loyaltok123~t1~op1]"}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetOnHoldDetailsWithOperationId`
This method returns the on-hold transaction details for a specified operation ID and token. This method can be called only by the `Token Admin`, `Token Auditor`, or transaction participant (sender, recipient, notary).

```
func (t *Controller) GetOnHoldDetailsWithOperationId(token_id string,
operation_id string) (interface{} error) {
    return t.Ctx.Hold.GetOnHoldDetailsWithOperationId(token_id, operation_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON hold object that includes the following properties:
- `HoldingId` – The holding ID of the transaction.
- `OperationId` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `FromAccountId` – The account ID of the current owner of the on-hold tokens.
- `ToAccountId` – The account ID of the receiver.
- `NotaryAccountId` – The account ID of the notary.
- `TokenId` – The ID of the saved token.
- `Quantity` – The amount of tokens that are on hold for the holding ID.
- `TimeToExpiration` – The duration until the hold expires.

Return Value Example:

```
{
    "AssetType": "ohold",
    "HoldingId": "ohold~digicur~digiCurr101~op1",
    "OperationId": "op1",
    "TokenName": "digicur",
    "FromAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "ToAccountId":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
    "NotaryAccountId":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
    "TokenId": "digiCurr101",
    "Quantity": 2,
    "TimeToExpiration": "0"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetOnHoldBalanceWithOperationId`
This method returns the on-hold balance for a specified operation ID and token. This method can be called only by the `Token Admin`, `Token Auditor`, or transaction participant (sender, recipient, notary).

```
func (t *Controller) GetOnHoldBalanceWithOperationId(token_id string,
operation_id string) (interface{} error) {
    return t.Ctx.Hold.GetOnHoldBalanceWithOperationId(token_id, operation_id)
}
```

Parameters:

- `token_id string` – The ID of the token.
- `operation_id string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a JSON string indicating the holding balance.

Return Value Example:

```
{
    "holding_balance": 10,
    "msg": "Current Holding Balance of OperationId opr_121 for token
digiCurr101 is : 10"
}
```

Methods for Token Behavior Management - Burnable Behavior

- Base
- Digital Assets

Base

`BurnTokens`
This method deactivates, or burns, tokens from the transaction caller's account. The caller of this method must have an account and the burner role. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. This method can be called by the `Account Owner` of the account with the burner role.

```
func (t *Controller) BurnTokens(token_id string, quantity float64)
(interface{} error) {
    tokenAssetValue, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, err
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    return t.Ctx.Token.Burn(quantity, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id string` – The ID of the token.
- `quantity float64` – The number of tokens to burn.

Returns:

- On success, a success message with the quantity of tokens burned and the account ID.

Return Value Example:

```
{"msg":"Successfully burned 1 tokens from account id:
oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c7
06e (org_id : Org1MSP, user_id : user2)"}
```

Digital Assets

`RequestBurn`
This method can be called by a burner to send a request to the burner notary to destroy a specified amount of tokens, which must be less than or equal to their available balance. When a burn request starts, the specified amount is immediately deducted from the available balance and added to the `onhold_burn_balance` field. If the request is approved, the tokens are burned. If the request is rejected, the tokens are returned from the `onhold_burn_balance` field to the available balance.

```
func (t *Controller) RequestBurn(token_id string, operation_id string,
notary_org_id string, notary_user_id string, quantity float64,
timeToExpiration string, info_details ...token.InfoDetails) (interface{},
error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
notary_account_id, err := t.Ctx.Account.GenerateAccountId(token_id,
notary_org_id, notary_user_id)
if err != nil {
return nil, fmt.Errorf("error in getting notary account id from org_id: %s
and user_id: %s with token_id: %s, error %s ", notary_org_id, notary_user_id,
token_id, err.Error())
 }
return t.Ctx.Token.Hold(operation_id, "", notary_account_id, quantity,
timeToExpiration, tokenAssetValue.Interface(), constants.HoldBurn,
info_details...)
}
```

Parameters:

- `token_id: string` – The ID of the token to burn.

Copyright © 2025, Oracle and/or its affiliates.

- `operation_id: string` – The unique operation ID that represents the burn request.
- `notary_org_id: string` – The membership service provider (MSP) ID of the burner notary who will process the request.
- `notary_user_id: string` – The user name or email ID of the burner notary who will process the request.
- `quantity: number` – The amount of tokens to burn.
- `time_to_expiration` – The time after which the burning request expires and is no longer valid.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"AccountId
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(org_id: Org1MSP, user_id: admin) has successfully submitted request to mint
100 tokens",
}
```

ApproveBurn
This method can be called by a burner notary to approve a burning request.

```
func (t *Controller) ApproveBurn(token_id string, operation_id string)
(interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
return t.Ctx.Token.ExecuteHold(operation_id, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.

Return Value Example:

```
{
msg:
"Successfully burned 100 tokens from account id:
```


Copyright © 2025, Oracle and/or its affiliates.

```
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(org_id: Org1MSP, user_id: admin)"
}
```

RejectBurn
This method can be called by a burner notary to reject a burning request.

```
func (t *Controller) RejectBurn(token_id string, operation_id string)
(interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
}
return t.Ctx.Token.ReleaseHold(operation_id, tokenAssetValue.Interface())
}
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.

Return Value Example:

```
{
 msg: "Successfully rejected burn request with Operation Id 'operation1' to
burn 100 tokens of token id token",
}
```

GetAccountOnHoldBurnBalance
This method returns the on-hold burn balance for a specified user. This method can be called only by the `Token Admin`, `Token Auditor`, `Org Admin`, `Org Auditor`, or by the account owner.

```
func (t *Controller) GetAccountOnHoldBurnBalance(token_id string, org_id
string, user_id string) (interface{}, error) {
    account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.Auth.CheckAuthorization("Account.GetAccountOnHoldBurnBalance", "TOKEN",
map[string]string{"account_id": account_id})
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    response, err := t.Ctx.Account.GetAccountOnHoldBurnBalance(account_id)
    return response, err
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `token_id: string` – The ID of the token to burn.
- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Return Value Example:

```
{
  "msg": "Total On Hold Burning Balance is: 10",
  "onhold_burn_balance": 10
}
```

`BurnTokens`
This method deactivates, or burns, tokens from the transaction caller's account.

```
func (t *Controller) BurnTokens(token_id string, quantity float64,
info_details ...token.InfoDetails) (interface{}, error) {
tokenAssetValue, err := t.getTokenObject(token_id)
if err != nil {
return nil, err
 }
return t.Ctx.Token.Burn(quantity, tokenAssetValue.Interface(),
info_details...)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to burn.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
msg:
"Successfully burned 100 tokens from account id:
oaccount~95be539b4e1e4136dd86a806020c97a930909325340481b8fd88d339874fa699
(Org-Id: Org1MSP, User-Id: admin)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Custom Methods

You can use the token SDK methods to write custom methods for your business application.

Make sure to track the return value when you use the token SDK methods. Also, to avoid double-spending, do not combine multiple async functions that operate on the same key-value pairs in the state database. Instead, use the `BulkTransferTokens` method, to make multiple transfers in one method.

The following example shows how to use token SDK methods in custom methods. When the `BuyTicket` method is called, it transfers 20 tokens from the caller's account to the seller's account, and returns the transaction message of the transfer.

```
func (t *Controller) BuyTicket(TokenId string, SellerOrgId string,
SellerUserId string) (interface{}, error){
    token, err := t.Ctx.Token.Get(TokenId)
    if err != nil {
        return nil, err
    }

    /**
    * The following method t.Ctx.Account.GenerateAccountId(TokenId,
SellerOrgId, SellerUserId) generates account id of the seller
    */
    sellerAccountId, err := t.Ctx.Account.GenerateAccountId(TokenId,
SellerOrgId, SellerUserId)
    if err != nil {
        return nil, err
    }

    /**
    * The following method t.Ctx.Token.Transfer(sellerAccountId, 20, token)
transfers the quantity 20 from caller's
    * account & to seller's account.
    */
    transaction, err := t.Ctx.Token.Transfer(sellerAccountId, 20, token)
    if err != nil {
        return nil, err
    }
    return transaction, nil
}
```

Token SDK Methods

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior

Copyright © 2025, Oracle and/or its affiliates.

– Holdable Behavior
– Burnable Behavior

Methods for Access Control Management

The token SDK provides an access control function. Some methods can be called only by a `Token Admin`, `Org Admin` or `AccountOwner` of the token. You can use this feature to ensure that operations are carried out only by the intended users. Any unauthorized access results in an error. To use the access control function, import the `Authorization` class from the `../lib/auth` module.

```
import { Authorization } from '../lib/auth';
```

- Base
- Digital Assets

Base

`AddAdmin`
This method adds a user as a `Token Admin` of the token chaincode.

```
Ctx.AddAdmin(org_id string, user_id string) (interface{}, error)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a success message and details for the `Token Admin` user that was added. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"msg":"Successfully added Admin (Org_Id: Org1MSP, User_Id: user2)"}
```

`RemoveAdmin`
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.RemoveAdmin(org_id string, user_id string) (interface{}, error)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a success message and details for the `Token Admin` user that was removed. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"msg":"Successfuly removed Admin (Org_Id Org1MSP User_Id user1)"}
```

`IsUserTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`.

```
Ctx.Auth.IsUserTokenAdmin()  (bool, error)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- A Boolean response and an error message if an error is encountered.

Return Value Example:

```
{"result":false}
```

`GetAllAdmins`
This method returns a list of all users who are a `Token Admin` of the token chaincode.

```
Ctx.GetAllAdmins() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a list of all users who are a `Token Admin` of the token chaincode. On error, a non-nil error object containing an error message.

Return Value Example:

```
[{"OrgId":"Org1MSP","UserId":"admin"},{"OrgId":"Org1MSP","UserId":"user1"}]
```

`GetAllAdminUsers`
This method returns a list of all users who are a `Token Admin` of the token chaincode.

```
Ctx.Admin.GetAllAdminUsers() (interface{}, error)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- none

Returns:

- On success, a list of all users who are a `Token Admin` of the token chaincode in `map[string]interface{}` form. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"admins":[{"OrgId":"Org1MSP","UserId":"admin"},
{"OrgId":"Org1MSP","UserId":"user1"}]}
```

`CheckAuthorization`
Use this method to add access control to your chaincode. Many of the automatically generated token methods use access control. The mapping between the SDK receiver and the methods which have access control is described in the `oChainUtil.go` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
var t TokenAccess
      var r RoleAccess
      var a AccountAccess
      var as AccountStatusAccess
      var h HoldAccess
      var ad AdminAccess
      var trx TransactionAccess
      var tc TokenConversionAccess
      var auth AuthAccess

      auth.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner"}

      trx.DeleteHistoricalTransactions = []string{"Admin"}
      ad.AddAdmin = []string{"Admin"}
      ad.RemoveAdmin = []string{"Admin"}
      ad.GetAllAdmins = []string{"Admin", "OrgAdmin"}
      ad.AddOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
      ad.RemoveOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
      ad.GetOrgAdmins = []string{"Admin", "OrgAdmin"}
      ad.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner", "OrgAdmin"}
      t.Save = []string{"Admin"}
      t.GetAllTokens = []string{"Admin", "OrgAdmin"}
      t.Update = []string{"Admin"}
      t.GetTokenDecimals = []string{"Admin", "OrgAdmin"}
      t.GetTokensByName = []string{"Admin", "OrgAdmin"}
      t.AddRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
      t.RemoveRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
      t.IsInRole = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
      t.GetTotalMintedTokens = []string{"Admin", "OrgAdmin"}
      t.GetNetTokens = []string{"Admin", "OrgAdmin"}
      t.Get = []string{"Admin", "OrgAdmin"}
      t.GetTokenHistory = []string{"Admin", "OrgAdmin"}
      a.CreateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
      a.AssociateToken = []string{"Admin", "OrgAdminAccountIdCheck"}
```


Copyright © 2025, Oracle and/or its affiliates.

```
      a.GetAllAccounts = []string{"Admin"}
      a.GetAllOrgAccounts = []string{"Admin", "OrgAdminOrgIdCheck"}
      a.GetAccount = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.History = []string{"Admin", "AccountOwner"}
      a.GetAccountTransactionHistory = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
      a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
      a.GetSubTransactionsById = []string{"Admin", "TransactionInvoker"}
      a.GetSubTransactionsByIdWithFilters = []string{"Admin",
"TransactionInvoker"}
      a.GetAccountBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.GetAccountOnHoldBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.GetOnHoldIds = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.GetAccountsByUser = []string{"Admin", "OrgAdminOrgIdCheck",
"MultipleAccountOwner"}

      as.Get = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
      as.ActivateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
      as.SuspendAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
      as.DeleteAccount = []string{"Admin", "OrgAdminOrgIdCheck"}

      r.GetAccountsByRole = []string{"Admin"}
      r.GetUsersByRole = []string{"Admin"}
      r.GetOrgAccountsByRole = []string{"Admin", "OrgAdminOrgIdCheck"}
      r.GetOrgUsersByRole = []string{"Admin", "OrgAdminOrgIdCheck"}

      tc.InitializeExchangePoolUser = []string{"Admin"}
      tc.AddConversionRate = []string{"Admin"}
      tc.UpdateConversionRate = []string{"Admin"}
      tc.GetConversionRate = []string{"Admin", "OrgAdmin", "AnyAccountOwner"}
      tc.GetConversionRateHistory = []string{"Admin", "OrgAdmin",
"AnyAccountOwner"}
      tc.TokenConversion = []string{"Admin", "AnyAccountOwner"}
      tc.GetExchangePoolUser = []string{"Admin"}


Ctx.Auth.CheckAuthorization(classFuncName string, args ...string) (bool,
error)
```

Parameters:

- `classFuncName string` – The map value between the receivers and methods as described in the `oChainUtil.go` file.
- `args` – A variable argument, where `args[0]` is the constant `TOKEN` and `args[1]` is the `account_id` argument, if required.

Returns:

- A Boolean response and an error message if an error is encountered.

Copyright © 2025, Oracle and/or its affiliates.

`AddOrgAdmin`
This method adds a user as an `Org Admin` of the organization.

```
Ctx.Admin.AddOrgAdmin(org_id, user_id) (interface{}, error)
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`RemoveOrgAdmin`
This method removes a user as an `Org Admin` of an organization.

```
Ctx.Admin.RemoveOrgAdmin(org_id, user_id) (interface{}, error)
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

`GetOrgAdmins`
This method returns a list of all users who are an `Org Admin` of an organization.

```
Ctx.Admin.GetAllOrgAdmins() (interface{}, error)
```

Parameters:

- none

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a JSON list that includes `OrgId` and `UserId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin1"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin2"
        }
    ]
}
```

Digital Assets

`AddAdmin`
This method adds a user as a `Token Admin` of the token chaincode.

```
Ctx.AddAdmin(org_id string, user_id string) (interface{}, error)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a success message and details for the `Token Admin` user that was added. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"msg":"Successfully added Admin (Org_Id: Org1MSP, User_Id: user2)"}
```

`RemoveAdmin`
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.RemoveAdmin(org_id string, user_id string) (interface{}, error)
```

Parameters:

- `user_id` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- On success, a success message and details for the `Token Admin` user that was removed. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"msg":"Successfuly removed Admin (Org_Id Org1MSP User_Id user1)"}
```

`IsUserTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`.

```
Ctx.Auth.IsUserTokenAdmin()  (bool, error)
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Returns:

- A Boolean response and an error message if an error is encountered.

Return Value Example:

```
{"result":false}
```

`GetAllAdmins`
This method returns a list of all users who are a `Token Admin` of the token chaincode.

```
Ctx.GetAllAdmins() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a list of all users who are a `Token Admin` of the token chaincode. On error, a non-nil error object containing an error message.

Return Value Example:

```
[{"OrgId":"Org1MSP","UserId":"admin"},{"OrgId":"Org1MSP","UserId":"user1"}]
```


Copyright © 2025, Oracle and/or its affiliates.

GetAllAdminUsers
This method returns a list of all users who are a `Token Admin` of the token chaincode.

```
Ctx.Admin.GetAllAdminUsers() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a list of all users who are a `Token Admin` of the token chaincode in `map[string]interface{}` form. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"admins":[{"OrgId":"Org1MSP","UserId":"admin"},
{"OrgId":"Org1MSP","UserId":"user1"}]}
```

CheckAuthorization
Use this method to add access control to your chaincode. Many of the automatically generated token methods use access control. The mapping between the SDK receiver and the methods which have access control is described in the `oChainUtil.go` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
var t TokenAccess
      var r RoleAccess
      var a AccountAccess
      var as AccountStatusAccess
      var h HoldAccess
      var ad AdminAccess
      var trx TransactionAccess
      var tc TokenConversionAccess
      var auth AuthAccess

      auth.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner"}

      trx.DeleteHistoricalTransactions = []string{"Admin"}
      ad.AddAdmin = []string{"Admin"}
      ad.RemoveAdmin = []string{"Admin"}
      ad.GetAllAdmins = []string{"Admin", "OrgAdmin"}
      ad.AddOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
      ad.RemoveOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
      ad.GetOrgAdmins = []string{"Admin", "OrgAdmin"}
      ad.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner", "OrgAdmin"}
      t.Save = []string{"Admin"}
      t.GetAllTokens = []string{"Admin", "OrgAdmin"}
      t.Update = []string{"Admin"}
      t.GetTokenDecimals = []string{"Admin", "OrgAdmin"}
      t.GetTokensByName = []string{"Admin", "OrgAdmin"}
      t.AddRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
      t.RemoveRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
```


Copyright © 2025, Oracle and/or its affiliates.

```
      t.IsInRole = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
      t.GetTotalMintedTokens = []string{"Admin", "OrgAdmin"}
      t.GetNetTokens = []string{"Admin", "OrgAdmin"}
      t.Get = []string{"Admin", "OrgAdmin"}
      t.GetTokenHistory = []string{"Admin", "OrgAdmin"}
      a.CreateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
      a.AssociateToken = []string{"Admin", "OrgAdminAccountIdCheck"}
      a.GetAllAccounts = []string{"Admin"}
      a.GetAllOrgAccounts = []string{"Admin", "OrgAdminOrgIdCheck"}
      a.GetAccount = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.History = []string{"Admin", "AccountOwner"}
      a.GetAccountTransactionHistory = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
      a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
      a.GetSubTransactionsById = []string{"Admin", "TransactionInvoker"}
      a.GetSubTransactionsByIdWithFilters = []string{"Admin",
"TransactionInvoker"}
      a.GetAccountBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.GetAccountOnHoldBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.GetOnHoldIds = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
      a.GetAccountsByUser = []string{"Admin", "OrgAdminOrgIdCheck",
"MultipleAccountOwner"}

      as.Get = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
      as.ActivateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
      as.SuspendAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
      as.DeleteAccount = []string{"Admin", "OrgAdminOrgIdCheck"}

      r.GetAccountsByRole = []string{"Admin"}
      r.GetUsersByRole = []string{"Admin"}
      r.GetOrgAccountsByRole = []string{"Admin", "OrgAdminOrgIdCheck"}
      r.GetOrgUsersByRole = []string{"Admin", "OrgAdminOrgIdCheck"}

      tc.InitializeExchangePoolUser = []string{"Admin"}
      tc.AddConversionRate = []string{"Admin"}
      tc.UpdateConversionRate = []string{"Admin"}
      tc.GetConversionRate = []string{"Admin", "OrgAdmin", "AnyAccountOwner"}
      tc.GetConversionRateHistory = []string{"Admin", "OrgAdmin",
"AnyAccountOwner"}
      tc.TokenConversion = []string{"Admin", "AnyAccountOwner"}
      tc.GetExchangePoolUser = []string{"Admin"}


Ctx.Auth.CheckAuthorization(classFuncName string, args ...string) (bool,
error)
```

Parameters:

- `classFuncName string` – The map value between the receivers and methods as described in the `oChainUtil.go` file.

Copyright © 2025, Oracle and/or its affiliates.

- `args` – A variable argument, where `args[0]` is the constant `TOKEN` and `args[1]` is the `account_id` argument, if required.

Returns:

- A Boolean response and an error message if an error is encountered.

`AddOrgAdmin`
This method adds a user as an `Org Admin` of the organization.

```
Ctx.Admin.AddOrgAdmin(org_id, user_id) (interface{}, error)
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`RemoveOrgAdmin`
This method removes a user as an `Org Admin` of an organization.

```
Ctx.Admin.RemoveOrgAdmin(org_id, user_id) (interface{}, error)
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
    "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetOrgAdmins`
This method returns a list of all users who are an `Org Admin` of an organization.

```
Ctx.Admin.GetAllOrgAdmins() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a JSON list that includes `OrgId` and `UserId` objects.

Return Value Example:

```
{
    "admins": [
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin1"
        },
        {
            "OrgId": "Org1MSP",
            "UserId": "orgadmin2"
        }
    ]
}
```

`AddTokenAuditor`
This method adds a user as a `Token Auditor` of the chaincode.

```
t.Ctx.Admin.AddTokenAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "result": {
        "txid":
"f0888dd52f39dfa669275cc8f35d0b47b37b8407d384493d16970fcbb377f937",
        "payload": {
            "msg": "Successfully added Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 268
    }
}
```

RemoveTokenAuditor
This method removes a user as a `Token Auditor` of the chaincode.

```
t.Ctx.Admin.RemoveTokenAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"2f01f48eceaf2dff175f98b96a5bdd22c949f48fc5683ce86d6141cc4892aee3",
        "payload": {
            "msg": "Successfully removed Token Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 270
    }
}
```

GetTokenAuditors
This method returns all `Token Auditors` of the chaincode.

```
t.Ctx.Admin.GetAllTokenAuditors()
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"18eaa81b04d43db64f28287bab1cf6609e2a1d8ff84852ff73849ddb9a9dfba1",
        "payload": {
            "auditors": [
                {
                    "OrgId": "CB",
                    "UserId": "auditor_user_cb"
                }
            ]
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 269
    }
}
```

`AddOrgAuditor`
This method adds a user as a `Org Auditor` of the chaincode.

```
t.Ctx.Admin.AddOrgAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"3d5ec46003c68c6208d43c82894bd6da5c0b763339cc5212e09b71d39d0d80e2",
        "payload": {
            "msg": "Successfully added Org Auditor (Org_Id: CB, User_Id: cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 271
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
}
```

RemoveOrgAuditor
This method removes a user as a `Org Auditor` of the chaincode.

```
t.Ctx.Admin.RemoveOrgAuditor(org_id, user_id)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"713a120641afbc4dfaeac73b82c9fd51df6fcfd7d4d9a82553d3c487bf11f530",
        "payload": {
            "msg": "Successfully removed Org Auditor (Org_Id: CB, User_Id:
cb)"
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 273
    }
}
```

GetOrgAuditors
This method returns all `Org Auditors` of the chaincode.

```
t.Ctx.Admin.GetAllOrgAuditors()
```

Return Value Example:

```
{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"07656bf434616d7a3d7fe4fb81dc80c8cc608991648adfd9f6f2f2b9f6ddf468",
        "payload": {
            "auditors": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
                {
                    "OrgId": "CB",
                    "UserId": "cb"
                },
                {
                    "OrgId": "CB",
                    "UserId": "issuer_user_cb"
                }
            ]
        },
        "encode": "JSON",
        "sourceURL": "cb-oabcs1-bom.blockchain.ocp.example.com:20010",
        "blockNumber": 272
    }
}
```

Methods for Token Configuration Management

Save
This method creates a token and saves its properties in the state database.

```
Ctx.Token.Save(args ...interface{}) (interface{}, error)
```

Parameters:

- `token_id: string` – The ID of the token to return.
- `args` – A variable argument, where the first argument `args[0]` is a reference of the token `struct` data of the required type to add to the ledger.

Returns:

- On success, an `interface{}` object with details about the token that was saved to the state database. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"roles"],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

Update
This method updates token properties. After a token asset is created, you can update only the `token_desc` value and its custom properties.

```
Ctx.Token.Update(args ...interface{}) (interface{}, error)
```

Parameters:

- An asset that contains a reference to the token `struct` data of required type to update in the ledger.

Returns:

- On success, a promise message with token details. On error, a rejection with an error message.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "Digital Currency equiv of dollar",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"roles"],
    "Roles": {
        "minter_role_name": "minter"
    },
    "Mintable": {
        "Max_mint_quantity": 1000
    },
    "Divisible": {
        "Decimal": 2
    },
    "Currency_name": "",
    "Token_to_currency_ratio": 1
}
```

GetTokenDecimals
This method returns the number of decimal places available for a fractional token. If the `divisible` behavior is not specified, then the default value is 0.

```
Ctx.Token.GetTokenDecimals(token_id string) (int, error)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- none

Returns:

- On success, the decimal places of the token, in the number data type. On error, a non-nil error object containing an error message.

Return Value Example:

```
1
```

Get
This method returns a token object if it is present in the state database. This method can be called only by a `Token Admin` of the token chaincode.

```
Ctx.Get(Id string, result ...interface{}) (interface{}, error)
```

Parameters:

- `token_id: string` – The ID of the token to return.
- `result` – A variable argument, where the first argument `result[0]` is a reference of an empty `Token` object of the required type.

Returns:

- On success, a map with the token asset data. The variable argument `result[0]` contains the token data. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Token_id": "digiCurr101",
    "Token_name": "digicur",
    "Token_desc": "Digital Currency equiv of dollar",
    "Token_type": "fungible",
    "Behavior": ["divisible", "mintable", "transferable", "burnable",
"holdable", "roles"],
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter",
        "notary_role_name": "notary"
    },
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "Divisible": {
        "Decimal": 1
    },
    "Token_to_currency_ratio": 1,
    "Currency_representation": "DOLLAR"
}
```


Copyright © 2025, Oracle and/or its affiliates.

History
This method returns the token history for a specified token ID.

```
Ctx.Token.History(tokenId string) (interface{}, error)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON array that represents the token history.

Return Value Example:

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2023-09-01T16:46:33Z",
        "TxId":
"12333b8a4f63aa9b3a34072efcbd7df546c6d1e7d82a7a9596e899383656d6f7",
        "Value": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "Currency_name1": "",
            "Divisible": {
                "Decimal": 2
            },
            "Mintable": {
                "Max_mint_quantity": 1000
            },
            "Roles": {
                "minter_role_name": "minter"
            },
            "Token_desc": "updated description",
            "Token_id": "token",
            "Token_name": "fiatmoneytok",
            "Token_to_currency_ratio": 0,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-09-01T16:04:25Z",
        "TxId":
"99702e2dad7554a5ee4716a0d01d3e394cbce39bea8bade265d8911f30ebad0b",
        "Value": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "Currency_name1": "",
            "Divisible": {
                "Decimal": 2
            },
            "Mintable": {
                "Max_mint_quantity": 1000
            },
            "Roles": {
                "minter_role_name": "minter"
            },
            "Token_desc": "",
            "Token_id": "token",
            "Token_name": "fiatmoneytok",
            "Token_to_currency_ratio": 0,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    }
]
```

GetAllTokens
This method returns all of the token assets that are saved in the state database. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Token.GetAllTokens()  (interface{}, error)
```

Parameters:

- none

Returns:

- On success, an array of a map of all token assets. On error, a non-nil error object containing an error message.

Return Value Example:

```
"payload": [
    {
        "key": "t1",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "Currency_name value",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "Token_desc value",
            "Token_id": "t1",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 999,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    }
]
```

`GetTokensByName`
This method returns all of the token assets with the specified name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Token.GetTokensByName(token_name string) (interface{}, error)
```

Parameters:

- `token_name string` – The name of the token, which corresponds to the `Token_name` property of the model. The value is the class name of the token.

Returns:

- It returns an array of a map of all of the token assets of the specified name.

Return Value Example:

```
"payload": [
    {
        "key": "t1",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "Currency_name value",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "Token_desc value",
            "Token_id": "t1",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 999,
            "Token_type": "fungible",
            "Token_unit": "fractional"
        }
    },
    {
        "key": "obp2",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "divisible",
                "mintable",
                "transferable",
                "holdable",
                "burnable",
                "roles"
            ],
            "Currency_name": "",
            "Divisible": {
                "Decimal": 8
            },
            "Mintable": {
                "Max_mint_quantity": 10000
            },
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter",
                "notary_role_name": "notary"
            },
            "Token_desc": "",
            "Token_id": "obp2",
            "Token_name": "obptok",
            "Token_to_currency_ratio": 0,
            "Token_type": "fungible",
            "Token_unit": "fractional"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    }
]
```

IsTokenType
This method tests whether a token asset exists for a specified token ID.

```
Ctx.Model.IsTokenType(token_id: string) error
```

Parameters:

- `token_id: string` – The ID of the token to check.

Returns:

- If a token asset exists with the specified ID, a nil error. Otherwise, a non-nil error object containing an error message.

Return Value Example:

```
nil
```

GetByRange
This method calls the fabric `getStateByRange` method internally. Even though any asset with the given ID is returned from the ledger, this method casts the asset into the caller Asset type.

```
Ctx.Token.GetByRange(startId string, endId string, asset ...interface{})
([]map[string]interface{}, error)
```

Parameters:

- `startId: string` – The starting key of the range. This key is included in the range.
- `endId: string` – The end key of the range. This key is excluded from the range.
- `asset[0]` – An empty slice of the token of the required type. If the method runs successfully, this contains the requested result.

Returns:

- On success, a slice of maps containing the token asset details for tokens where the `token_id` value is in the specified range. On error, a non-nil error object containing an error message.

Return Value Example:

```
[{
    "Key":
"oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9121ff95b0526bc05
6c4b8974",
    "Record": {
        "AccountId":
"oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9121ff95b0526bc05
6c4b8974",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "AssetType": "oaccount",
            "Balance": 99,
            "BalanceOnHold": 1,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "t1",
            "TokenName": "loyaltok123",
            "UserId": "u1"
        }
    }, {
        "Key":
    "oaccount~loyaltok123~ac30c5ca924a2c7def61acf596d91e0cca70bc8cd233179df4efb279
    1b56336b",
        "Record": {
            "AccountId":
    "oaccount~loyaltok123~ac30c5ca924a2c7def61acf596d91e0cca70bc8cd233179df4efb279
    1b56336b",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "t1",
            "TokenName": "loyaltok123",
            "UserId": "u2"
        }
    }, {
        "Key":
    "oaccount~loyaltok123~aef96c40d99e09ef17f9bdda7038e8fbe829a327bae2b4d8d9fcf752
    190f3ff0",
        "Record": {
            "AccountId":
    "oaccount~loyaltok123~aef96c40d99e09ef17f9bdda7038e8fbe829a327bae2b4d8d9fcf752
    190f3ff0",
            "AssetType": "oaccount",
            "Balance": 0,
            "BapAccountVersion": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "OrgId": "Org1MSP",
            "TokenId": "t1",
            "TokenName": "loyaltok123",
            "UserId": "admin"
        }
    }, {
        "Key": "oadmin~Org1MSP~admin",
        "Record": {
            "AssetType": "oadmin",
            "Key": "oadmin~Org1MSP~admin",
            "OrgId": "Org1MSP",
            "UserId": "admin"
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
}, {
    "Key": "ohold~loyaltok123~t1~op1",
    "Record": {
        "AssetType": "ohold",
        "FromAccountId":
"oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9121ff95b0526bc05
6c4b8974",
        "HoldingId": "ohold~loyaltok123~t1~op1",
        "NotaryAccountId":
"oaccount~loyaltok123~ac30c5ca924a2c7def61acf596d91e0cca70bc8cd233179df4efb279
1b56336b",
        "OperationId": "op1",
        "Quantity": 1,
        "TimeToExpiration": "0",
        "ToAccountId":
"oaccount~loyaltok123~aef96c40d99e09ef17f9bdda7038e8fbe829a327bae2b4d8d9fcf752
190f3ff0",
        "TokenId": "t1",
        "TokenName": "loyaltok123"
    }
}, {
    "Key": "ometadata~loyaltok123~t1",
    "Record": {
        "AssetType": "ometadata",
        "Metadata_id": "ometadata~loyaltok123~t1",
        "Token_id": "t1",
        "Token_name": "loyaltok123",
        "Total_minted_amount": 100,
        "Total_supply": 100
    }
}, {
    "Key":
"orole~t1~minter~oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9
121ff95b0526bc056c4b8974",
    "Record": {
        "AccountID":
"oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9121ff95b0526bc05
6c4b8974",
        "AssetType": "orole",
        "Key":
"orole~t1~minter~oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9
121ff95b0526bc056c4b8974",
        "RoleName": "minter",
        "TokenId": "t1"
    }
}, {
    "Key":
"otransaction~4a774f6493f6521cab9eda96822cb3bb4103c0738ee2dbb9a193b868ace36fa5
",
    "Record": {
        "Amount": 100,
        "AssetType": "otransaction",
        "FromAccountId": "",
        "HoldingId": "",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "NumberOfSubTransactions": 0,
        "Timestamp": "2021-08-25T23:04:42+05:30",
        "ToAccountId":
"oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9121ff95b0526bc05
6c4b8974",
        "TokenId": "t1",
        "TransactionId":
"otransaction~4a774f6493f6521cab9eda96822cb3bb4103c0738ee2dbb9a193b868ace36fa5
",
        "TransactionType": "MINT"
    }
}, {
    "Key":
"otransaction~69f3cefbcb64b73f01a0eadff87169f456873ccebe61ca8da3eef3f465f0c129
",
    "Record": {
        "Amount": 1,
        "AssetType": "otransaction",
        "FromAccountId":
"oaccount~loyaltok123~a4bd3d8abfb1708198971311df77bb527233bcf9121ff95b0526bc05
6c4b8974",
        "HoldingId": "ohold~loyaltok123~t1~op1",
        "NumberOfSubTransactions": 0,
        "Timestamp": "2021-08-25T23:06:13+05:30",
        "ToAccountId":
"oaccount~loyaltok123~aef96c40d99e09ef17f9bdda7038e8fbe829a327bae2b4d8d9fcf752
190f3ff0",
        "TokenId": "t1",
        "TransactionId":
"otransaction~69f3cefbcb64b73f01a0eadff87169f456873ccebe61ca8da3eef3f465f0c129
",
        "TransactionType": "ONHOLD"
    }
}, {
    "Key": "t1",
    "Record": {
        "AssetType": "otoken",
        "Behavior": ["divisible", "mintable", "transferable", "burnable",
"holdable", "roles"],
        "Currency_Name": "a",
        "Divisible": {
            "Decimal": 2
        },
        "Effective_From_Date": "2020-09-09T00:00:00Z",
        "Mintable": {
            "Max_mint_quantity": 10000
        },
        "Roles": {
            "minter_role_name": "minter"
        },
        "Token_To_Currency_Ratio": 1,
        "Token_desc": "",
        "Token_id": "t1",
        "Token_name": "loyaltok123",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "Token_type": "fungible"
    }
}]
```

Methods for Account Management

`GenerateAccountId`
This method returns an account ID, which is an alphanumeric set of characters, prefixed with `oaccount~<token asset name>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization and the unique token ID (`token_id`).

```
Ctx.Account.GenerateAccountId(token_id string, org_id string, user_id string)
(string, error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, the generated account ID. On error, a rejection with an error message.

Return Value Example:

```
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f
```

`CreateAccount`
This method creates an account for a specified user and token. Every user who has tokens at any point must have an account. Accounts track a user's balance, on-hold balance, and transaction history. An account ID is an alphanumeric set of characters, prefixed with `oaccount~<token asset name>~` and followed by a hash of the user name or email ID (`user_id`) of the instance owner or the user who is logged in to the instance, the membership service provider ID (`org_id`) of the user in the current network organization. This method can be called only by the `Token Admin` of the chaincode.

```
t.Ctx.Account.CreateAccount(org_id string, user_id string, token_type string)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.
- `token_type: string` – The type of the token, which must be `fungible`.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, the account object that was created. On error, a rejection with an error message.

Return Value Example:

```
{
 "AssetType":"oaccount",
"AccountId":"oaccount~a73085a385bc96c4a45aa2dff032e7dede82c0664dee5f396b7c5854
eeafd4bd",
   "UserId":"user1",
   "OrgId":"Org1MSP",
   "BapAccountVersion": 0,
   "AccountType":"fungible",
   "TokenId":"",
   "TokenName":"",
   "Balance":0,
   "BalanceOnHold":0,
   "OnHoldBurnBalance":0
}
```

`AssociateToken`
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode.

```
t.Ctx.Account.AssociateToken(account_id, token_id)
```

Parameters:

- `account_id string` – The ID of the account.
- `token_id string` – The ID of the token.

Returns:

- On success, a JSON object of the updated account.

Return Value Example:

```
{
"AssetType":"oaccount",
"AccountId":"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bd
bebf48eb",
"BapAccountVersion": 0,
"UserId":"admin",
"OrgId":"Org1MSP",
"AccountType":"fungible",
"TokenId":"token1",
"TokenName":"loyaltok",
"Balance":0,
"BalanceOnHold":0,
"OnHoldBurnBalance":0
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAccountWithStatus
This method returns account details for a specified account, including account status.

```
Ctx.Account.GetAccountWithStatus(account_id string) (interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, the requested account details. On error, a rejection with an error message.

Return Value Example:

```
{
  "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "AssetType": "oaccount",
  "Balance": 95,
  "BalanceOnHold": 0,
  "OnHoldBurnBalance": 0,
  "BapAccountVersion": 8,
  "OrgId": "appdev",
  "Status": "active",
  "TokenId": "obp1",
  "TokenName": "obptok",
  "TokenType": "fungible",
  "UserId": "idcqa"
}
```

GetAccount
This method returns account details for a specified account.

```
Ctx.Account.GetAccount(account_id string) (Account, error)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, the requested account details. On error, a rejection with an error message.

Return Value Example:

```
{
    "AssetType": "oaccount",
    "BapAccountVersion": 0,
    "AccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "UserId": "user1",
    "OrgId": "Org1MSP",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "TokenId": "digiCurr101",
    "TokenName": "digicur",
    "Balance": 0,
    "BalanceOnHold": 0,
    "OnHoldBurnBalance": 0
}
```

GetAccountHistory
This method returns an array of the account history details for a specified account.

```
Ctx.Account.History(account_id string) ([]interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a `map[string]interface{}` array that contains the account history details. The account data is shown under the `Value` key in the map. On error, a non-nil error object containing an error message. The return value is the same as the `GetAccountHistory` method.

Return Value Example:

```
[
  {
      "IsDelete": "false",
      "Timestamp": "2023-08-28T19:31:15Z",
      "TxId":
"adde470a63860ec1013bd5c5987e8a506a48942a91b0f39fc8e561374042bd27",
      "Value": {
          "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
          "AssetType": "oaccount",
          "Balance": 100,
          "BalanceOnHold": 0,
          "OnHoldBurnBalance": 0,
          "BapAccountVersion": 1,
          "OrgId": "Org1MSP",
          "TokenId": "t1",
          "TokenName": "obptok",
          "TokenType": "fungible",
          "UserId": "idcqa"
      }
  },
  {
      "IsDelete": "false",
      "Timestamp": "2023-08-28T19:30:23Z",
      "TxId":
"8fbeda2ba60ba175091faae5ae369247775f2cba45c4d6d1ead6f0b05be84743",
      "Value": {
          "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "t1",
            "TokenName": "obptok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-08-28T19:29:54Z",
        "TxId":
"19bb296ae71709e91b097ba5d9ebd7f7522095880382fbf5913334a46a6026aa",
        "Value": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "Org1MSP",
            "TokenId": "",
            "TokenName": "",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    }
]
```


`GetAccountOnHoldBalance`
This method returns the on-hold balance for a specified account.

```
Ctx.Account.getAccountOnHoldBalance(account_id: string)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise object with the current on-hold balance and a success message. On error, a non-nil error object containing an error message.

Return Value Example:


```
{
   "holding_balance":0,
   "msg":"Total Holding Balance of Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
```


Copyright © 2025, Oracle and/or its affiliates.

```
22f (org_id: Org1MSP, user_id: user1) is 0"
}
```

GetAllAccounts
This method returns a list of all accounts. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.func (t *Controller) GetAllAccounts() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a JSON array that lists all accounts.

Return Value Example:

```
"payload": [
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 100,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 1,
            "OrgId": "appdev",
            "TokenId": "t1",
            "TokenName": "obptok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    }
]
```

GetUserByAccountId
This method returns the user details for a specified account.

```
Ctx.Account.GetUserByAccountById(account_id string) (interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a promise with a JSON object that includes three properties:

Copyright © 2025, Oracle and/or its affiliates.

  - `user_id` – The user name or email ID of the user.
  - `org_id` – The membership service provider (MSP) ID of the user in the current network organization.
  - `token_id` – The ID of the token.
- On error, a non-nil error object containing an error message.

Return Value Example:

```
{
   "org_id":"Org1MSP",
   "token_id":"digiCurr101",
   "user_id":"user1"
}
```

`GetAccountBalance`
This method returns the account balance for a specified account.

```
Ctx.GetAccountBalance(token_id string, org_id string, user_id string)
(interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, an interface with a message string and the current balance. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "msg": "Current Balance of +p2uaMTsU9D74l9XpHQ2c55ic/2gbO4NZITC4Zq4P8E=
is: 200",
    "user_balance": 200
}
```

`GetAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization.

```
Ctx.Account.GetAllOrgAccounts(org_id string) (interface{}, error)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "key":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
        "valueJson": {
            "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "appdev",
            "TokenId": "token",
            "TokenName": "fiatmoneytok",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "key":
"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850",
        "valueJson": {
            "AccountId":
"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850",
            "AssetType": "oaccount",
            "Balance": 0,
            "BalanceOnHold": 0,
            "OnHoldBurnBalance": 0,
            "BapAccountVersion": 0,
            "OrgId": "appdev",
            "TokenId": "token",
            "TokenName": "fiatmoneytok",
            "TokenType": "fungible",
            "UserId": "example_minter"
        }
    }
]
```

Methods for Role Management

`AddRoleMember`
This method adds a role to a specified user and token.

```
Ctx.Token.AddRoleMember(role string, account_id string, tokenAsset
interface{}) (interface{}, error)
```

Parameters:

- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name`

Copyright © 2025, Oracle and/or its affiliates.

properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.

- `account_id: number` – The account ID to add the role to.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, it returns a map with a success message indicating the addition of the role to the account.. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
   "msg":"Successfully added role minter to
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"
}
```

`RemoveRoleMember`
This method removes a role from a specified user and token.

```
Ctx.Token.RemoveRoleMember(role string, account_id string, tokenAsset
interface{}) (interface{}, error)
```

Parameters:

- `role: string` – The name of the role to remove from to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.
- `account_id: number` – The account ID to remove the role from.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, it returns a map with a success message indicating the removal of the role from the account.. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
  "msg":"successfully removed member_id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) from role minter"
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAccountsByRole
This method returns a list of all accounts for a specified role and token.

```
Ctx.Role.GetAccountsByRole(token_id string, user_role string) (interface{},
error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"accounts":
["oaccount~obptok~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f"]}
```

GetUsersByRole
This method returns a list of all users for a specified role and token.

```
Ctx.Role.GetUsersByRole(token_id string, user_role string) (interface{},
error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of user objects. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "Users": [
        {
            "token_id":"digiCurr101",
            "user_id": "user1",
            "org_id": "Org1MSP"
        }
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

IsInRole
This method indicates whether a user and token has a specified role.

```
Ctx.Token.IsInRole(role string, account_id string, tokenAsset interface{})
(bool, error)
```

Parameters:

- `role: string` – The name of the role to check.
- `account_id: number` – The account ID to check.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, it returns a map with a success message indicating the removal of the role from the account.. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "result": false
}
```

GetOrgAccountsByRole
This method returns information about all accounts that have a specified role in a specified organization.

```
Ctx.Role.GetOrgAccountsByRole(token_id string, user_role string, org_id
string) (interface{}, error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts with the specified role in the specified organization.

Return Value Example:

```
{
    "accounts": [

"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",

"oaccount~9c650574af9025a6106c8d12a801b079eda9ae2e3399fc2fbd5bd683d738a850"
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetOrgUsersByRole`
This method returns information about all users that have a specified role in a specified organization.

```
Ctx.Role.GetOrgUsersByRole(token_id string, user_role string, org_id string)
(interface{}, error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all users with the specified role in the specified organization.

Return Value Example:

```
{
    "Users": [
        {
            "org_id": "Org1MSP",
            "token_id": "token",
            "user_id": "admin"
        },
        {
            "org_id": "Org1MSP",
            "token_id": "token",
            "user_id": "orgAdmin"
        }
    ]
}
```

`RoleCheck`
This method checks if the provided account ID is a member of any role.

```
Ctx.Token.RoleCheck(account_id string, tokenAsset interface{}) (bool, error)
```

Parameters:

- `account_id: string` – The account ID to check.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- If the specified account has any role, a success message and the Boolean value `true`. Otherwise, the Boolean value `false`. On error, a non-nil error object containing an error message.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{ result: true }
```

Methods for Transaction History Management

`GetAccountTransactionHistory`
This method returns an array of the transaction history details for a specified account.

```
Ctx.Account.GetAccountTransactionHistory(account_id string) (interface{},
error)
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- The return value is the same as the `GetAccountTransactionHistory` method.
- On success, an array of JSON account transaction objects.
- On error, a non-nil error object containing an error message.

Return Value Example:

```
[
  {
      "NumberOfSubTransactions": 2,
      "balance": 160,
      "onhold_balance": 0,
      "timestamp": "2023-09-06T06:51:48Z",
      "token_id": "t1",
      "transacted_amount": 20,
      "transaction_id":
"otransaction~bd3e8d7d0bcdbed0469a2fccfe95f7ebbeb1987d8385bccf5c84bf80251e748c
",
      "transaction_type": "BULKTRANSFER"
  },
  {
      "balance": 180,
      "onhold_balance": 0,
      "timestamp": "2023-09-06T06:47:14Z",
      "token_id": "t1",
      "transacted_account":
"oaccount~692a7465c01e36b694cb8ae86e6c6584240aa1f865fde54f95f32429eadd4097",
      "transacted_amount": 10,
      "transaction_id":
"otransaction~250996f1df6a36a1b647f522efcaaf48fd70452d711c247fc4cd475b8e752b08
",
      "transaction_type": "DEBIT"
  },
  {
      "balance": 190,
      "onhold_balance": 0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "timestamp": "2023-09-06T06:47:08Z",
      "token_id": "t1",
      "transacted_account":
"oaccount~bb5a0b57d895327c8a8cd1f267310cbf3ae542bc854fab8188b5083a969d72fb",
      "transacted_amount": 10,
      "transaction_id":
"otransaction~664325a25ae6b19b23693c66f83811184e0a78fabb49122359a2dbf209f32976
",
      "transaction_type": "DEBIT"
  },
  {
      "balance": 200,
      "onhold_balance": 0,
      "timestamp": "2023-09-06T06:46:46Z",
      "token_id": "t1",
      "transacted_account":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
      "transacted_amount": 100,
      "transaction_id":
"otransaction~7f49564b1eb61d4c8be0ef61cd5e635b533ca533907944e4ec500f390237fd6b
",
      "transaction_type": "MINT"
  },
  {
      "balance": 100,
      "onhold_balance": 0,
      "timestamp": "2023-08-28T19:31:15Z",
      "token_id": "t1",
      "transacted_account":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
      "transacted_amount": 100,
      "transaction_id":
"otransaction~adde470a63860ec1013bd5c5987e8a506a48942a91b0f39fc8e561374042bd27
",
      "transaction_type": "MINT"
  }
]
```


`GetAccountTransactionHistoryWithFilters`
This method returns an array of the transaction history details for a specified transaction. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
t.Ctx.Account.GetAccountTransactionHistoryWithFilters (transaction_id:
string, filters?: SubTransactionFilters)
```

Parameters:

- `Transaction_id: string` – The ID of the transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Copyright © 2025, Oracle and/or its affiliates.

Example:

```
ochain invoke GetAccountTransactionHistoryWithFilters 'token1' 'appbuilder12'
'user_minter'
'{"PageSize":10,"Bookmark":"1","StartTime":"2022-01-25T17:41:42Z","EndTime":"202
2-01-25T17:59:10Z"}'

[
    {
        "balance": 90,
        "onhold_balance": 0,
        "timestamp": "2022-04-20T19:43:36Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~7a4d67118e623a876b77c67e76b819269a8d4a509aece5d2263fb274a9beb3b8",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~dd9986d3686e52264935558e42026fbf8a9af48b06a3256a58b453f5ada4e636
",
        "transaction_type": "DEBIT"
    },
    {
        "balance": 95,
        "onhold_balance": 0,
        "timestamp": "2022-04-20T19:43:22Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~0642308fc4c514c257ebf04326c63f990e2531bfd59d0b952056094da61e04ab",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~5e53424de3d691cf6b2a55ea3dc478c555d8784111c11847e594194d6c2e7755
",
        "transaction_type": "DEBIT"
    },
    {
        "balance": 100,
        "onhold_balance": 0,
        "timestamp": "2022-04-20T19:42:54Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~b63935592a702d30bedb87ae97b9b1ba7d0f346716adc4f5a4192220bf410d4e",
        "transacted_amount": 100,
        "transaction_id":
"otransaction~94c467825ce9f66cc69958d38b169022a69eebc66b75b7d6e0b0585af2c3c228
",
        "transaction_type": "MINT"
    }
]
```

GetSubTransactionsById

This method returns an array of the transaction history details for a specified transaction.

```
t.Ctx.Account.GetSubTransactionsById(transaction_id string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `transaction_id: string` – The ID of the transaction.

Example:

```
ochain invoke GetSubTransactionsById
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'

[
    {
        "balance": 80,
        "onhold_balance": 0,
        "timestamp": "2022-04-21T05:02:33Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~7a4d67118e623a876b77c67e76b819269a8d4a509aece5d2263fb274a9beb3b8",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~33de5d63058d5e9abc011bc850878dfb7ac3080495729aed345c45b2f21735fa
~c81e728d9d4c2f636f067f89cc14862c",
        "transaction_type": "DEBIT"
    },
    {
        "balance": 85,
        "onhold_balance": 0,
        "timestamp": "2022-04-21T05:02:33Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~0642308fc4c514c257ebf04326c63f990e2531bfd59d0b952056094da61e04ab",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~33de5d63058d5e9abc011bc850878dfb7ac3080495729aed345c45b2f21735fa
~c4ca4238a0b923820dcc509a6f75849b",
        "transaction_type": "DEBIT"
    }
]
```

`GetSubTransactionsByIdWithFilters`

This method returns an array of the transaction history details for a specified transaction.

```
t.Ctx.Account.GetSubTransactionsByIdWithFilters(transaction_id string,
filters ...SubTransactionFilters)
```

Parameters:

- `transaction_id: string` – The ID of the transaction.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
ochain invoke GetSubTransactionsByIdWithFilters
'otransaction~21972b4d206bd52ea77924efb259c67217edb23b4386580d1bee696f6f864b9b'
'{"PageSize":10,"Bookmark":"1"}'

[
    {
        "balance": 80,
        "onhold_balance": 0,
        "timestamp": "2022-04-21T05:02:33Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~7a4d67118e623a876b77c67e76b819269a8d4a509aece5d2263fb274a9beb3b8",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~33de5d63058d5e9abc011bc850878dfb7ac3080495729aed345c45b2f21735fa
~c81e728d9d4c2f636f067f89cc14862c",
        "transaction_type": "DEBIT"
    },
    {
        "balance": 85,
        "onhold_balance": 0,
        "timestamp": "2022-04-21T05:02:33Z",
        "token_id": "tokenId",
        "transacted_account":
"oaccount~0642308fc4c514c257ebf04326c63f990e2531bfd59d0b952056094da61e04ab",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~33de5d63058d5e9abc011bc850878dfb7ac3080495729aed345c45b2f21735fa
~c4ca4238a0b923820dcc509a6f75849b",
        "transaction_type": "DEBIT"
    }
]
```

`GetTransactionById`
This method returns the history of a `Transaction` asset.

```
t.Ctx.Transaction.GetTransactionById(transaction_id string)
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Return Value Example:

```
{
    "history": [
        {
            "IsDelete": "false",
            "Timestamp": "2021-08-16 20:19:05.028 +0530 IST",
            "TxId":
"67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220",
            "Value": {
                "Amount": 3,
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "AssetType": "otransaction",
                "FromAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "HoldingId": "ohold~digicur~digiCurr101~op2",
                "NumberOfSubTransactions": 0,
                "Timestamp": "2021-08-16T20:19:05+05:30",
                "ToAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
                "TokenId": "digiCurr101",
                "TransactionId":
"otransaction~67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220
",
                "TransactionType": "RELEASEHOLD"
            }
        }
    ],
    "transaction_id":
"otransaction~67042154a6853011d111b13f73943f06d2a6ae3cfb9a84cb104482c359eb2220
"
}
```

`DeleteHistoricalTransactions`
This method deletes older transactions from the state database.

```
func (t *Controller) DeleteHistoricalTransactions(timestamp string)
(interface{}, error)
```

Parameters:

- `time_to_expiration: Date` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted..

Return Value Example:

```
"payload": {
    "msg": "Successfuly deleted transaction older than
date:2021-08-18T05:43:30Z",
    "transactions": [

"otransaction~57d81f681aa215bb73d6c017d16be8b283d3fcb50051c85891a97d1d407fc342
"
    ]
}
```

Methods for Token Behavior Management - Mintable Behavior

`Mint`
This method mints tokens, which are then owned by the caller of the method. The caller must have an account and the minter role. The number of tokens that can be minted is limited by the `max_mint_quantity` property of `mintable` behavior in the specification file. If the `max_mint_quantity` property is not specified, an unlimited number of tokens can be minted. The quantity must be within the decimal values specified by the `decimal` parameter of the

Copyright © 2025, Oracle and/or its affiliates.

`divisible` behavior in the specification file. This method can be called only by the `AccountOwner` of the account with the minter role.

```
Ctx.Token.Mint(quantity float64, tokenAsset interface{}) (interface{}, error)
```

Parameters:

- `quantity: number` – The number of tokens to mint.
- `tokenAsset` – The reference to the token asset to mint.

Returns:

- On success, a success message. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
  "msg":"Successfully minted 1000 tokens to Account Id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin)"
}
```

`GetTotalMintedTokens`
This method returns the total number of tokens minted.

```
Ctx.Token.GetTotalMintedTokens(tokenAsset interface{})
(map[string]interface{}, error)
```

Parameters:

- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message and a map of total minted tokens in the number data type. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"msg":"total minted amount for token with id digiCurr101 is 0","quantity":0}
```

`GetNetTokens`
This method returns the net quantity of tokens that are available in the system for a specified token. The net tokens are the amount of tokens remaining after tokens are burned. In equation form: net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens.

```
Ctx.Token.GetNetTokens(tokenAsset interface{}) (map[string]interface{}, error)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message and a map of the net quantity of tokens in the number data type. On error, an error message.

Return Value Example:

```
{"msg":"net minted amount for token with id digiCurr101 is 0","quantity":0}
```

`GetMaxMintQuantity`
This method returns the maximum mintable quantity for a token. If the `max_mint_quantity` behavior is not specified, then the default value is 0, which allows any number of tokens to be minted.

```
Ctx.Token.GetMaxMintQuantity(token_id string) (float64, error)
```

Parameters:

- `token_id: string` – The token ID to check.

Returns:

- On success, the maximum mintable quantity of the token, in the number data type. On error, a non-nil error object containing an error message.

Return Value Example:

```
20000
```

Methods for Token Behavior Management - Transferable Behavior

`Transfer`
This method transfers tokens from the caller to a specified account. The caller of the method must have an account. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. This method can be called only by the `AccountOwner` of the account.

```
Ctx.Token.Transfer(to_account_id string, quantity float64, tokenAsset
interface{}) (interface{}, error)
```

Parameters:

- `to_account_id: string` – The account ID to receive the tokens.
- `quantity: number` – The total number of tokens to transfer.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a success message. On error, a non-nil error object containing an error message. The return value is the same as the `TransferTokens` method.

Return Value Example:

```
{     "msg":"Successfully transferred 50 tokens from account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP,  User-Id: admin) to account id:
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)"
}
```

`BulkTransfer`
This method transfers tokens in bulk from the caller account to the accounts that are specified in the `flow` object. The caller of this method must have an account already created.

```
Ctx.Token.BulkTransfer(flow []map[string]interface{}, tokenAsset interface{})
(interface{}, error)
```

Parameters:

- `flow: object[]` – An array of JSON objects specifying the receiver details and quantity. The transfer quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file. For example:

  ```
  [{
      "to_org_id": "Org1MSP",
      "to_user_id": "user1",
      "quantity": 10
  }, {
      "to_org_id": "Org1MSP",
      "to_user_id": "user2",
      "quantity": 10
  }]
  ```

- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message that includes the number of tokens transferred. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "from_account_id":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "msg": "Successfully transferred 2 tokens from Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (Org-Id: Org1MSP, User-Id: user1)",
    "sub_transactions": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e"
        },
        {
            "amount": 1,
            "to_account_id":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df"
        }
    ]
}
```

Methods for Token Behavior Management - Holdable Behavior

`Hold`

This method creates a hold on behalf of the owner of the tokens with the `to_account_id` account. A notary account is specified, which is responsible to either complete or release the hold. When the hold is created, the specified token balance from the payer is put on hold. A held balance cannot be transferred until the hold is either completed or released. The caller of this method must have an account already created.

```
Ctx.Token.Hold(operation_id string, to_account_id string, notary_account_id
string, quantity float64, TimeToExpiration string, tokenAsset)) (interface{},
error)
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_account_id: string` – The ID of the account to receive the tokens.
- `notary__account_id: string` – The ID of the notary account.
- `quantity: number` – The total number of tokens to put on hold.
- `time_to_expiration: date` – The duration until the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12`.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
 "msg": "account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
```


Copyright © 2025, Oracle and/or its affiliates.

```
2df (org_id : Org1MSP, user_id : user1) is successfully holding 10 tokens",
}
```

ExecuteHold
This method completes a hold on tokens, transferring the specified quantity of tokens previously on hold to the receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can be called only by the `AccountOwner` ID with the `notary` role.

```
Ctx.Token.ExecuteHold(operation_id string, quantity float64, tokenAsset
interface{}) (interface{}, error)
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity: number` – The total number of tokens to put on hold.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message. On error, a non-nil error object containing an error message.

Return Value Example:

```
{"msg":"Account Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1) has successfully executed '1'
tokens(digiCurr101) from the hold with Operation Id 'op1'"}
```

ReleaseHold
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `Account Owner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
Ctx.Token.ReleaseHold(operation_id string, tokenAsset interface{})
(interface{}, error)
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message. On error, a non-nil error object containing an error message.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg":"Successfully released '3' tokens from Operation Id 'op2' to Account
Id
oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f6
22f (org_id : Org1MSP, user_id : user1)"}
```

`GetOnHoldIds`
This method returns a list of all the holding IDs for a specified user and token.

```
Ctx.Account.GetOnHoldIDs(account_id string) (map[string]interface{}, error)
```

Parameters:

- `token_id` – The ID of the token.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.
- `user_id` – The user name or email ID of the user.

Returns:

- On success, a JSON object with the list of holding IDs. A holding ID is formed by concatenating the asset type (`ohold`), the token name, the token ID, and the operation ID.

Return Value Example:

```
{"holding_ids":["ohold~loyaltok123~t1~op1"],"msg":"Holding Ids are:
[ohold~loyaltok123~t1~op1]"}
```

`GetOnHoldDetailsWithOperationID`
This method returns the on-hold transaction details for a specified operation ID and token..

```
Ctx.Hold.GetOnHoldDetailsWithOperationID(token_id string, operation_id
string) (Hold, error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID that identifies the hold operation. Typically this ID is passed by the client application.

Returns:

- The return value is the same as the `GetOnHoldDetailsWithOperationId` method.
- On success, a promise object that includes the on-hold transaction details for the specified operation ID and token. The hold object includes the following properties:
  - `holding_id` – The holding ID of the transaction.
  - `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Copyright © 2025, Oracle and/or its affiliates.

  - `from_account_id` – The account ID of the current owner of the on-hold tokens.
  - `to_account_id` – The account ID of the receiver.
  - `notary_account_id` – The account ID of the notary.
  - `token_id: string` – The ID of the saved token.
  - `quantity` – The amount of tokens that are on hold for the holding ID.
  - `time_to_expiration` – The duration until the hold expires.
- On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "AssetType": "ohold",
    "HoldingId": "ohold~digicur~digiCurr101~op1",
    "OperationId": "op1",
    "TokenName": "digicur",
    "FromAccountId":
"oaccount~digicur~b4f45440aa2a7942db64443d047027e9d714d62cba5c3d546d64f368642f
622f",
    "ToAccountId":
"oaccount~digicur~38848e87296d67c8a90918f78cf55f9c9baab2cdc8c928535471aaa1210c
706e",
    "NotaryAccountId":
"oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f03761
52df",
    "TokenId": "digiCurr101",
    "Quantity": 2,
    "TimeToExpiration": "0"
}
```

`GetOnHoldBalanceWithOperationID`
This method returns the on-hold balance for a specified operation ID and token..

```
Ctx.Hold.GetOnHoldBalanceWithOperationID(token_id string, operation_id
string) (map[string]interface{}, error)
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID that identifies the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, the on-hold balance of the specified operation ID and token. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
    "holding_balance": 10,
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "msg": "Current Holding Balance of OperationId opr_121 for token
digiCurr101 is : 10"
}
```

Methods for Token Behavior Management - Burnable Behavior

`Burn`
This method deactivates, or burns, tokens from the transaction caller's account. The caller of this method must have an account and the burner role. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.

```
Ctx.Token.Burn(quantity float64 , tokenAsset interface{}) (interface{}, error)
```

Parameters:

- `quantity: number` – The total number of tokens to burn.
- `tokenAsset` – The `tokenAsset` argument contains the reference of the token data to operate on.

Returns:

- On success, a success message. On error, a non-nil error object containing an error message.

Return Value Example:

```
{
 "msg":"Successfully burned 10 tokens from account id:
oaccount~digicur~682bb71de419602af74e3f226345ae308445ca51010737900c1d85f037615
2df (Org-Id: Org1MSP, User-Id: admin)"
}
```

Go Methods for Token Conversion

Blockchain App Builder automatically generates methods that you can use to convert fungible tokens that use the Token Taxonomy Framework standard.

The token conversion methods include the concept of the **exchange pool**. The exchange pool account is funded by other token accounts. When you mint tokens, you can specify that a percentage of the minted tokens are transferred to the exchange pool account.

- Token Conversion Process
- Automatically Generated Token Conversion Methods
- Token Conversion SDK Methods

Token Conversion Process

A typical flow for converting tokens follows these steps:

1. Call the `InitializeExchangePoolUser` method to initialize the exchange pool user.
2. Call the `CreateExchangePoolAccounts` method to create exchange pool accounts. Create an exchange pool account for every type of fungible token that you want to convert from or convert to.

Copyright © 2025, Oracle and/or its affiliates.

3. Call the `AddConversionRate` method to set the conversion rate for each pair of tokens that you want to convert between.
4. Fund the exchange pool token accounts in one of the following ways:
   - Transfer tokens to the exchange pool token accounts using the standard transfer methods.
   - Call the `MintWithFundingExchangePoolToken` method when minting tokens, which can transfer a percentage of minted tokens to an exchange pool account.
5. Call the `TokenConversion` method to convert between two fungible tokens. A single user can convert tokens between two of their token accounts, or a pair of users can directly convert tokens from one account to another.
6. The exchange pool user can view the exchange pool account balances and account transactions.
   - Call the `GetAccount` method to view the balances of each of the exchange pool token accounts.
   - Call the `GetAccountTransactionHistory` and `GetAccountTransactionHistoryWithFilters` methods to view account transactions for each of the exchange pool token accounts.

Automatically Generated Token Conversion Methods

Blockchain App Builder automatically generates methods to convert between different types of fungible tokens. Controller methods must be public to be invokable. Public method names begin with an upper case character. Method names that begin with a lower case character are private.

- Base
- Digital Assets

Base

`InitializeExchangePoolUser`
This method initializes the exchange pool user, which is a one-time activity. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) InitializeExchangePoolUser(org_id string, user_id
string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.InitializeExchangePoolUser",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.InitializeExchangePoolUser(org_id, user_id)
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the exchange pool user.

Return Value Example:

```
{
    "AssetType": "oconversion",
    "ConvertorId":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "OrgId": "Org1MSP",
    "UserId": "exchangepooluser"
}
```

`CreateExchangePoolAccounts`
This method creates exchange pool token accounts for a given array of token IDs. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) CreateExchangePoolAccounts(token_ids []string)
(interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.InitializeExchangePoolUser",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    var tokens []interface{}
    for _, tokenId := range token_ids {
        token, err := t.getTokenObject(tokenId)
        if err != nil {
            return nil, fmt.Errorf("error in getting from_token asset
details. Error: %s", err)
        }
        tokens = append(tokens, token.Interface())
    }
    return t.Ctx.TokenConvertor.CreateExchangePoolAccounts(tokens)
}
```

Parameters:

- `token_ids: string []` – An array of token IDs. You can specify up to ten token IDs.

Returns:

- On success, a list of objects that includes details of the exchange pool accounts that were created.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "AccountId":
"oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034",
        "Status": "created",
        "TokenId": "USD"
    },
    {
        "AccountId":
"oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0",
        "Status": "created",
        "TokenId": "INR"
    }
]
```

`AddConversionRate`
This method adds a conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddConversionRate(from_token_id string, to_token_id
string, token_conversion_rate float64) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.AddConversionRate", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.AddConversionToken(from_token_id,
to_token_id, token_conversion_rate)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: float64` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 10
}
```

GetConversionRate
This method gets the current conversion rate for a pair of tokens. This method can be called by the `Token Admin` of the chaincode, any `Org Admin`, and by any token account owner.

```
func (t *Controller) GetConversionRate(from_token_id string, to_token_id
string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetConversionRate", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    conversionRateId, err :=
t.Ctx.TokenConversionRate.GetConversionRateId(from_token_id, to_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting conversationRateId. Error:
%s", err)
    }
    return t.Ctx.TokenConversionRate.Get(conversionRateId)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 10
}
```


Copyright © 2025, Oracle and/or its affiliates.

`UpdateConversionRate`
This method updates the current conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) UpdateConversionRate(from_token_id string, to_token_id
string, token_conversion_rate float64) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.UpdateConversionRate", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.UpdateTokenConversionRate(from_token_id,
to_token_id, token_conversion_rate)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: float64` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the updated conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 15
}
```

`MintWithFundingExchangePool`
This method mints tokens in the caller's account based on the specified token ID and quantity. A percentage of tokens from the minted quantity is then transferred to the exchange pool token account.

```
func (t *Controller) MintWithFundingExchangePool(token_id string,
token_quantity float64, percentage_token_to_exchangePool float64)
(interface{}, error) {
    token, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting from_token asset details.
Error: %s", err)
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    return
t.Ctx.TokenConvertor.MintWithFundingExchangePool(token.Interface(),
token_quantity, percentage_token_to_exchangePool)
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `token_quantity: float64` – The total number of tokens to mint.
- `percentage_token_to_exchange_pool: float64` – The percentage of minted tokens to transfer to the exchange pool token account.

Returns:

- On success, a message that indicates that minting and funding the exchange pool were successful.

Return Value Example:

```
{
    "msg": "successfully minted 100 tokens to AccountId:
'oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1'
(OrgId: Org1MSP, User-Id: admin) and Successfully transfered 80 tokens to
exchange pool account with AccountId:
'oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0'
(OrgId: Org1MSP, UserId: exchangepooluser) "
}
```

`TokenConversion`
This method converts tokens from the caller's account to the account specified by the `to_token_id`, `to_org_id`, and `to_user_id` values. This method can be called by the `Token Admin` of the chaincode and by any token account owner. An exchange pool user cannot call this method.

```
func (t *Controller) TokenConversion(from_token_id string, to_token_id
string, to_org_id string, to_user_id string, token_quantity float64)
(interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.TokenConversion", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    from_token, err := t.getTokenObject(from_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting from_token asset details.
Error: %s", err)
    }
    to_token, err := t.getTokenObject(to_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting to_token asset details.
error: %s", err)
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    return t.Ctx.TokenConvertor.TokenConversion(from_token.Interface(),
to_token.Interface(), to_org_id, to_user_id, token_quantity)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `to_org_id: string` – The membership service provider (MSP) ID of the user in the current organization to receive the tokens.
- `to_user_id: string` – The user name or email ID of the user to receive the tokens.
- `token_quantity: float64` – The total number of tokens to transfer.

Returns:

- On success, a message that indicates the token conversion was successful.

Return Value Example:

```
{
    "msg": "succesfully converted 5 of tokens with tokenId: [USD] from
AccountId:
'oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb'
(OrgId: Org1MSP, UserId: admin) to 75 of tokens with tokenId: [INR] to
AccountId:
'oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e'
( OrgId: Org1MSP, UserId: user ) as per the conversion rate of 15"
}
```

`GetConversionHistory`
This method returns the token conversion history for a specified token account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, and by the token account owner.

```
func (t *Controller) GetConversionHistory(token_id string, org_id string,
user_id string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetConversionHistory", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
    if err != nil {
        return nil, fmt.Errorf("error in generating the accoint_id. Error:
%s", err)
    }
    return t.Ctx.Account.GetTokenConversionHistory(account_id, org_id,
```


Copyright © 2025, Oracle and/or its affiliates.

```
user_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object with conversion history details.

Return Value Example:

```
[
    {
        "balance": 95,
        "conversion_rate": 15,
        "converted_amount": 75,
        "from_account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
        "from_token_id": "USD",
        "onhold_balance": 0,
        "timestamp": "2022-12-01T00:54:33+05:30",
        "to_account_id":
"oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e",
        "to_token_id": "INR",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~e1f5c32ab3cdc17a51ff0edda6bcc71b5acec3320a69f68a4ae455ed416657fa
",
        "transaction_type": "TOKEN_CONVERSION_DEBIT"
    }
]
```

`GetConversionRateHistory`
This method returns the token conversion rate history for a pair of tokens. This method can be called by the `Token Admin` of the chaincode, any `Org Admin`, and by any token account owner.

```
func (t *Controller) GetConversionRateHistory(from_token_id string,
to_token_id string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetConversionRateHistory",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    conversion_rate_id, err :=
t.Ctx.TokenConversionRate.GetConversionRateId(from_token_id, to_token_id)
    if err != nil {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        return nil, fmt.Errorf("error in getting conversionRateId. Error:
%s", err)
    }
    return t.Ctx.TokenConversionRate.History(conversion_rate_id)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from, for the purpose of calculating the conversion rate.
- `to_token_id: string` – The ID of the token to convert to, for the purpose of calculating the conversion rate.

Returns:

- On success, a JSON object with conversion rate history details.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-01T00:48:58+05:30",
    "TxId":
"d6c5332278d33beddbc48e535029af424fef2129bf49f4906f9b527e101d95f1",
    "Value": {
      "AssetType": "oconversionRate",
      "ConversionRate": 15,
      "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
      "FromTokenId": "USD",
      "ToTokenId": "INR"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-01T00:47:15+05:30",
    "TxId":
"e8796578351e948827d5dfe242ab4be59019ae67d69d3bfd6db255a268d57017",
    "Value": {
      "AssetType": "oconversionRate",
      "ConversionRate": 10,
      "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
      "FromTokenId": "USD",
      "ToTokenId": "INR"
    }
  }
]
```


Copyright © 2025, Oracle and/or its affiliates.

GetExchangePoolUser
This method returns the `org_id` and `user_id` values for the exchange pool user. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetExchangePoolUser() (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetExchangePoolUser", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.GetExchangePoolUser()
}
```

Parameters:

- none

Returns:

- On success, a message with information about the exchange pool user.

Return Value Example:

```
{
    "AssetType": "oconversion",
    "ConvertorId":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "OrgId": "Org1MSP",
    "UserId": "exchangepooluser"
}
```

Digital Assets

InitializeExchangePoolUser
This method initializes the exchange pool user, which is a one-time activity. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) InitializeExchangePoolUser(org_id string, user_id
string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.InitializeExchangePoolUser",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.InitializeExchangePoolUser(org_id, user_id)
}
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the exchange pool user.

Return Value Example:

```
{
    "AssetType": "oconversion",
    "ConvertorId":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "OrgId": "Org1MSP",
    "UserId": "exchangepooluser"
}
```

`CreateExchangePoolAccounts`
This method creates exchange pool token accounts for a given array of token IDs. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) CreateExchangePoolAccounts(token_ids []string)
(interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.InitializeExchangePoolUser",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    var tokens []interface{}
    for _, tokenId := range token_ids {
        token, err := t.getTokenObject(tokenId)
        if err != nil {
            return nil, fmt.Errorf("error in getting from_token asset
details. Error: %s", err)
        }
        tokens = append(tokens, token.Interface())
    }
    return t.Ctx.TokenConvertor.CreateExchangePoolAccounts(tokens)
}
```

Parameters:

- `token_ids: string []` – An array of token IDs. You can specify up to ten token IDs.

Returns:

- On success, a list of objects that includes details of the exchange pool accounts that were created.

Return Value Example:

```
[
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "AccountId":
"oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034",
        "Status": "created",
        "TokenId": "USD"
    },
    {
        "AccountId":
"oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0",
        "Status": "created",
        "TokenId": "INR"
    }
]
```

AddConversionRate
This method adds a conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddConversionRate(from_token_id string, to_token_id
string, token_conversion_rate float64) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.AddConversionRate", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.AddConversionToken(from_token_id,
to_token_id, token_conversion_rate)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: float64` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 10
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetConversionRate`
This method gets the current conversion rate for a pair of tokens. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by any token account owner.

```
func (t *Controller) GetConversionRate(from_token_id string, to_token_id
string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetConversionRate", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    conversionRateId, err :=
t.Ctx.TokenConversionRate.GetConversionRateId(from_token_id, to_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting conversationRateId. Error:
%s", err)
    }
    return t.Ctx.TokenConversionRate.Get(conversionRateId)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 10
}
```

`UpdateConversionRate`
This method updates the current conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) UpdateConversionRate(from_token_id string, to_token_id
string, token_conversion_rate float64) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.UpdateConversionRate", "TOKEN")
    if err != nil && !auth {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.TokenConvertor.UpdateTokenConversionRate(from_token_id,
to_token_id, token_conversion_rate)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: float64` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the updated conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 15
}
```

`MintWithFundingExchangePool`
This method mints tokens in the caller's account based on the specified token ID and quantity. A percentage of tokens from the minted quantity is then transferred to the exchange pool token account.

```
func (t *Controller) MintWithFundingExchangePool(token_id string,
token_quantity float64, percentage_token_to_exchangePool float64)
(interface{}, error) {
    token, err := t.getTokenObject(token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting from_token asset details.
Error: %s", err)
    }
    return
t.Ctx.TokenConvertor.MintWithFundingExchangePool(token.Interface(),
token_quantity, percentage_token_to_exchangePool)
}
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `token_quantity: float64` – The total number of tokens to mint.

Copyright © 2025, Oracle and/or its affiliates.

- `percentage_token_to_exchange_pool: float64` – The percentage of minted tokens to transfer to the exchange pool token account.

Returns:

- On success, a message that indicates that minting and funding the exchange pool were successful.

Return Value Example:

```
{
    "msg": "successfully minted 100 tokens to AccountId:
'oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1'
(OrgId: Org1MSP, User-Id: admin) and Successfully transfered 80 tokens to
exchange pool account with AccountId:
'oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0'
(OrgId: Org1MSP, UserId: exchangepooluser) "
}
```

`TokenConversion`
This method converts tokens from the caller's account to the account specified by the `to_token_id`, `to_org_id`, and `to_user_id` values. This method can be called by the `Token Admin` of the chaincode and by any token account owner. An exchange pool user cannot call this method.

```
func (t *Controller) TokenConversion(from_token_id string, to_token_id
string, to_org_id string, to_user_id string, token_quantity float64)
(interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.TokenConversion", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    from_token, err := t.getTokenObject(from_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting from_token asset details.
Error: %s", err)
    }
    to_token, err := t.getTokenObject(to_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting to_token asset details.
error: %s", err)
    }
    return t.Ctx.TokenConvertor.TokenConversion(from_token.Interface(),
to_token.Interface(), to_org_id, to_user_id, token_quantity)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.

Copyright © 2025, Oracle and/or its affiliates.

- `to_org_id: string` – The membership service provider (MSP) ID of the user in the current organization to receive the tokens.
- `to_user_id: string` – The user name or email ID of the user to receive the tokens.
- `token_quantity: float64` – The total number of tokens to transfer.

Returns:

- On success, a message that indicates the token conversion was successful.

Return Value Example:

```
{
    "msg": "succesfully converted 5 of tokens with tokenId: [USD] from
AccountId:
'oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb'
(OrgId: Org1MSP, UserId: admin) to 75 of tokens with tokenId: [INR] to
AccountId:
'oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e'
( OrgId: Org1MSP, UserId: user ) as per the conversion rate of 15"
}
```

`GetConversionHistory`
This method returns the token conversion history for a specified token account. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, and by the specified account owner.

```
func (t *Controller) GetConversionHistory(token_id string, org_id string,
user_id string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetConversionHistory", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
    if err != nil {
        return nil, fmt.Errorf("error in generating the accoint_id. Error:
%s", err)
    }
    return t.Ctx.Account.GetTokenConversionHistory(account_id, org_id,
user_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON object with conversion history details.

Return Value Example:

```
[
    {
        "balance": 95,
        "conversion_rate": 15,
        "converted_amount": 75,
        "from_account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
        "from_token_id": "USD",
        "onhold_balance": 0,
        "timestamp": "2022-12-01T00:54:33+05:30",
        "to_account_id":
"oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e",
        "to_token_id": "INR",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~e1f5c32ab3cdc17a51ff0edda6bcc71b5acec3320a69f68a4ae455ed416657fa
",
        "transaction_type": "TOKEN_CONVERSION_DEBIT"
    }
]
```

`GetConversionRateHistory`
This method returns the token conversion rate history for a pair of tokens. This method can be called by the `Token Admin`, `Token Auditor`, `Org Admin`, `Org Auditor`, or any token account owner.

```
func (t *Controller) GetConversionRateHistory(from_token_id string,
to_token_id string) (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetConversionRateHistory",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    conversion_rate_id, err :=
t.Ctx.TokenConversionRate.GetConversionRateId(from_token_id, to_token_id)
    if err != nil {
        return nil, fmt.Errorf("error in getting conversionRateId. Error:
%s", err)
    }
    return t.Ctx.TokenConversionRate.History(conversion_rate_id)
}
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from, for the purpose of calculating the conversion rate.

Copyright © 2025, Oracle and/or its affiliates.

- `to_token_id: string` – The ID of the token to convert to, for the purpose of calculating the conversion rate.

Returns:

- On success, a JSON object with conversion rate history details.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-01T00:48:58+05:30",
    "TxId":
"d6c5332278d33beddbc48e535029af424fef2129bf49f4906f9b527e101d95f1",
    "Value": {
      "AssetType": "oconversionRate",
      "ConversionRate": 15,
      "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
      "FromTokenId": "USD",
      "ToTokenId": "INR"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-01T00:47:15+05:30",
    "TxId":
"e8796578351e948827d5dfe242ab4be59019ae67d69d3bfd6db255a268d57017",
    "Value": {
      "AssetType": "oconversionRate",
      "ConversionRate": 10,
      "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
      "FromTokenId": "USD",
      "ToTokenId": "INR"
    }
  }
]
```

`GetExchangePoolUser`
This method returns the `org_id` and `user_id` values for the exchange pool user. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode.

```
func (t *Controller) GetExchangePoolUser() (interface{}, error) {
    auth, err :=
t.Ctx.Auth.CheckAuthorization("TokenConversion.GetExchangePoolUser", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return t.Ctx.TokenConvertor.GetExchangePoolUser()
}
```

Parameters:

- none

Returns:

- On success, a message with information about the exchange pool user.

Return Value Example:

```
{
    "AssetType": "oconversion",
    "ConvertorId":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "OrgId": "Org1MSP",
    "UserId": "exchangepooluser"
}
```

Token Conversion SDK Methods

`InitializeExchangePoolUser`
This method initializes the exchange pool user, which is a one-time activity. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.InitializeExchangePoolUser(org_id string, user_id string)
(interface{}, error)
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the exchange pool user.

Return Value Example:

```
{
    "AssetType": "oconversion",
    "ConvertorId":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "OrgId": "Org1MSP",
    "UserId": "exchangepooluser"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`CreateExchangePoolAccounts`
This method creates exchange pool token accounts for a given array of token IDs. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.CreateExchangePoolAccounts(tokens []interface{})
(interface{}, error)
```

Parameters:

- `token_ids: string []` – An array of token IDs.

Returns:

- On success, a list of objects that includes details of the exchange pool accounts that were created.

Return Value Example:

```
[
    {
        "AccountId":
"oaccount~cc9d84f6d4a5976532493ef5200c9603e138adc35166ffd5fd1aad9c1647f034",
        "Status": "created",
        "TokenId": "USD"
    },
    {
        "AccountId":
"oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0",
        "Status": "created",
        "TokenId": "INR"
    }
]
```

`AddConversionToken`
This method adds tokens with a new conversion rate for a specified token. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.AddConversionToken(from_token_id string, to_token_id
string, token_conversion_rate float64) (interface{}, error)
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: float64` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Returns:

- On success, a JSON representation of the conversion rate object.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 10
}
```

`Get`
This method gets the current conversion rate for a pair of tokens. This method can be called by the `Token Admin` of the chaincode and by any token account owner.

```
Ctx.TokenConversionRate.Get(id string) (ConversionRate, error)
```

Parameters:

- `id: string` – The ID of the token conversion rate object.

Returns:

- On success, a JSON representation of the conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 10
}
```

`UpdateTokenConversionRate`
This method updates the current conversion rate for a pair of tokens. The token conversion rate can be specified up to eight decimal places. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.UpdateTokenConversionRate(from_token_id string,
to_token_id string, token_conversion_rate float64) (interface{}, error)
```

Parameters:

- `from_token_id: string` – The ID of the token to convert from.
- `to_token_id: string` – The ID of the token to convert to.
- `token_conversion_rate: float64` – The rate at which to convert `from_token_id` token to the `to_token_id` token.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a JSON representation of the updated conversion rate object.

Return Value Example:

```
{
    "AssetType": "oconversionRate",
    "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
    "FromTokenId": "USD",
    "ToTokenId": "INR",
    "ConversionRate": 15
}
```

`MintWithFundingExchangePool`
This method mints tokens in the caller's account based on the specified token ID and quantity. A percentage of tokens from the minted quantity is then transferred to the exchange pool token account.

```
Ctx.TokenConvertor.MintWithFundingExchangePool(token interface{},
token_quantity float64, percentage_token_to_exchangePool float64)
(interface{}, error)
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `token_quantity: float64` – The total number of tokens to mint.
- `percentage_token_to_exchange_pool: float64` – The percentage of minted tokens to transfer to the exchange pool token account.

Returns:

- On success, a message that indicates that minting and funding the exchange pool were successful.

Return Value Example:

```
{
    "msg": "successfully minted 100 tokens to AccountId:
'oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1'
(OrgId: Org1MSP, User-Id: admin) and Successfully transfered 80 tokens to
exchange pool account with AccountId:
'oaccount~3d4933111ec8bd6cc1ebb43f2b2c390deb929cfa534f9c6ada8e63bac04a13c0'
(OrgId: Org1MSP, UserId: exchangepooluser) "
}
```

`TokenConversion`
This method converts tokens from the caller's account to the account specified by the `to_token_id`, `to_org_id`, and `to_user_id` values. This method can be called by the `Token`

Copyright © 2025, Oracle and/or its affiliates.

Admin of the chaincode and by any token account owner. An exchange pool user cannot call this method.

```
Ctx.TokenConvertor.TokenConversion(from_token interface{}, to_token
interface{}, to_org_id string, to_user_id string, token_quantity float64)
(interface{}, error)
```

Parameters:

- from_token_id: string – The ID of the token to convert from.
- to_token_id: string – The ID of the token to convert to.
- to_org_id: string – The membership service provider (MSP) ID of the user in the current organization to receive the tokens.
- to_user_id: string – The user name or email ID of the user to receive the tokens.

Returns:

- On success, a message that indicates the token conversion was successful.

Return Value Example:

```
{
    "msg": "succesfully converted 5 of tokens with tokenId: [USD] from
AccountId:
'oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb'
(OrgId: Org1MSP, UserId: admin) to 75 of tokens with tokenId: [INR] to
AccountId:
'oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e'
( OrgId: Org1MSP, UserId: user ) as per the conversion rate of 15"
}
```

GetTokenConversionHistory
This method returns the token conversion history for a specified token account. This method can be called by the Token Admin of the chaincode and by the token account owner.

```
Ctx.Account.GetTokenConversionHistory(account_id string, org_id string,
user_id string) (interface{}, error)
```

Parameters:

- account_id: string – The ID of the fungible token account.
- org_id: string – The membership service provider (MSP) ID of the user in the current organization.
- user_id: string – The user name or email ID of the user.

Returns:

- On success, a JSON object with conversion history details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "balance": 95,
        "conversion_rate": 15,
        "converted_amount": 75,
        "from_account_id":
"oaccount~abc74791148b761352b98df58035601b6f5480448ac2b4a3a7eb54bdbebf48eb",
        "from_token_id": "USD",
        "onhold_balance": 0,
        "timestamp": "2022-12-01T00:54:33+05:30",
        "to_account_id":
"oaccount~25e2e66718b6dbb59aea9c32acebec60e09d912b2578d4933d377ae5d0628f1e",
        "to_token_id": "INR",
        "transacted_amount": 5,
        "transaction_id":
"otransaction~e1f5c32ab3cdc17a51ff0edda6bcc71b5acec3320a69f68a4ae455ed416657fa
",
        "transaction_type": "TOKEN_CONVERSION_DEBIT"
    }
]
```

`history`
This method returns the token conversion rate history for a pair of tokens. This method can be called by the `Token Admin` of the chaincode, any `Org Admin`, and by any token account owner.

```
Ctx.TokenConversionRate.History(conversion_rate_id string) (interface{},
error)
```

Parameters:

- `conversion_rate_id: string` – The ID of the conversion rate object.

Returns:

- On success, a JSON object with conversion rate history details.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-01T00:48:58+05:30",
    "TxId":
"d6c5332278d33beddbc48e535029af424fef2129bf49f4906f9b527e101d95f1",
    "Value": {
      "AssetType": "oconversionRate",
      "ConversionRate": 15,
      "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
      "FromTokenId": "USD",
      "ToTokenId": "INR"
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-01T00:47:15+05:30",
    "TxId":
"e8796578351e948827d5dfe242ab4be59019ae67d69d3bfd6db255a268d57017",
    "Value": {
      "AssetType": "oconversionRate",
      "ConversionRate": 10,
      "ConvertorRateId":
"oconversionRate~79eacc670928bbc4c9ba4ebee135c8b4d6411af3110f8a9b782c383d5e18b
150",
      "FromTokenId": "USD",
      "ToTokenId": "INR"
    }
  }
]
```

`GetExchangePoolUser`
This method returns the `org_id` and `user_id` values for the exchange pool user. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.TokenConvertor.GetExchangePoolUser()
```

Parameters:

- none

Returns:

- On success, a message with information about the exchange pool user.

Return Value Example:

```
{
    "AssetType": "oconversion",
    "ConvertorId":
"bcb1f3b1442c625d3ce205660c5e717c5858a1fe1e12c325df799a851ceaa09b",
    "OrgId": "Org1MSP",
    "UserId": "exchangepooluser"
}
```

Go Methods for Token Account Status

Blockchain App Builder automatically generates methods that you can use to manage account status for fungible tokens that use the Token Taxonomy Framework standard.

You can use the following methods to put token user accounts in the active, suspended, or deleted states.

When an account is suspended, the account user cannot complete any write operations, which include minting, burning, transferring, and holding tokens. Additionally, other users cannot transfer tokens to or hold tokens in a suspended account. A suspended account can still complete read operations.

Copyright © 2025, Oracle and/or its affiliates.

An account with a non-zero token balance cannot be deleted. You must transfer or burn all tokens in an account before you can delete the account. After an account is in the deleted state, the account state cannot be changed back to active or suspended.

- Automatically Generated Account Status Methods
- Account Status SDK Methods

Automatically Generated Account Status Methods

Blockchain App Builder automatically generates methods to manage token account status. Controller methods must be public to be invokable. Public method names begin with an upper case character. Method names that begin with a lower case character are private.

- Base
- Digital Assets

Base

`GetAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner. This method also supports data migration for existing chaincode that is upgraded to a newer version.

```
func (t *Controller) GetAccountStatus(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("AccountStatus.Get",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      accountStatus, err := t.Ctx.AccountStatus.GetAccountStatus(account_id)
      if err != nil {
            return t.Ctx.AccountStatus.GetDefaultAccountStatus(account_id)
      }
      return accountStatus, nil
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`GetAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner.

```
func (t *Controller) GetAccountStatusHistory(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      _, err = t.Ctx.Account.GetAccount(account_id)
      if err != nil {
            return nil, fmt.Errorf("error in GetAccountStatusHistory: %s",
err)
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("AccountStatus.Get",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      status_id, err :=
t.Ctx.AccountStatus.GenerateAccountStatusId(account_id)
      if err != nil {
            return nil, err
      }
      account_status_history, err := t.Ctx.AccountStatus.History(status_id)
      if err != nil {
            return []map[string]interface{}{}, nil
      }
      return account_status_history, nil
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the account status history.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

`ActivateAccount`
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization. Deleted accounts cannot be activated.

```
func (t *Controller) ActivateAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("AccountStatus.ActivateAccount", "TOKEN",
map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.ActivateAccount(account_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`SuspendAccount`
This method suspends a token account. Suspended accounts can still complete read operations. Users with suspended accounts cannot send, receive, mint, or burn tokens. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
func (t *Controller) SuspendAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
```


Copyright © 2025, Oracle and/or its affiliates.

```
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("AccountStatus.SuspendAccount", "TOKEN",
map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.SuspendAccount(account_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

`DeleteAccount`
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance and the on-hold balance must be zero.

```
func (t *Controller) DeleteAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("AccountStatus.DeleteAccount", "TOKEN",
```


Copyright © 2025, Oracle and/or its affiliates.

```
map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.DeleteAccount(account_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```

Digital Assets

`GetAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the token account owner. This method also supports data migration for existing chaincode that is upgraded to a newer version.

```
func (t *Controller) GetAccountStatus(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("AccountStatus.Get",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
```


Copyright © 2025, Oracle and/or its affiliates.

```
      }
      accountStatus, err := t.Ctx.AccountStatus.GetAccountStatus(account_id)
      if err != nil {
            return t.Ctx.AccountStatus.GetDefaultAccountStatus(account_id)
      }
      return accountStatus, nil
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`GetAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or by the token account owner.

```
func (t *Controller) GetAccountStatusHistory(token_id string, org_id string,
user_id string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      _, err = t.Ctx.Account.GetAccount(account_id)
      if err != nil {
            return nil, fmt.Errorf("error in GetAccountStatusHistory: %s",
err)
      }
      auth, err := t.Ctx.Auth.CheckAuthorization("AccountStatus.Get",
"TOKEN", map[string]string{"account_id": account_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
```


Copyright © 2025, Oracle and/or its affiliates.

```
err.Error())
      }
      status_id, err :=
t.Ctx.AccountStatus.GenerateAccountStatusId(account_id)
      if err != nil {
            return nil, err
      }
      account_status_history, err := t.Ctx.AccountStatus.History(status_id)
      if err != nil {
            return []map[string]interface{}{}, nil
      }
      return account_status_history, nil
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the account status history.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

`ActivateAccount`
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization. Deleted accounts cannot be activated.

```
func (t *Controller) ActivateAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("AccountStatus.ActivateAccount", "TOKEN",
map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.ActivateAccount(account_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```


Copyright © 2025, Oracle and/or its affiliates.

SuspendAccount
This method suspends a token account. Suspended accounts can still complete read operations. Users with suspended accounts cannot send, receive, mint, or burn tokens. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
func (t *Controller) SuspendAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("AccountStatus.SuspendAccount", "TOKEN",
map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.SuspendAccount(account_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

DeleteAccount
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final

Copyright © 2025, Oracle and/or its affiliates.

state and cannot be changed to any other state. To delete an account, the account balance and the on-hold balance must be zero.

```
func (t *Controller) DeleteAccount(token_id string, org_id string, user_id
string) (interface{}, error) {
      account_id, err := t.Ctx.Account.GenerateAccountId(token_id, org_id,
user_id)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating
account_id of (Org-Id: %s, User-Id: %s)", org_id, user_id)
      }
      auth, err :=
t.Ctx.Auth.CheckAuthorization("AccountStatus.DeleteAccount", "TOKEN",
map[string]string{"org_id": org_id})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.Account.DeleteAccount(account_id)
}
```

Parameters:

- `token_id: string` – The ID of the token.
- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Account Status SDK Methods

`GetDefaultAccountStatus`
This method gets the current status of a token account, with the status as `active` for any account that does not have account status stored in the ledger (because the account was created prior to the account status functionality).

```
Ctx.AccountStatus.GetDefaultAccountStatus(account_id string)
(FungibleAccountStatus, error)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`GetAccountStatus`
This method gets the current status of the token account.

```
Ctx.AccountStatus.GetAccountStatus(account_id string) (FungibleAccountStatus,
error)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "Status": "active"
}
```

SaveAccountStatus
This method saves the status object (if a status object is not present) for the token account, and sets the status to the specified value.

```
Ctx.AccountStatus.SaveAccountStatus(account_id string, status string)
```

Parameters:

- `account_id: string` – The ID of the token account.
- `status: string` – The status to set for the specified account. Only three values are supported: `active`, `suspended`, or `deleted`.

Returns:

- On success, a JSON representation of the account status object.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

GetAccountStatusHistory
This method gets the history of the account status.

```
Ctx.AccountStatus.History(status_id string) (interface{}, error)
```

Parameters:

- `status_id: string` – The ID of the account status object.

Returns:

- On success, a JSON representation of the account status history.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

`ActivateAccount`
This method activates a token account.

```
Ctx.Account.ActivateAccount(account_id: string) (interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```


Copyright © 2025, Oracle and/or its affiliates.

SuspendAccount
This method suspends a token account.

```
Ctx.Account.SuspendAccount(account_id string) (interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

DeleteAccount
This method deletes a token account.

```
Ctx.Account.DeleteAccount(account_id string) (interface{}, error)
```

Parameters:

- `account_id: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```


Copyright © 2025, Oracle and/or its affiliates.

ERC-721

Blockchain App Builder supports an extended version of the ERC-721 standard to work with non-fungible tokens.

- Input Specification File for ERC-721
- Scaffolded TypeScript NFT Project for ERC-721
- Scaffolded Go NFT Project for ERC-721

Input Specification File for ERC-721

The Blockchain App Builder initialization command reads the input specification file and generates the scaffolded project with several tools to assist in the chaincode development process.

You can define standard assets and token assets that are based on the ERC-721 standard in the same specification file. The following restrictions apply to a specification file that includes an ERC-721 token asset definition:

- You cannot define token assets based on more than one standard in the same specification file.
- You cannot define multiple non-fungible token assets in a single specification file.

Custom attribute values for non-fungible token assets can be updated by the token owner.

For information on including standard assets in the specification file, see Input Specification File.

The following sample specification file for an ERC-721 token asset is available in the Blockchain App Builder package:

- `NFTArtCollectionMarketPlace with ERC-721-Typescript.yml`

In addition to the standard properties and sections, non-fungible token assets support the `behavior`, `anatomy` and `metadata` sections in the specification file. Non-fungible token assets also support the `standard` and `symbol` properties. The following example shows the structure of a specification file for an ERC-721 token asset:

```
assets:
    - name: ArtCollection #Asset name
      type: token #Asset type
      symbol: ART # Token symbol
      standard: erc721+ # Token standard

      anatomy:
          type: nonfungible # Token type
          unit: whole #Token unit

      behavior:
        - indivisible
        - singleton
        - mintable:
            max_mint_quantity: 20000
        - transferable
        - burnable
```


Copyright © 2025, Oracle and/or its affiliates.

```
        - lockable
        - roles:
            minter_role_name: minter

      properties: # Custom asset attributes for non-fungible tokens

          - name: price # Custom asset attribute to set the price of a non-
fungible token in the marketplace
            type: number

          - name: on_sale_flag # Custom asset attribute maintains non-
fungible token selling status in the marketplace
            type: boolean

      metadata: # Use this section to maintain the metadata on the
blockchain. Only the user creating the non-fungible token can assign metadata
attribute values, which cannot be updated later.

          - name: painting_name
            type: string

          - name: description
            type: string

          - name: image
            type: string

          - name: painter_name
            type: string
```

Table 4-2 Parameter Descriptions and Examples for an ERC-721 Token Specification File

Entry	Description	Examples

`type:`	You must specify `type: token` in the `assets` section.	`assets:`
		`  - name: ArtCollection #Asset name`
		`  type: token #Asset type`

`symbol:`	The `symbol` property represents the symbol that identifies the token contract, for example ART. Typically the symbol is 3 to 4 characters long.	`symbol: ART # Token symbol`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-2 (Cont.) Parameter Descriptions and Examples for an ERC-721 Token Specification File

Entry	Description	Examples
`standard:`	The `standard` property is mandatory for non-fungible tokens. It represents the token standard to follow during chaincode generation. In Blockchain App Builder, non-fungible tokens follow a partial version of the ERC-721 standard. For more information, see Limitations.	`standard: erc721+ # Token standard`

`anatomy:`	The `anatomy` section has two mandatory parameters for non-fungible tokens:
	• `type: nonfungible` A non-fungible token is unique.
	• `unit: whole` A whole token cannot be subdivided into smaller fractional units.
		`anatomy:`
		`  type: nonfungible # Token type`
		`  unit: whole #Token unit`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-2 (Cont.) Parameter Descriptions and Examples for an ERC-721 Token Specification File

Entry	Description	Examples

`behavior:`	This section describes the capabilities and restrictions of the token. The `mintable`, `transferable`, `singleton`, and `indivisible` behaviors are mandatory for non-fungible tokens.
	• `singleton`: This mandatory behavior supports a restriction so that there can be only one whole token in the class, which cannot be divided.
	• `indivisible`: This mandatory behavior supports a restriction so that whole tokens cannot be subdivided into fractions.
	• `mintable`: This mandatory behavior supports minting new token instances. The optional `max_mint_quantity` parameter specifies the total number of tokens that can be minted. If you do not specify the `max_mint_quantity` parameter, any number of tokens can be minted.
	• `transferable`: This mandatory behavior supports transferring ownership of tokens.
	• `burnable`: This optional behavior supports deactivating, or burning, tokens.
		`behavior:`
		`- indivisible`
		`- singleton`
		`- mintable:`
		`max_mint_quantity: 20000`
		`- transferable`
		`- burnable`
		`- lockable`
		`- roles:`
		`minter_role_name: minter`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-2 (Cont.) Parameter Descriptions and Examples for an ERC-721 Token Specification File

Entry	Description	Examples
	Burning does not delete a token but instead places it in a permanent state where it cannot be used. Burning is not reversible. • `lockable`: This behavior is optional and is supported only by non-fungible tokens. This behavior allows the token owner to lock a non-fungible token. A locked token cannot be transferred to or burned by any other users. • `roles`: This optional behavior restricts token behaviors to users with specific roles. Currently two roles are available: `minter_role_name` and `burner_role_name`. If you do not specify roles, then any user can act as a minter or burner. For example, if the burner role is not specified, any account user implicitly has the burner role. If the burner role is specified, then during the token setup process, the `Token Admin` user must assign the burner role to other users explicitly.	

Copyright © 2025, Oracle and/or its affiliates.

Table 4-2 (Cont.) Parameter Descriptions and Examples for an ERC-721 Token Specification File

Entry	Description	Examples

`metadata:`	The `metadata` property is optional and is supported only by non-fungible tokens. This property specifies metadata information, which is stored on the blockchain, for a non-fungible token. Metadata attribute values can be assigned only by the token owner who mints the token, and cannot be updated after the token is minted.
	In the example, `name` is the name of the metadata attribute and `type` is the type of value that the attribute has.
		`metadata:`
		`- name: painting_name`
		`type: string`
		`- name: description`
		`type: string`
		`- name: image`
		`type: string`
		`- name: painter_name`
		`type: string`

Limitations

Blockchain App Builder provides partial support for the ERC-721 standard. Currently, the following ERC-721 events and methods are not supported.

Events:

- `event Transfer`
- `event Approval`
- `event ApprovalForAll`

Methods:

- `approve`
- `getApproved`
- `setApprovalForAll`
- `isApprovedForAll`

Scaffolded TypeScript NFT Project for ERC-721

Blockchain App Builder takes the input from your NFT specification file and generates a fully-functional scaffolded chaincode project.

The project automatically generates NFT lifecycle classes and functions, including CRUD and non-CRUD methods, as well as a tokenization SDK. Validation of arguments, marshalling/unmarshalling, and transparent persistence capability are all supported automatically.

For information on the scaffolded project and methods that are not directly related to NFTs, see Scaffolded TypeScript Chaincode Project.

Copyright © 2025, Oracle and/or its affiliates.

Reference:

- Model
- Controller
  - Automatically Generated NFT Methods
  - Custom Methods
- NFT SDK Methods

Model

Every tokenized model class extends the `OchainModel` class. Transparent Persistence Capability, or simplified ORM, is captured in the `OchainModel` class.

```
import * as yup from 'yup';
import { Id, Mandatory, Validate, Default, Embedded, Derived, ReadOnly } from
'../../lib/decorators';
import { OchainModel } from '../../lib/ochain-model';
import { STRATEGY } from '../../lib/utils';
import { EmbeddedModel } from '../../lib/ochain-embedded-model';

export class ArtCollectionMetadata extends
EmbeddedModel<ArtCollectionMetadata> {
    @Validate(yup.string())
    public painting_name: string;

    @Validate(yup.string())
    public description: string;

    @Validate(yup.string())
    public image: string;

    @Validate(yup.string())
    public painter_name: string;

}

@Id('tokenId')
export class ArtCollection extends OchainModel<ArtCollection> {

    public readonly assetType = 'otoken';

    @Mandatory()
    @Validate(yup.string().required().matches(/^[A-Za-z0-9][A-Za-
z0-9_-]*$/).max(16))
    public tokenId: string;

    @ReadOnly('artcollection')
    public tokenName: string;

    @Validate(yup.string().trim().max(256))
    public tokenDesc: string;

    @ReadOnly('ART')
    public symbol: string;
```


Copyright © 2025, Oracle and/or its affiliates.

```
    @ReadOnly('erc721+')
    public tokenStandard: string;

    @ReadOnly('nonfungible')
    public tokenType: string;

    @ReadOnly('whole')
    public tokenUnit: string;


@ReadOnly(["indivisible","singleton","mintable","transferable","burnable","rol
es"])
    public behaviors: string[];

    @ReadOnly({minter_role_name: "minter"})
    public roles: object;

    @ReadOnly({max_mint_quantity: 20000})
    public mintable: object;

    @Validate(yup.string())
    public owner: string;

    @Validate(yup.string())
    public createdBy: string;

    @Validate(yup.string())
    public transferredBy: string;

    @Validate(yup.string())
    public creationDate: string;

    @Validate(yup.string())
    public transferredDate: string;

    @Validate(yup.bool())
    public isBurned: boolean;

    @Validate(yup.string())
    public burnedBy: string;

    @Validate(yup.string())
    public burnedDate: string;

    @Validate(yup.string().max(2000))
    public tokenUri: string;

    @Embedded(ArtCollectionMetadata)
    public metadata: ArtCollectionMetadata;

    @Validate(yup.number())
    public price: number;

    @Validate(yup.boolean())
    public on_sale_flag: boolean;
```


Copyright © 2025, Oracle and/or its affiliates.

```
}
```

Controller

The main controller class extends the `OchainController` class. There is only one main controller.

```
export class DigiCurrCCController extends OchainController{
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable. The other methods are hidden.

You can use the token SDK methods to write custom methods for your business application.

Automatically Generated NFT Methods

Blockchain App Builder automatically generates methods to support NFTs and NFT life cycles. You can use these methods to initialize NFTs, manage roles and accounts, and complete other NFT lifecycle tasks without any additional coding. Controller methods must have a `@Validator(...params)` decorator to be invokable.

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

`addTokenAdmin`
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async addTokenAdmin(orgId: string, userId: string) {
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721ADMIN.addAdmin',
'TOKEN');
    return await this.Ctx.ERC721Admin.addAdmin(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfully added Admin (orgId: Org1MSP, userId: User1)"}
```

`removeTokenAdmin`
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async removeTokenAdmin(orgId: string, userId: string) {
   await this.Ctx.ERC721Auth.checkAuthorization('ERC721ADMIN.removeAdmin',
'TOKEN');
   return await this.Ctx.ERC721Admin.removeAdmin(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg": "Successfully removed Admin (orgId: Org1MSP, userId: User1)"}
```

`isTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. A `Token Admin` can call this function on any other user in the blockchain network. Other users can call this method only on their own accounts.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async isTokenAdmin(orgId: string, userId: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721ADMIN.isUserTokenAdmin',
'TOKEN');
    return await this.Ctx.ERC721Auth.isUserTokenAdmin(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `userId: string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

Return Value Example:

```
{"result": true}
```

`getAllTokenAdmins`
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by the `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async getAllTokenAdmins() {
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721ADMIN.getAllAdmins',
'TOKEN');
    return await this.Ctx.ERC721Admin.getAllAdmins();
}
```

Parameters:

- none

Returns:

- On success, an `admins` array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{"admins":[{"orgId":"Org1MSP","userId":"admin"}]}
```

Methods for Token Configuration Management

`init`
This method is called when the chaincode is instantiated. Every `Token Admin` is identified by the `userId` and `orgId` information in the `adminList` parameter. The `userId` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `orgId` is the membership service provider (MSP) ID of the user in the current network organization. The `adminList` parameter is mandatory the first time you deploy the chaincode. If you are upgrading the chaincode, pass an empty list (`[]`). Any other information in the `adminList` parameter is ignored during upgrades.

```
@Validator(yup.array().of(yup.object()).nullable())
public async init(adminList: ERC721TokenAdminAsset[]) {
   await this.Ctx.ERC721Admin.initAdmin(adminList);
   await this.Ctx.ERC721Token.saveClassInfo(<NFT_NAME>);
   await this.Ctx.ERC721Token.saveDeleteTransactionInfo();
   return;
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `adminList array` – An array of `{orgId, userId}` information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

`getAllTokens`
This method returns all of the token assets that are saved in the state database. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@GetMethod()
@Validator()
public async getAllTokens() {
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.getAllTokens',
'TOKEN');
    return await this.Ctx.ERC721Token.getAllTokens();
}
```

Parameters:

- none

Returns:

- A list of all token assets in JSON format.

Return Value Example:

```
[
    {
        "key": "monalisa",
        "valueJson": {
            "metadata": {
                "PaintingName": "Mona_Lisa",
                "Description": "Mona Lisa Painting",
                "Image": "monalisa.jpeg",
                "PainterName": "Leonardo_da_Vinci"
            },
            "assetType": "otoken",
            "tokenId": "monalisa",
            "tokenName": "ravinft",
            "tokenDesc": "token Description",
            "symbol": "PNT",
            "tokenStandard": "erc721+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "createdBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "creationDate": "2022-04-07T21:17:48.000Z",
            "isBurned": false,
            "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "NftBasePrice": 100
        }
    },
    {
        "key": "monalisa1",
        "valueJson": {
            "metadata": {
                "PaintingName": "Mona_Lisa",
                "Description": "Mona Lisa Painting",
                "Image": "monalisa.jpeg",
                "PainterName": "Leonardo_da_Vinci"
            },
            "assetType": "otoken",
            "tokenId": "monalisa1",
            "tokenName": "ravinft",
            "tokenDesc": "token Description",
            "symbol": "PNT",
            "tokenStandard": "erc721+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "createdBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "creationDate": "2022-04-07T21:17:59.000Z",
            "isBurned": false,
            "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "NftBasePrice": 100
        }
    }
]
```

getAllTokensByUser
This method returns all of the token assets that are owned by a specified user. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAllTokensByUser(orgId: string, userId: string) {
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.getAllTokensByUser',
'TOKEN', { accountId });
    return await this.Ctx.ERC721Token.getAllTokensByUser(accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- A list of token assets in JSON format.

Return Value Example:

```
[
    {
        "key": "monalisa",
        "valueJson": {
            "metadata": {
                "PaintingName": "Mona_Lisa",
                "Description": "Mona Lisa Painting",
                "Image": "monalisa.jpeg",
                "PainterName": "Leonardo_da_Vinci"
            },
            "assetType": "otoken",
            "tokenId": "monalisa",
            "tokenName": "ravinft",
            "tokenDesc": "token Description",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "symbol": "PNT",
                "tokenStandard": "erc721+",
                "tokenType": "nonfungible",
                "tokenUnit": "whole",
                "behaviors": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "createdBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "creationDate": "2022-04-07T21:17:48.000Z",
                "isBurned": false,
                "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
                "NftBasePrice": 100
            }
        },
        {
            "key": "monalisa1",
            "valueJson": {
                "metadata": {
                    "PaintingName": "Mona_Lisa",
                    "Description": "Mona Lisa Painting",
                    "Image": "monalisa.jpeg",
                    "PainterName": "Leonardo_da_Vinci"
                },
                "assetType": "otoken",
                "tokenId": "monalisa1",
                "tokenName": "ravinft",
                "tokenDesc": "token Description",
                "symbol": "PNT",
                "tokenStandard": "erc721+",
                "tokenType": "nonfungible",
                "tokenUnit": "whole",
                "behaviors": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "createdBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "creationDate": "2022-04-07T21:17:59.000Z",
            "isBurned": false,
            "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "NftBasePrice": 100
        }
    }
]
```

getTokenById
This method returns a token object if the token is present in the state database. This method can be called only by a `Token Admin` of the chaincode or the token owner.

```
@GetMethod()
@Validator(yup.string())
public async getTokenById(tokenId: string) {
   await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.get', 'TOKEN',
{ tokenId });
   let token = await this.getTokenObject(tokenId);
   return token;
}
```

Parameters:

- `tokenId: string` – The ID of the token to get.

Returns:

- The token asset in JSON format.

Return Value Example:

```
{
    "metadata": {
        "painting_name": "Mona_Lisa",
        "description": "Mona Lisa Painting",
        "image": "monalisa.jpeg",
        "painter_name": "Leonardo_da_Vinci"
    },
    "assetType": "otoken",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "tokenId": "monalisa",
    "tokenName": "artcollection",
    "tokenDesc": "token description",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "transferredBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-04-05T08:30:42.000Z",
    "transferredDate": "2022-04-05T09:28:30.000Z",
    "isBurned": false,
    "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
    "price": 100,
    "on_sale_flag": true
}
```

`getTokenHistory`
This method returns the history for a specified token ID. This is an asynchronous method. This method can only be called when connected to the remote Oracle Blockchain Platform network. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async getTokenHistory(tokenId: string) {
   // await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.history',
'TOKEN');
   return await this.Ctx.ERC721Token.history(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
[
    {
        "trxId":
"ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7",
        "timeStamp": 1649150910,
        "value": {
            "metadata": {
                "painting_name": "Mona_Lisa",
                "description": "Mona Lisa Painting",
                "image": "monalisa.jpeg",
                "painter_name": "Leonardo_da_Vinci"
            },
            "assetType": "otoken",
            "tokenId": "monalisa",
            "tokenName": "artcollection",
            "tokenDesc": "token description",
            "symbol": "ART",
            "tokenStandard": "erc721+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "transferredBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "creationDate": "2022-04-05T08:30:42.000Z",
            "transferredDate": "2022-04-05T09:28:30.000Z",
            "isBurned": false,
            "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "price": 100,
            "on_sale_flag": true
        }
    },
    {
        "trxId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb",
        "timeStamp": 1649149545,
        "value": {
            "metadata": {
                "painting_name": "Mona_Lisa",
                "description": "Mona Lisa Painting",
                "image": "monalisa.jpeg",
                "painter_name": "Leonardo_da_Vinci"
            },
            "assetType": "otoken",
            "tokenId": "monalisa",
            "tokenName": "artcollection",
            "tokenDesc": "token description",
            "symbol": "ART",
            "tokenStandard": "erc721+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "owner":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
            "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "transferredBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "creationDate": "2022-04-05T08:30:42.000Z",
            "transferredDate": "2022-04-05T09:05:45.000Z",
            "isBurned": false,
            "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "price": 100,
            "on_sale_flag": true
        }
    },
    {
        "trxId":
"702e61cc8d6d2982521023d0d5f3195900f35e146d6a90ef66daae551e6075d2",
        "timeStamp": 1649147729,
        "value": {
            "metadata": {
                "painting_name": "Mona_Lisa",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "description": "Mona Lisa Painting",
                    "image": "monalisa.jpeg",
                    "painter_name": "Leonardo_da_Vinci"
                },
                "assetType": "otoken",
                "tokenId": "monalisa",
                "tokenName": "artcollection",
                "tokenDesc": "token description",
                "symbol": "ART",
                "tokenStandard": "erc721+",
                "tokenType": "nonfungible",
                "tokenUnit": "whole",
                "behaviors": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "creationDate": "2022-04-05T08:30:42.000Z",
                "isBurned": false,
                "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
                "price": 100,
                "on_sale_flag": true
            }
        },
        {
            "trxId":
"e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c",
            "timeStamp": 1649147442,
            "value": {
                "metadata": {
                    "painting_name": "Mona_Lisa",
                    "description": "Mona Lisa Painting",
                    "image": "monalisa.jpeg",
                    "painter_name": "Leonardo_da_Vinci"
                },
                "assetType": "otoken",
                "tokenId": "monalisa",
                "tokenName": "artcollection",
                "tokenDesc": "token description",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "symbol": "ART",
            "tokenStandard": "erc721+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "creationDate": "2022-04-05T08:30:42.000Z",
            "isBurned": false,
            "tokenUri": "\"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg\"",
            "price": 100,
            "on_sale_flag": false
        }
    }
]
```

getTokenObject

This is a utility method that returns an instance of the token for a specified token ID. This method is used by many of the automatically generated methods to fetch token objects. You can call this method as needed from your custom methods. When you create a tokenized asset or class, update the switch case with the corresponding `Token` class to return the correct token object. The `ochain sync` command in Blockchain App Builder automatically creates a switch case when a tokenized asset is created in the specification file. This method has no `@Validator()` method decorator, which means this method is not directly invokable and can only be called from other methods.

```
public async getTokenObject<T extends OchainModel<any>>(tokenId: string):
Promise<T> {
   if (!tokenId) {
        throw Error('TokenID cannot be null/empty.');
    }
    const token = await this.Ctx.ERC721Token.get(tokenId);
    if (token.tokenName && token.assetType && token.assetType === 'otoken') {
        let tokenAsset;
        switch (token.tokenName) {
           case '<NFT_NAME in lowercase>':
```


Copyright © 2025, Oracle and/or its affiliates.

```
                tokenAsset = new <NFT_NAME>(token, false, true);
                return tokenAsset;
            default:
                throw new Error(`No token exists with ID [${tokenId}]`);
        }
    } else {
        throw new Error(`No token exists with ID [${tokenId}]`);
    }
}
```

Parameters:

- `tokenId: string` – The ID of the token.

`ownerOf`
This method returns the account ID of the owner of the specified token ID. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async ownerOf(tokenId: string) {
   return await this.Ctx.ERC721Token.ownerOf(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- A JSON object of the owner's account ID.

Return Value Example:

```
{"owner":
"oaccount~d6d22c3167e3c6ab9ee5653e1a008c37c20cc47ebb0229ca0aedfafe64c675b8"}
```

`name`
This method returns the name of the token class. Anyone can call this method.

```
@GetMethod()
@Validator()
public async name() {
   return await this.Ctx.ERC721Token.name();
}
```

Parameters:

- none

Returns:

- A JSON object of the token name.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"tokenName": "artcollection"}
```

symbol
This method returns the symbol of the token class. Anyone can call this method.

```
@GetMethod()
@Validator()
public async symbol() {
   return await this.Ctx.ERC721Token.symbol();
}
```

Parameters:

- none

Returns:

- A JSON object of the token symbol.

Return Value Example:

```
{"symbol": "PNT"}
```

tokenURI
This method returns the URI of a specified token. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async tokenURI(tokenId: string) {
   return await this.Ctx.ERC721Token.tokenURI(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object of the token URI.

Return Value Example:

```
{"tokenURI": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\
.ipfs.infura-ipfs.io/?filename=MonaLisa.jpeg"}
```


Copyright © 2025, Oracle and/or its affiliates.

`totalSupply`
This method returns the total number of minted tokens. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async totalSupply() {
  await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.totalSupply',
'TOKEN');
  return await this.Ctx.ERC721Token.totalSupply();
}
```

Parameters:

- none

Returns:

- On success, a JSON object of the token count.

Return Value Example:

```
{"totalSupply": 3}
```

`totalNetSupply`
This method returns the total number of minted tokens minus the number of burned tokens. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async totalNetSupply() {
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.totalNetSupply', 'TOKEN');
    return await this.Ctx.ERC721Token.getTotalMintedTokens();
}
```

Parameters:

- none

Returns:

- On success, a JSON object of the token count.

Return Value Example:

```
{"totalNetSupply": 1}
```

Methods for Account Management

`createAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track the number of NFTs a user has.

Copyright © 2025, Oracle and/or its affiliates.

Users must have accounts in the network to complete token-related operations. You can create only one NFT account per user.
An account ID is an alphanumeric set of characters, prefixed with `oaccount~` and followed by an SHA-256 hash of the membership service provider ID (`orgId`) of the user in the current network organization, the user name or email ID (`userId`) of the instance owner or the user who is logged in to the instance, and the constant string `nft`. This method can be called only by the `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async createAccount(org_id: string, user_id: string, token_type:
string) {
  await this.Ctx.Auth.checkAuthorization("ACCOUNT.createAccount", "TOKEN",
{ org_id });
  return await this.Ctx.Account.createAccount(org_id, user_id, token_type);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenType: string` – The only supported token type is `nonfungible`.

Returns:

- On success, a JSON object of the account that was created. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
    "assetType": "oaccount",
    "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "bapAccountVersion": 0,
    "userId": "admin",
    "orgId": "Org1MSP",
    "tokenType": "nonfungible",
    "noOfNfts": 0
}
```

`balanceOf`
This method returns the total number of NFTs that a specified user holds. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string())
 public async balanceOf(orgId: string, userId: string) {
     await this.Ctx.ERC721Auth.checkAuthorization('ERC721ACCOUNT.balanceOf',
'TOKEN', { orgId, userId });
     const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.ERC721Account.balanceOf(accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- A JSON object of the current NFT count.

Return Value Example:

```
{"totalNfts": 0}
```

`getAllAccounts`
This method returns a list of all accounts. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@GetMethod()
@Validator()
public async getAllAccounts() {
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721ACCOUNT.getAllAccounts',
'TOKEN');
    return await this.Ctx.ERC721Account.getAllAccounts();
}
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

Return Value Example:

```
[
    {
        "key":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "valueJson": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
]
```

getAccountByUser
This method returns account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAccountByUser(orgId: string, userId: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721ACCOUNT.getAccountByUser',
'TOKEN', { orgId, userId });
    return await this.Ctx.ERC721Account.getAccountWithStatusByUser(orgId,
userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `bapAccountVersion` – An account object parameter for internal use.
- `status` – The current status of the user account.
- `accountId` – The ID of the user account.
- `userId` – The user name or email ID of the user.
- `orgId` – The membership service provider (MSP) ID of the user in the current organization.
- `tokenType` – The type of token that the account holds.
- `noOfNfts` – The total number of NFTs held by the account.

Return Value Example:

```
{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "accountId":
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
  "userId": "idcqa",
  "orgId": "appdev",
  "tokenType": "nonfungible",
  "noOfNfts": 0
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getUserByAccountId`
This method returns the user details of a specified account. This method can be called by any user.

```
@GetMethod()
@Validator(yup.string())
public async getUserByAccountId(accountId: string) {
    return await this.Ctx.ERC721Account.getUserByAccountId(accountId);
}
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`orgId` and `userId`).

Return Value Example:

```
{
  "userId": "admin",
  "orgId": "Org1MSP"
}
```

`getAccountHistory`
This method returns account history for a specified user. This is an asynchronous method. This method can be called only by the `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAccountHistory(orgId: string, userId: string) {
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721ACCOUNT.history',
'TOKEN', { accountId });
    return await this.Ctx.ERC721Account.history(accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object of the account history. The `bapAccountVersion` parameter is defined in the account object for internal use.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
[
    {
        "trxId":
"6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4",
        "timeStamp": 1649151044,
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion" : 5,
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    },
    {
        "trxId":
"a605f1fa62e511c2945fce5437f983a5e70ec814b82520d3ecd2d81e3ecf53a3",
        "timeStamp": 1649151022,
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion" : 4,
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 2
        }
    },
    {
        "trxId":
"ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7",
        "timeStamp": 1649150910,
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion" : 3,
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    },
    {
        "trxId":
"cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb",
        "timeStamp": 1649149545,
        "value": {
            "assetType": "oaccount",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "bapAccountVersion" : 2,
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 0
        }
    },
    {
        "trxId":
"e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c",
        "timeStamp": 1649147442,
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion" : 1,
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    },
    {
        "trxId":
"d2d1f9c898707ae831e9361bc25da6369eac37b10c87dc04d18d6f3808222f08",
        "timeStamp": 1649137534,
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion" : 0,
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 0
        }
    }
]
```

Methods for Role Management

`addRole`
This method adds a role to a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async addRole(role: string, orgId: string, userId: string) {
   const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
   await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.addRoleMember',
'TOKEN');
```


Copyright © 2025, Oracle and/or its affiliates.

```
   return await this.Ctx.ERC721Token.addRoleMember(role, accountId);
}
```

Parameters:

- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Return Value Example:

```
{"msg": "Successfully added role 'minter' to Account Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin)"}
```

`removeRole`
This method removes a role from a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async removeRole(role: string, orgId: string, userId: string) {
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.removeRoleMember',
'TOKEN');
    return await this.Ctx.ERC721Token.removeRoleMember(role, accountId);
}
```

Parameters:

- `role: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message with account details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg": "Successfully removed role 'minter' from Account Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1)"}
```

getAccountsByRole
This method returns a list of all account IDs for a specified role. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async getAccountsByRole(role: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721ROLE.getAccountsByRole',
'TOKEN');
    return await this.Ctx.ERC721Role.getAccountsByRole(role);
}
```

Parameters:

- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
    "accounts": [

"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d"
    ]
}
```

getUsersByRole
This method returns a list of all users for a specified role. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async getUsersByRole(role: string) {
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721ROLE.getUsersByRole',
'TOKEN');
    return await this.Ctx.ERC721Role.getUsersByRole(role);
}
```

Parameters:

- `role: string` – The name of the role to search for.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON array of the user objects (`orgId` and `userId`).

Return Value Example:

```
{
    "users": [
        {
            "userId": "admin",
            "orgId": "Org1MSP"
        }
    ]
}
```

`isInRole`
This method returns a Boolean value to indicate if a user has a specified role. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async isInRole(orgId: string, userId: string, role: string) {
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.isInRole',
'TOKEN',{ accountId });
    return { result: await this.Ctx.ERC721Token.isInRole(role, accountId) };
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Return Value Example:

```
{"result":"true"}
```

Methods for Transaction History Management

`getAccountTransactionHistory`
This method returns account transaction history for a specified user. This is an asynchronous method. This method can be called only by the `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAccountTransactionHistory(orgId: string, userId: string) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721ACCOUNT.getAccountTransactionHis
tory', 'TOKEN', { accountId });
    return await
this.Ctx.ERC721Account.getAccountTransactionHistory(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Return Value Example:

```
[
    {
        "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
",
        "timestamp": "2022-04-05T09:30:44.000Z",
        "tokenId": "monalisa1",
        "noOfNfts": 1,
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "BURN"
    },
    {
        "transactionId":
"otransaction~a605f1fa62e511c2945fce5437f983a5e70ec814b82520d3ecd2d81e3ecf53a3
",
        "timestamp": "2022-04-05T09:30:22.000Z",
        "tokenId": "monalisa1",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    },
    {
        "transactionId":
"otransaction~ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7
",
        "timestamp": "2022-04-05T09:28:30.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "CREDIT"
    },
    {
        "transactionId":
"otransaction~cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb
",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "timestamp": "2022-04-05T09:05:45.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "DEBIT"
    },
    {
        "transactionId":
"otransaction~e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c
",
        "timestamp": "2022-04-05T08:30:42.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    }
]
```

`getAccountTransactionHistoryWithFilters`
This method returns account transaction history for a specified user, filtered by `PageSize`, `Bookmark`, `startTime` and `endTime`. This is an asynchronous method. This method can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by the `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.object().nullable())
public async getAccountTransactionHistoryWithFilters(orgId: string, userId:
string, filters ?: Filters) {
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    await
this.Ctx.ERC721Auth.checkAuthorization('ERC721ACCOUNT.getAccountTransactionHis
toryWithFilters', 'TOKEN', { accountId });
    return await
this.Ctx.ERC721Account.getAccountTransactionHistoryWithFilters(accountId,
filters)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `filters: object` – An object of the Filter class that contains four attributes: `pageSize`, `bookmark`, `startTime` and `endTime`.

Return Value Example:

```
[
    {
        "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
        "timestamp": "2022-04-05T09:30:44.000Z",
        "tokenId": "monalisa1",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "BURN"
    },
    {
        "transactionId":
"otransaction~a605f1fa62e511c2945fce5437f983a5e70ec814b82520d3ecd2d81e3ecf53a3
",
        "timestamp": "2022-04-05T09:30:22.000Z",
        "tokenId": "monalisa1",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    },
    {
        "transactionId":
"otransaction~ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7
",
        "timestamp": "2022-04-05T09:28:30.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "CREDIT"
    },
    {
        "transactionId":
"otransaction~cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb
",
        "timestamp": "2022-04-05T09:05:45.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "DEBIT"
    },
    {
        "transactionId":
"otransaction~e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c
",
        "timestamp": "2022-04-05T08:30:42.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

getTransactionById
This method returns transaction history for a specified transaction ID. This is an asynchronous method. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string())
public async getTransactionById(transactionId: string) {
    return await this.Ctx.ERC721Transaction.getTransactionById(transactionId);
}
```

Parameters:

- `transactionId: string` – The id of the transaction, which is the prefix `otransaction~` followed by the 64-bit hash in hexadecimal format.

Return Value Example:

```
{
    "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
",
    "history": [
        {
            "trxId":
"6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4",
            "timeStamp": 1649151044,
            "value": {
                "assetType": "otransaction",
                "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
",
                "tokenId": "monalisa1",
                "fromAccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "toAccountId": "",
                "triggeredByAccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "transactionType": "BURN",
                "timestamp": "2022-04-05T09:30:44.000Z",
            }
        }
    ]
}
```

deleteHistoricalTransactions
This method deletes transactions older than a specified time stamp in the state database. This is an asynchronous method. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.date())
public async deleteHistoricalTransactions(timeToExpiration: Date) {
    await
```


Copyright © 2025, Oracle and/or its affiliates.

```
this.Ctx.ERC721Auth.checkAuthorization('ERC721TRANSACTION.deleteTransactions',
 'TOKEN');
    return await
this.Ctx.ERC721Transaction.deleteTransactions(timeToExpiration);
}
```

Parameters:

- `timestamp: string` – A time stamp. All transactions before the time stamp will be deleted.

Return Value Example:

```
{
    "msg": "Successfuly deleted transaction older than date: Thu Apr 07 2022
21:18:59 GMT+0000 (Coordinated Universal Time).",
    "transactions": [

"otransaction~30513757d8b647fffaafac440d743635f5c1b2e41b25ebd6b70b5bbf78a2643f
",

"otransaction~ac0e908c735297941ba58bb208ee61ff4816a1e54c090d68024f82adf743892b
"
    ]
}
```

Methods for Token Behavior Management - Mintable Behavior

`create<Token Name>Token`
This method creates (mints) an NFT. The asset and associated properties are saved in the state database. The caller of this transaction must have a token account. The caller of this transaction becomes the owner of the NFT. If the token specification file includes the `roles` section for `behaviors` and the `minter_role_name` property for `roles`, then the caller of the transaction must have the minter role. Otherwise, any caller can mint NFTs.

```
@Validator(< Token Class >)
public async create< Token Name >Token(tokenAsset: <Token Class>) {
    return await this.Ctx.ERC721Token.createToken(tokenAsset);
}
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset to mint. For more information about the properties of the token asset, see the input specification file.

Returns:

- On success, a JSON token asset object that includes the following properties:
- `metadata` – JSON information that describes the token.
- `createdBy` – The account ID of the user who called the transaction to mint the token.
- `creationDate` – The time stamp of the transaction.
- `isBurned` – A Boolean value that indicates if the NFT identified by `tokenId` is burned.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenName` – The name of the token.
- `tokenDesc` – The description of the token.
- `symbol` – The symbol of the token.
- `tokenStandard` – The standard of the token.
- `tokenType` – The type of token held by this account.
- `tokenUnit` – The unit of the token.
- `behaviors` – A description of all token behaviors.
- `mintable` – A description of the properties of mintable behavior. The `max_mint_quantity` property specifies the maximum number of NFTs of this token class that can be created.
- `owner` – The account ID of the current owner of the token. During the minting process, the caller of this method becomes the owner of the token.
- `tokenUri` – The URI of the token.

Return Value Example:


```
{
    "metadata": {
        "painting_name": "Mona_Lisa",
        "description": "Mona Lisa Painting",
        "image": "monalisa.jpeg",
        "painter_name": "Leonardo_da_Vinci"
    },
    "assetType": "otoken",
    "tokenId": "monalisa",
    "tokenName": "artcollection",
    "tokenDesc": "token description",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-04-05T08:30:42.000Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "isBurned": false,
    "tokenUri": "\"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg\"",
    "price": 100,
    "on_sale_flag": false
}
```

update<Token Name>Token
This method updates token properties. This method can be called only by the user who is the owner or creator of the token. After a token asset is created, only the token owner can update the token custom properties. If the user is both token owner and creator of a token, they can also update the `TokenDesc` property. Token metadata cannot be updated. You must pass all token properties to this method, even if you want to update only certain properties.

```
@Validator(<Token Class>)
public async update<Token name>Token(tokenAsset: <Token Class>) {
    return await this.Ctx.ERC721Token.updateToken(tokenAsset);
}
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset to update. For more information about the properties of the token asset, see the input specification file.

Returns:

- On success, an updated JSON token asset object

Return Value Example:

```
{
    "metadata": {
        "painting_name": "Mona_Lisa",
        "description": "Mona Lisa Painting",
        "image": "monalisa.jpeg",
        "painter_name": "Leonardo_da_Vinci"
    },
    "assetType": "otoken",
    "tokenId": "monalisa",
    "tokenName": "artcollection",
    "tokenDesc": "token description",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-04-05T08:30:42.000Z",
    "isBurned": false,
    "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
    "price": 100,
    "on_sale_flag": true
}
```

Methods for Token Behavior Management - Transferable Behavior

`safeTransferFrom`
This is an asynchronous function. This method transfers ownership of the specified NFT from the caller to another account. This method includes the following validations:

- The token exists and is not burned.
- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.
- The caller of the function is the sender.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string(), yup.string().max(2000))
public async safeTransferFrom(fromOrgId: string, fromUserId: string, toOrgId:
string, toUserId: string, tokenId: string, data?: string) {
    const tokenAsset = await this.getTokenObject(tokenId);
    const fromAccountId = await
this.Ctx.ERC721Account.generateAccountId(fromOrgId, fromUserId);
    const toAccountId = await
this.Ctx.ERC721Account.generateAccountId(toOrgId, toUserId);
    return await this.Ctx.ERC721Token.safeTransferFrom(fromAccountId,
toAccountId, tokenAsset, data);
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the sender and token owner in the current organization.
- `fromUserId: string` – The user name or email ID of the sender and token owner.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenId: string` – The ID of the token to transfer.
- `data: string` – Optional additional information to store in the transaction record.

Returns:

- On success, a message with the sender and receiver account details.

Return Value Example:

```
{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin) to Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1)"}
```

`transferFrom`
This is an asynchronous function. This method transfers ownership of the specified NFT from a sender account to a receiver account. It is the responsibility of the caller to pass the correct parameters. This method can be called by any user, not only the token owner. This method includes the following validations:

- The token exists and is not burned.
- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string())
public async transferFrom(fromOrgId: string, fromUserId: string, toOrgId:
string, toUserId: string, tokenId: string) {
    const tokenAsset = await this.getTokenObject(tokenId);
    const fromAccountId = await
this.Ctx.ERC721Account.generateAccountId(fromOrgId, fromUserId);
    const toAccountId = await
this.Ctx.ERC721Account.generateAccountId(toOrgId, toUserId);
    return await this.Ctx.ERC721Token.transferFrom(fromAccountId,
toAccountId, tokenAsset);
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the sender in the current organization.
- `fromUserId: string` – The user name or email ID of the sender.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.
- `tokenId: string` – The ID of the token to transfer.

Returns:

- On success, a message with the sender and receiver account details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1) to Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin)"}
```

Methods for Token Behavior Management - Burnable Behavior

`burn`
This method deactivates, or burns, the specified NFT from the caller's account. The caller of this method must have an account. A token cannot be burned unless the token specification file includes the `burnable` behavior. If no `burner_role_name` property is specified in the `roles` section of the specification file, then the owner of the token can burn the token. If a `burner_role_name` property is specified in the `roles` section, then the user assigned the burner role who is also the minter (creator) or owner of the token can burn the token.

```
@Validator(yup.string())
public async burn(tokenId: string) {
    const tokenAsset = await this.getTokenObject(tokenId);
    return await this.Ctx.ERC721Token.burn(tokenAsset);
}
```

Parameters:

- `tokenId: string` – The ID of the token to burn.

Returns:

- On success, a message with the account details.

Return Value Example:

```
{"msg": "Successfully burned NFT token: 'monalisa1' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin)"}
```

`burnNFT`
This method deactivates, or burns, the specified NFT from the caller's account, and returns a token object and token history. The caller of this method must have an account. A token cannot be burned unless the token specification file includes the `burnable` behavior. If no `burner_role_name` property is specified in the `roles` section of the specification file, then the owner of the token can burn the token. If a `burner_role_name` property is specified in the `roles` section, then the user assigned the burner role who is also the minter (creator) or owner of the token can burn the token.

```
@Validator(yup.string())
public async burnNFT(tokenId: string) {
    const token = await this.Ctx.ERC721Token.get(tokenId)
    if (token.isBurned === true) {
      throw new Error(`token with tokenId ${tokenId} is already burned`);
    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    const tokenHistory = await this.Ctx.ERC721Token.history(tokenId);
    await this.burn(tokenId);
    token.tokenId = parseInt(token.tokenId);
    if(Number.isNaN(token.tokenId)) {
      throw new Error(`tokenId is expected to be integer but found $
{tokenId}`)
    }
    token.isBurned = true;
    return {...token, tokenHistory: JSON.stringify(tokenHistory)};
}
```

Parameters:

- `tokenId: string` – The ID of the token to burn.

Returns:

- On success, a token object that includes token history information.

Return Value Example:

```
{
    "assetType": "otoken",
    "tokenId": 1,
    "tokenName": "artcollection",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2023-08-22T07:36:50.000Z",
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "isBurned": true,
    "tokenUri": "example.com",
    "price": 120,
    "on_sale_flag": false,
    "tokenHistory":
"[{\"trxId\":\"732438a85b5e8fc76c5254e54602b29d583543b103fafb5a28c0df384428bb5
0\",\"timeStamp\":\"2023-08-22T07:36:50.000Z\",\"value\":
```


Copyright © 2025, Oracle and/or its affiliates.

{\"assetType\":\"otoken\",\"tokenId\":\"1\",\"tokenName\":\"artcollection\",\"
symbol\":\"ART\",\"tokenStandard\":\"erc721+
\",\"tokenType\":\"nonfungible\",\"tokenUnit\":\"whole\",\"behaviors\":
[\"indivisible\",\"singleton\",\"mintable\",\"transferable\",\"burnable\",\"ro
les\"],\"roles\":{\"minter_role_name\":\"minter\"},\"mintable\":
{\"max_mint_quantity\":20000},\"createdBy\":\"oaccount~42e89f4c72dfde950281487
6423c6da630d466e87436dd1aae201d347ad1288d\",\"creationDate\":\"2023-08-22T07:3
6:50.000Z\",\"owner\":\"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436d
d1aae201d347ad1288d\",\"isBurned\":false,\"tokenUri\":\"example.com\",\"price\
":120,\"on_sale_flag\":false}}]"
}

Custom Methods

You can use the token SDK methods to write custom methods for your business application.

The following example shows how to use token SDK methods in custom methods. When the `sell` method is called, it posts a token for sale for a specified price.

```
@Validator(yup.string(), yup.number())
public async sell(token_id: string, selling_price: number) {
    try {
        const token = await this.Ctx.ERC721Token.get(token_id);
        const t = new ArtCollection(token)
 /**  * price is a custom asset
      attribute to set the price of a non-fungible token in the
      marketplace  */
        t.price =  selling_price;
 /**  * on_sale_flag is a
      custom asset attribute that maintains non-fungible token selling status
in the
      marketplace  */
        t.on_sale_flag = true;
        await this.Ctx.ERC721Token.updateToken(t);
        let msg = `Token ID : '${token_id}' has been posted for selling in
the marketplace'`;
        return {msg}
        } catch(error) {
            throw new Error(error.message);
    }
}
```

NFT SDK Methods

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior

Copyright © 2025, Oracle and/or its affiliates.

– Burnable Behavior

Methods for Access Control Management

The NFT SDK provides an access control function. Some methods can be called only by a `Token Admin` or `Account Owner` of the token. You can use this feature to ensure that operations are carried out only by the intended users. Any unauthorized access results in an error. To use the access control function, import the `Authorization` class from the `../lib/erc721-auth` module.

```
import { ERC721Authorization } from '../lib/erc721-auth';
```

`addAdmin`
This method adds a user as a `Token Admin` of the token chaincode.

```
Ctx.ERC721Admin.addAdmin(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user added as a `Token Admin` of the token chaincode. On error, a non-nil error object that contains an error message.

Example:

```
await this.Ctx.ERC721Admin.addAdmin(orgId, userId)

{"msg": "Successfully added Admin (orgId: Org1MSP, userId: user1)"}
```

`removeAdmin`
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.ERC721Admin.removeAdmin(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user removed as a `Token Admin` of the token chaincode. On error, a non-nil error object that contains an error message.

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
await this.Ctx.ERC721Admin.RemoveAdmin(orgId, userId)
```

```
{"msg": "Successfully removed Admin (orgId: Org1MSP, userId: user1)"}
```

isUserTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`. This is an asynchronous static function.

```
Ctx.ERC721Auth.isUserTokenAdmin(orgId: string, userId: string)
```

Parameters:

- `orgId` – The membership service provider (MSP) ID of the user in the current network organization.
- `userId` – The user name or email ID of the user.

Returns:

- A Boolean response and an error message if an error is encountered.

Example:

```
await this.Ctx.Auth.isUserTokenAdmin('Org1MSP', 'user1');
```

```
{"result":false}
```

getAllAdmins
This method returns a list of all `Token Admin` users.

```
Ctx.ERC721Admin.getAllAdmins()
```

Parameters:

- none

Returns:

- On success, a list of all `Token Admin` users. On error, a non-nil error object that contains an error message.

Example:

```
await this.Ctx.ERC721Admin.getAllAdmins()
```

```
{
    "admins": [
        {
            "orgId": "Org1MSP",
            "userId": "admin"
        }
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

checkAuthorization
Use this method to add an access control check to an operation. This is an asynchronous function. Most automatically generated methods include access control. Certain token methods can be run only by the `ERC721Admin` or `Account Owner` of the token or by the `MultipleAccountOwner` for users with multiple accounts. The `checkAuthorization` method is part of the `Authorization` class, which you access via the `Ctx` object. The access control mapping is described in the `../lib/constant.ts` file, as shown in the following text. You can modify access control by editing the `../lib/constant.ts` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
export const TOKENACCESS = {
  ADMIN: {
    isUserTokenAdmin: ['Admin', 'MultipleAccountOwner'],
    addAdmin: ['Admin'],
    removeAdmin: ['Admin'],
    getAllAdmins: ['Admin'],
  },
  TOKEN: {
    save: ['Admin'],
    getAllTokens: ['Admin'],
    get: ['Admin'],
    update: ['Admin'],
    getDecimals: ['Admin'],
    getTokensByName: ['Admin'],
    addRoleMember: ['Admin'],
    removeRoleMember: ['Admin'],
    isInRole: ['Admin', 'AccountOwner'],
    getTotalMintedTokens: ['Admin'],
    getNetTokens: ['Admin'],
  },
  ROLE: {
    getAccountsByRole: ['Admin'],
    getUsersByRole: ['Admin'],
  },
  TRANSACTION: {
    deleteTransactions: ['Admin'],
  },
  ACCOUNT: {
    createAccount: ['Admin'],
    getAllAccounts: ['Admin'],
    getAccountsByUser: ['Admin', 'MultipleAccountOwner'],
    getAccount: ['Admin', 'AccountOwner'],
    history: ['Admin', 'AccountOwner'],
    getAccountTransactionHistory: ['Admin', 'AccountOwner'],
    getAccountBalance: ['Admin', 'AccountOwner'],
    getAccountOnHoldBalance: ['Admin', 'AccountOwner'],
    getOnHoldIds: ['Admin', 'AccountOwner'],
  },
  ERC721ADMIN: {
    isUserTokenAdmin: ['Admin'],
    addAdmin: ['Admin'],
    removeAdmin: ['Admin'],
    getAllAdmins: ['Admin'],
```


Copyright © 2025, Oracle and/or its affiliates.

```
  },
  ERC721TOKEN: {
    getAllTokens: ['Admin'],
    getAllTokensByUser: ['Admin', 'AccountOwner'],
    get: ['Admin', TOKEN_OWNER],
    getTokensByName: ['Admin'],
    addRoleMember: ['Admin'],
    removeRoleMember: ['Admin'],
    isInRole: ['Admin', 'AccountOwner'],
    totalSupply: ['Admin'],
    totalNetSupply: ['Admin'],
    history: ['Admin'],
  },
  ERC721ROLE: {
    getAccountsByRole: ['Admin'],
    getUsersByRole: ['Admin'],
  },
  ERC721TRANSACTION: {
    deleteTransactions: ['Admin'],
  },
  ERC721ACCOUNT: {
    createAccount: ['Admin'],
    getAllAccounts: ['Admin'],
    getAccountsByUser: ['Admin', 'MultipleAccountOwner'],
    history: ['Admin', 'AccountOwner'],
    getAccountTransactionHistory: ['Admin', 'AccountOwner'],
    getAccountTransactionHistoryWithFilters: ['Admin', 'AccountOwner'],
    balanceOf: ['Admin', 'MultipleAccountOwner'],
  }
}
```

`Ctx.ERC721Auth.checkAuthorization(classFuncName: string, ...args)`

Parameters:

- `classFuncName: string` – The map value between the class and methods as described in the `../lib/constant.ts` file.
- `...args` – A variable argument where `args[0]` takes the constant `'TOKEN'` and `args[1]` takes the `accountId` parameter to add an access control check for an `AccountOwner`. To add an access control check for a `MultipleAccountOwner`, `args[1]` takes the `orgId` parameter and `args[2]` takes the `userId` parameter.

Returns:

- On success, a promise. On error, a rejection with an error message.

Examples:
`Admin` access

```
await this.Ctx.ERC721Auth.checkAuthorization('ADMIN.addAdmin', 'TOKEN');
```


Copyright © 2025, Oracle and/or its affiliates.

`AccountOwner` access

```
await this.Ctx.ERC721Auth.checkAuthorization('ACCOUNT.getAccountBalance',
'TOKEN', accountId);
```

`MultipleAccountOwner` access

```
await this.Ctx.ERC721Auth.checkAuthorization('ADMIN.isUserTokenAdmin',
'TOKEN', orgId, userId);
```

Methods for Token Configuration Management

The token configuration management methods are based on the ERC-721 standard. To use the token configuration management methods, import the `Token` class from the `../lib/erc721-token` module.

`createToken`
This method creates a token and saves its properties in the state database. This method can be called only by users with the minter role. This is an asynchronous function.

```
Ctx.ERC721Token.createToken(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to create.

Returns:

- On success, a promise message with token details. On error, a rejection with an error message.

Example:

```
await this.Ctx.ERC721Token.createToken(tokenAsset);
```

```
{
    "metadata": {
        "painting_name": "Mona_Lisa",
        "description": "Mona Lisa Painting",
        "image": "monalisa.jpeg",
        "painter_name": "Leonardo_da_Vinci"
    },
    "assetType": "otoken",
    "tokenId": "monalisa",
    "tokenName": "artcollection",
    "tokenDesc": "token description",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-04-05T08:30:42.000Z",
    "isBurned": false,
    "tokenUri": "\"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg\"",
    "price": 100,
    "on_sale_flag": false
}
```

getAllTokens

This method returns all of the token assets that are saved in the state database. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This is an asynchronous static function.

```
Ctx.ERC721Token.getAllTokens()
```

Parameters:

- none

Returns:

- On success, a promise with all of the token assets. On error, a rejection with an error message.

Example:

```
await this.Ctx.ERC721Token.getAllTokens();
```

```
{
   "returnCode":"Success",
   "error":"",
   "result":{

"txid":"98e0a0a115803d25b843d630e6b23c435a192a03eb0a301fc9375f05da49a8b2",
      "payload":[
         "           "{
            "metadata":{
               "painting_name":"Mona_Lisa",
               "description":"Mona Lisa Painting",
               "image":"monalisa.jpeg",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                  "painter_name":"Leonardo_da_Vinci"
               },
               "assetType":"otoken",
               "tokenId":"monalisa",
               "tokenName":"artcollection",
               "tokenDesc":"token description",
               "symbol":"ART",
               "tokenStandard":"erc721+",
               "tokenType":"nonfungible",
               "tokenUnit":"whole",
               "behaviors":[
                  "indivisible",
                  "singleton",
                  "mintable",
                  "transferable",
                  "burnable",
                  "roles"
               ],
               "roles":{
                  "minter_role_name":"minter"
               },
               "mintable":{
                  "max_mint_quantity":20000
               },

"owner":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1
288d",

"createdBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d34
7ad1288d",
               "creationDate":"2022-04-05T08:30:42.000Z",
               "isBurned":false,
               "tokenUri":"\"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg\"",
               "price":100,
               "on_sale_flag":false
            }"       "
         ],
         "encode":"JSON"
      }
}
```

getAllTokensByUser
This method returns all tokens that are owned by a specified account ID. This is an asynchronous static function.

```
Ctx.ERC721Token.getAllTokensByUser(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise history query iterator for the specified account. On error, a rejection with an error message.

Return Value Example:


```
{
   "returnCode":"Success",
   "error":"",
   "result":{

"txid":"98e0a0a115803d25b843d630e6b23c435a192a03eb0a301fc9375f05da49a8b2",
      "payload":[
         "           "{
            "metadata":{
               "painting_name":"Mona_Lisa",
               "description":"Mona Lisa Painting",
               "image":"monalisa.jpeg",
               "painter_name":"Leonardo_da_Vinci"
            },
            "assetType":"otoken",
            "tokenId":"monalisa",
            "tokenName":"artcollection",
            "tokenDesc":"token description",
            "symbol":"ART",
            "tokenStandard":"erc721+",
            "tokenType":"nonfungible",
            "tokenUnit":"whole",
            "behaviors":[
               "indivisible",
               "singleton",
               "mintable",
               "transferable",
               "burnable",
               "roles"
            ],
            "roles":{
               "minter_role_name":"minter"
            },
            "mintable":{
               "max_mint_quantity":20000
            },

"owner":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1
288d",

"createdBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d34
7ad1288d",
            "creationDate":"2022-04-05T08:30:42.000Z",
            "isBurned":false,
            "tokenUri":"\"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg\"",
            "price":100,
            "on_sale_flag":false
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }"       "
      ],
      "encode":"JSON"
   }
```

getTokenUri
This method returns the URI for a specified token. This is an asynchronous static function.

```
Ctx.ERC721Token.getTokenUri(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, it returns a promise history query iterator for the specified token ID. On error, it rejects with an error message

Return Value Example:

```
{"tokenUri": https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"}
```

history
This method returns history for the specified token. This is an asynchronous static function.

```
Ctx.ERC721Token.history(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a promise history query iterator for the specified token. On error, a rejection with an error message.

Return Value Example:

```
[
   {

"trxId":"e17a3154d5271be0492cbc7c12390b3480fec5a792d1cb1083e5335de56ebbd9",
      "timeStamp":1622614032,
      "isDelete":false,
      "value":{
         "metadata":{
            "painting_name":"Mona_Lisa",
            "description":"Mona Lisa Painting",
            "image":"monalisa.jpeg",
            "painter_name":"Leonardo_da_Vinci"
```


Copyright © 2025, Oracle and/or its affiliates.

```
            },
            "assetType":"otoken",
            "tokenId":"monalisa",
            "tokenName":"artcollection",
            "tokenDesc":"token description",
            "symbol":"ART",
            "tokenStandard":"erc721+",
            "tokenType":"nonfungible",
            "tokenUnit":"whole",
            "behaviors":[
               "indivisible",
               "singleton",
               "mintable",
               "transferable",
               "burnable",
               "roles"
            ],
            "roles":{
               "minter_role_name":"minter"
            },
            "mintable":{
               "max_mint_quantity":20000
            },

"owner":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1
288d",

"createdBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d34
7ad1288d",
            "creationDate":"2022-04-05T08:30:42.000Z",
            "isBurned":false,
            "tokenUri":"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "price":100,
            "on_sale_flag":"true"
         }
      },
      {

"trxId":"dbcc4da410ad4d4a80996f090b313240f3f3d08aa2b5086afa8d0921f7b4c1e5",
         "timeStamp":1622643853,
         "isDelete":false,
         "value":{
            "metadata":{
               "painting_name":"Mona_Lisa",
               "description":"Mona Lisa Painting",
               "image":"monalisa.jpeg",
               "painter_name":"Leonardo_da_Vinci"
            },
            "assetType":"otoken",
            "tokenId":"monalisa",
            "tokenName":"artcollection",
            "tokenDesc":"token description",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "symbol":"ART",
            "tokenStandard":"erc721+",
            "tokenType":"nonfungible",
            "tokenUnit":"whole",
            "behaviors":[
               "indivisible",
               "singleton",
               "mintable",
               "transferable",
               "burnable",
               "roles"
            ],
            "roles":{
               "minter_role_name":"minter"
            },
            "mintable":{
               "max_mint_quantity":20000
            },

"owner":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1
288d",

"createdBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d34
7ad1288d",

"transferredBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae20
1d347ad1288d",
            "creationDate":"2022-04-05T08:30:42.000Z",
            "transferredDate":"2022-04-05T09:28:30.000Z",
            "isBurned":false,
            "tokenUri":"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            "price":100,
            "on_sale_flag":true
         }
      }
]
```

get
This method returns the specified token object if it is present in the state database. This is an asynchronous static function.

```
Ctx.ERC721Token.get(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a promise that includes a JSON object of the token asset. On error, a rejection with an error message

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
await this.Ctx.ERC721Token.get(tokenId);

{
    "metadata": {
        "painting_name": "Mona_Lisa",
        "description": "Mona Lisa Painting",
        "image": "monalisa.jpeg",
        "painter_name": "Leonardo_da_Vinci"
    },
    "assetType": "otoken",
    "tokenId": "monalisa",
    "tokenName": "artcollection",
    "tokenDesc": "token description",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "transferredBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-04-05T08:30:42.000Z",
    "transferredDate": "2022-04-05T09:28:30.000Z",
    "isBurned": false,
    "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
    "price": 100,
    "on_sale_flag": true
}
```

ownerOf

This method returns the account ID of the owner of a specified token. This is an asynchronous static function.

```
Ctx.ERC721Token.ownerOf(tokenId: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `tokenId: string` – The ID of the token.

Returns:

- On success, it returns a promise history query iterator for the specified token ID. On error, it rejects with an error message

Return Value Example:

```
{"owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d"}
```

`symbol`
This method returns the symbol of the token class.

```
Ctx.ERC721Token.symbol()
```

Parameters:

- none

Returns:

- On success, a JSON object with the token symbol.

Return Value Example:

```
{"symbol": "PNT"}
```

`tokenUri`
This method returns the URI for a specified token. This is an asynchronous static function.

```
Ctx.ERC721Token.tokenUri(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, it returns a promise history query iterator for the specified token ID. On error, it rejects with an error message

Return Value Example:

```
{"uri": "https://bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq
https://bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\
\ .ipfs.infura-ipfs.io/?filename=MonaLisa.jpeg"}
```

`totalSupply`
This method returns the total number of minted NFTs. This is an asynchronous function.

```
Ctx.ERC721Token.totalSupply()
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- none

Returns:

- On success, the total net tokens, in the number data type. On error, it returns with an error message.

Example:

```
await this.Ctx.ERC721Token.totalSupply(tokenAsset);

2000
```

`updateToken`

This method updates token properties. This method can be called only by the owner or creator of the token. After a token asset is created, only the token owner can update the token custom properties. If the user is both token owner and creator of a token, they can also update the `TokenDesc` property. Token metadata cannot be updated. You must pass all token properties to this method, even if you want to update only certain properties. This is an asynchronous function.

```
Ctx.ERC721Token.updateToken(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to update.

Returns:

- On success, a promise message with token details. On error, a rejection with an error message.

Example:

```
await this.Ctx.ERC721Token.updateToken(tokenAsset)

{
    "metadata": {
        "painting_name": "Mona_Lisa",
        "description": "Mona Lisa Painting",
        "image": "monalisa.jpeg",
        "painter_name": "Leonardo_da_Vinci"
    },
    "assetType": "otoken",
    "tokenId": "monalisa",
    "tokenName": "artcollection",
    "tokenDesc": "token description",
    "symbol": "ART",
    "tokenStandard": "erc721+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "transferable",
            "burnable",
            "roles"
        ],
        "roles": {
            "minter_role_name": "minter"
        },
        "mintable": {
            "max_mint_quantity": 20000
        },
        "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "creationDate": "2022-04-05T08:30:42.000Z",
        "isBurned": false,
        "tokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
        "price": 100,
        "on_sale_flag": true
}
```

getByRange

This method calls the fabric `getStateByRange` method internally. Even though any asset with the given ID is returned from the ledger, this method casts the asset into the caller Asset type. This is an asynchronous static function.

```
@validator(yup.string(), yup.string())
public async getDigiCurrGetByRange(startId: string, endId: string) {
   return await this.Ctx.ERC721TOken.getByRange(startId, endId, PaintingNft);
}
```

```
Ctx.ERC721Token.getByRange(startId: string, endId: string,
tokenClassReference?: <Instance of Token Class> )
```

Parameters:

- `startId: string` – The starting key of the range. This key is included in the range.
- `endId: string` – The end key of the range. This key is excluded from the range.
- `tokenClassReference: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, a promise with an array of `<Token Class>`. On error, a rejection with an error message.

Return Value Example:

```
[
  {
      "metadata":{
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "painting_name":"Mona_Lisa",
            "description":"Mona Lisa Painting",
            "image":"monalisa.jpeg",
            "painter_name":"Leonardo_da_Vinci"
         },
         "assetType":"otoken",
         "tokenId":"monalisa",
         "tokenName":"artcollection",
         "tokenDesc":"token description",
         "symbol":"ART",
         "tokenStandard":"erc721+",
         "tokenType":"nonfungible",
         "tokenUnit":"whole",
         "behaviors":[
            "indivisible",
            "singleton",
            "mintable",
            "transferable",
            "burnable",
            "roles"
         ],
         "roles":{
            "minter_role_name":"minter"
         },
         "mintable":{
            "max_mint_quantity":20000
         },

"owner":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1
288d",

"createdBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d34
7ad1288d",

"transferredBy":"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae20
1d347ad1288d",
         "creationDate":"2022-04-05T08:30:42.000Z",
         "transferredDate":"2022-04-05T09:28:30.000Z",
         "isBurned":false,
         "tokenUri":"https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
         "price":100,
         "on_sale_flag":true
      }
]
```

isTokenType
This method indicates whether a token asset exists with the specified ID. This is an asynchronous static function.

```
Ctx.ERC721Token.isTokenType(tokenId: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `tokenId: string` – The ID of the token.

Returns:

- On success, a promise with `true` if a token asset exists with the specified ID. On error, a rejection with an error message.

Example:

```
await this.Ctx.ERC721Token.isTokenType(tokenId);

true
```

Methods for Account Management

`generateAccountId`

This method returns an account ID, which is formed by concatenating the membership service provider ID (`orgId`) and the user name or email ID (`userId`) and then creating a SHA-256 hash.

```
Ctx.ERC721Account.generateAccountId(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a promise with the generated account ID. On error, a rejection with an error message.

Example:

```
await this.Ctx.ERC721Account.generateAccountId(orgId, userId)

oaccount~a0a60d54ba9e2ff349737d292ea10ebd9cc8f1991c11443c19d20aea299a9507
```

`createAccount`

This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track the number of NFTs a user has. Users must have accounts in the network to complete token-related operations. You can create only one NFT account per user.

An account ID is an alphanumeric set of characters, prefixed with `oaccount~` and followed by an SHA-256 hash of the membership service provider ID (`orgId`) of the user in the current network organization, the user name or email ID (`userId`) of the instance owner or the user who is logged in to the instance, and the constant string `nft`. This method can be called only by the `Token Admin` of the chaincode.

```
Ctx.ERC721Account.createAccount(orgId: string, userId: string, tokenType:
string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenType: string` – The only supported token type is `nonfungible`.

Returns:

- On success, a promise with the new account object. On error, a rejection with an error message

Example:

```
await this.Ctx.ERC721Account.CreateAccount(orgId, userId, tokenType)

{
    "assetType": "oaccount",
    "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "userId": "admin",
    "orgId": "Org1MSP",
    "tokenType": "nonfungible",
    "noOfNfts": 0
}
```

`balanceOf`

This method returns the total number of NFTs the specified user holds.

```
Ctx.ERC721Account.balanceOf(accountId: string)
```

Parameters:

- `accountId: string` – The account ID of the user.

Returns:

- On success, a JSON object of the current NFT count. On error, a non-nil error object that contains an error message.

Example:

```
await this.Ctx.ERC721Account.balanceOf(accountId)

{"totalNfts": 0}
```

`getAllAccounts`

This method returns a list of all accounts. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC721Account.getAllAccounts()
```

Parameters:

- none

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise with a JSON object that lists all accounts. On error, a rejection with an error message.

Return Value Example:

```
[
    {
        "key":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "valueJson": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    }
]
```

`getAccountByUser`
This method returns user details for a specified account. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC721Account.getAccountByUser(orgId, userId)
```

Parameters:

- `orgId` – The membership service provider (MSP) ID of the user in the current organization.
- `userId` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `accountId` – The ID of the user account.
- `userId` – The user name or email ID of the user.
- `orgId` – The membership service provider (MSP) ID of the user in the current organization.
- `tokenType` – The type of token that the account holds.
- `noOfNfts` – The total number of NFTs held by the account.
- On error, a non-nil object that contains an error message.

Example:
```
await this.Ctx.ERC721Account.getUserByAccountById(orgId, userId)
```

```
{
    "assetType": "oaccount",
    "accountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "userId": "admin",
    "orgId": "Org1MSP",
    "tokenType": "nonfungible",
    "noOfNfts": 0
}
```

`getUserByAccountId`
This method returns the user details for a specified account.

```
Ctx.ERC721Account.getUserByAccountId(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a JSON object that includes user details in the following properties:
  - `orgId` – The membership service provider (MSP) ID of the user in the current network organization.
  - `userId` – The user name or email ID of the user.
- On error, a rejection with an error message.

Example:

```
await this.Ctx.ERC721Account.getUserByAccountById(accountId)
```

```
{
  "userId": "admin",
  "orgId": "Org1MSP"
}
```

`history`
This method returns an array of the account history details for a specified account.

```
Ctx.ERC721Account.history(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a `map[string]interface{}` array that contains the account history details for the specified account. The account data is shown under the `value` key in the map. On error, a non-nil error object containing an error message.

Example:

```
await this.Ctx.ERC721Account.history(accountId)
```

```
[
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "trxId":
"6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4",
        "timeStamp": 1649151044,
        "value": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    },
    {
        "trxId":
"a605f1fa62e511c2945fce5437f983a5e70ec814b82520d3ecd2d81e3ecf53a3",
        "timeStamp": 1649151022,
        "value": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 2
        }
    },
    {
        "trxId":
"ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7",
        "timeStamp": 1649150910,
        "value": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    },
    {
        "trxId":
"cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb",
        "timeStamp": 1649149545,
        "value": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 0
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "trxId":
"e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c",
        "timeStamp": 1649147442,
        "value": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 1
        }
    },
    {
        "trxId":
"d2d1f9c898707ae831e9361bc25da6369eac37b10c87dc04d18d6f3808222f08",
        "timeStamp": 1649137534,
        "value": {
            "assetType": "oaccount",
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "userId": "admin",
            "orgId": "Org1MSP",
            "tokenType": "nonfungible",
            "noOfNfts": 0
        }
    }
]
```

`getAccountWithStatusByUser`
This method returns user details for a specified account, including account status. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC721Account.getAccountWithStatusByUser(orgId, userId)
```

Parameters:

- `orgId` – The membership service provider (MSP) ID of the user in the current organization.
- `userId` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `accountId` – The ID of the user account.
- `userId` – The user name or email ID of the user.
- `orgId` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenType` – The type of token that the account holds.
- `noOfNfts` – The total number of NFTs held by the account.
- `bapAccountVersion` – An account object parameter for internal use.
- `status` – The current status of the user account.
- On error, a non-nil object that contains an error message.

Example:

```
await this.Ctx.ERC721Account.getAccountWithStatusByUser(orgId, userId)

{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "accountId":
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
  "userId": "idcqa",
  "orgId": "appdev",
  "tokenType": "nonfungible",
  "noOfNfts": 0
}
```

Methods for Role Management

`addRoleMember`

This method adds a role to a specified user and token. An account ID is formed by creating an SHA-256 hash of the concatenated membership service provider ID (`orgId`) and the user name or email ID (`userId`). This is an asynchronous function.

```
Ctx.ERC721Token.addRoleMember(role: string, accountId: string)
```

Parameters:

- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `accountId: string` – The account ID to operate on.

Returns:

- On success, a promise with a message including the added role and account ID. On error, a rejection with an error message

Example:

```
await this.Ctx.ERC721Token.addRoleMember(role, accountId);

{"msg": "Successfully added role 'minter' to Account Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin)"}
```


Copyright © 2025, Oracle and/or its affiliates.

`removeRoleMember`
This method removes a role from a specified user and token. An account ID is formed by creating an SHA-256 hash of the concatenated membership service provider ID (`orgId`) and the user name or email ID (`userId`). This is an asynchronous function.

```
Ctx.ERC721Token.removeRoleMember(role: string, accountId: string)
```

Parameters:

- `role: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `accountId: string` – The account ID to operate on.

Returns:

- On success, a promise with a message including the removed role and account ID. On error, a rejection with an error message

Example:

```
await this.Ctx.ERC721Token.removeRoleMember(role, accountId);

{"msg": "Successfully removed role 'minter' from Account Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1)"}
```

`getAccountsByRole`
This method returns a list of all account IDs for a specified role.

```
Ctx.ERC721Role.getAccountsByRole(roleName: string)
```

Parameters:

- `roleName: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs. On error, a non-nil error object that contains an error message.

Example:

```
await this.Ctx.ERC721Role.getAccountsByRole(userRole)

{
    "accounts": [

"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d"
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getUsersByRole`
This method returns a list of all users for a specified role.

```
Ctx.ERC721Role.getUsersByRole(userRole: string)
```

Parameters:

- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON array of user objects. Each object contains the user ID and organization ID. On error, a non-nil error object that contains an error message.

Example:
```
await this.Ctx.ERC721Role.getUsersByRole(userRole)
```

```
{
    "users": [
        {
            "userId": "admin",
            "orgId": "Org1MSP"
        }
    ]
}
```

`isInRole`
This method returns a Boolean value to indicate if a user and token has a specified role. An account ID is formed by creating an SHA-256 hash of the concatenated membership service provider ID (`orgId`) and the user name or email ID (`userId`). This is an asynchronous function.

```
Ctx.ERC721Token.isInRole(role: string, accountId: string)
```

Parameters:

- `role: string` – The name of the role to check for the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `accountId: string` – The account ID to operate on.

Returns:

- On success, a promise that is true if the role is present for the specified account ID, otherwise false. On error, a rejection with an error message

Example:
```
await this.Ctx.ERC721Token.isInRole(role, accountId, tokenAsset)
```

```
{"result": "true"}
```

Methods for Transaction History Management

Copyright © 2025, Oracle and/or its affiliates.

`getAccountTransactionHistory`
This method returns an array of the transaction history details for a specified account.

```
Ctx.ERC721Account.getAccountTransactionHistory(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, an array of account transaction objects in JSON format:
  - `transactionId` – The ID of the transaction.
  - `transactedAccount` – The account with which the transaction took place.
  - `transactionType` – The type of transaction.
  - `timestamp` – The time of the transaction.
  - `noOfNfts` – The balance of the caller account.
  - On error, a non-nil error object that contains an error message.

Example:

```
await this.Ctx.ERC721Account.GetAccountTransactionHistory(accountId)

[
    {
        "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
",
        "timestamp": "2022-04-05T09:30:44.000Z",
        "tokenId": "monalisa1",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "BURN"
    },
    {
        "transactionId":
"otransaction~a605f1fa62e511c2945fce5437f983a5e70ec814b82520d3ecd2d81e3ecf53a3
",
        "timestamp": "2022-04-05T09:30:22.000Z",
        "tokenId": "monalisa1",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    },
    {
        "transactionId":
"otransaction~ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7
",
        "timestamp": "2022-04-05T09:28:30.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transactionType": "CREDIT"
    },
    {
        "transactionId":
"otransaction~cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb
",
        "timestamp": "2022-04-05T09:05:45.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "DEBIT"
    },
    {
        "transactionId":
"otransaction~e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c
",
        "timestamp": "2022-04-05T08:30:42.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    }
]
```

`getAccountTransactionHistoryWithFilters`
This method returns account transaction history for a specified user, filtered by `PageSize`, `Bookmark`, `startTime` and `endTime`. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
async getAccountTransactionHistoryWithFilters(orgId: string, userId: string,
filters?: Filters)
```

Parameters:

- `accountId: string` – The ID of the account.
- `filters: object` – An object of the Filter class that contains four attributes: `pageSize`, `bookmark`, `startTime` and `endTime`. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Returns:

- On success, an array of account transaction objects in JSON format:
  - `transactionId` – The ID of the transaction.
  - `transactedAccount` – The account with which the transaction took place.
  - `transactionType` – The type of transaction.
  - `timestamp` – The time of the transaction.

Copyright © 2025, Oracle and/or its affiliates.

- `noOfNfts` – The balance of the caller account.
- On error, a non-nil error object that contains an error message.

Example:

```
await this.Ctx.ERC721Account.getAccountTransactionHistoryWithFilters(accountId,
filters)
```

```
[
    {
        "transactionId":
"otransaction~ca4c07bf04240345de918cbf1f4f3da4b4d0ab044c5b8bea94343e427d9ed4e7
",
        "timestamp": "2022-04-05T09:28:30.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "CREDIT"
    },
    {
        "transactionId":
"otransaction~cfb52ffc8c34c7fd86210fcf8c5f53d9f92a056c45ed3a33671d638020c1f9cb
",
        "timestamp": "2022-04-05T09:05:45.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
        "transactionType": "DEBIT"
    },
    {
        "transactionId":
"otransaction~e7747b3001a170f88688620956320e9402e1dd8edad8afb4818a08a34647337c
",
        "timestamp": "2022-04-05T08:30:42.000Z",
        "tokenId": "monalisa",
        "transactedAccount":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "transactionType": "MINT"
    }
]
```

`getTransactionById`

This method returns the history of a `Transaction` asset.

```
Ctx.ERC721Transaction.getTransactionById(transactionId: string)
```

Parameters:

- `transactionId: string` – The ID of the transaction asset.

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
await this.Ctx.ERC721Transaction.getTransactionById(transactionId)

{
    "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
",
    "history": [
        {
            "trxId":
"6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4",
            "timeStamp": 1649151044,
            "value": {
                "assetType": "otransaction",
                "transactionId":
"otransaction~6ffd0d94f234c12444a5d5aa559563b59dff4d2280b573fea956dc632bdaf5d4
",
                "tokenId": "monalisa1",
                "fromAccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "toAccountId": "",
                "triggeredByAccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "transactionType": "BURN",
                "timestamp": "2022-04-05T09:30:44.000Z",
            }
        }
    ]
}
```

`deleteHistoricalTransactions`
This method deletes transactions that are older than a specified date from the state database.

```
Ctx.ERC721Transaction.deleteTransactions(timeToExpiration: Date)
```

Parameters:

- `timeToExpiration: Date` – The date and time. Transactions older than the specified time will be deleted.

Example:

```
await this.Ctx.ERC721Transaction.deleteTransactions(timeToExpiration)

{
    "returnCode": "Success",
    "error": "",
    "result": {
        "txid":
"62ad6753cf2bfa54816b4c2f0ea325478b1cb1b84f8e13e6742c00f277310081",
        "payload": {
            "msg": "Successfuly deleted transaction older than date: Fri Apr
08 2022 00:00:00 GMT+0000 (Coordinated Universal Time).",
            "transactions": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~e687531b71d943da2fb129638784fb93a96e7698013dfc51c8c6bf4f5f797059
",

"otransaction~18446adf59b669e12990a1cf3ea0a7a15764f967fa694cf263aee0cd5a21d952
",

"otransaction~5560d4b5e0b0d0b9a6e97dcd7f81241a5daf56497a7b6819c6a55cebacc106f2
",

"otransaction~f0a0a64ec1a0c92ac732706dd75ffbd3feecd9c48fc79e42c551485edf0542cb
"
            ]
        },
        "encode": "JSON"
    }
}
```

Token Behavior Management - Mintable Behavior

`getMaxMintQuantity`
This method returns the maximum mintable quantity of a token. If the `max_mint_quantity` behavior is not configured in the specification file, then the default value is 0 and an infinite number of tokens can be minted.

```
Ctx.ERC721Token.getMaxMintQuantity(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

- On success, the maximum mintable quantity of the token, in the number data type. On error, it returns with an error message.

Example:

```
await this.Ctx.ERC721Token.getMaxMintQuantity(tokenAsset);

20000
```

`getTotalMintedTokens`
This method returns the total minted number of tokens available in the system for the specified token. The net number of tokens available is the total number of minted tokens minus the number of burned tokens. This is an asynchronous function.

```
Ctx.ERC721Token.getTotalMintedTokens()
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to operate on.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, the total minted tokens, in the number data type. On error, it returns with an error message.

Example:
```
await this.Ctx.ERC721Token.getTotalMintedTokens(tokenAsset);

4000
```

Token Behavior Management - Transferable Behavior

`safeTransferFrom`
This is an asynchronous function. This method transfers ownership of the specified NFT from the caller to another account. This method includes the following validations:

- The token exists and is not burned.
- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.
- The caller of the function is the sender.

```
Ctx.ERC721Token.safeTransferFrom(fromAccountId: string, toAccountId: string,
token: <Instance of Token Class>, data?: string)
```

Parameters:

- `fromAccountId: string` – The account ID of the sender in the current organization.
- `toAccountId: string` – The account ID of the receiver in the current organization.
- `token: <Instance of Token Class>` – The token asset to transfer.
- `data: string` – Optional additional information to store in the transaction.

Returns:

- On success, a promise with a success message that includes account details. Account IDs have the prefix `oaccount~`. On error, a rejection with an error message.

Example:
```
await this.Ctx.ERC721Token.safeTransferFrom(fromAccountId, toAccountId,
tokenAsset, data);

{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin) to Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1)"}
```

`transferFrom`
This is an asynchronous function. This method transfers ownership of the specified NFT from a sender account to a receiver account. It is the responsibility of the caller to pass the correct parameters. This method can be called by any user, not only the token owner. This method includes the following validations:

- The token exists and is not burned.

Copyright © 2025, Oracle and/or its affiliates.

- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.

```
Ctx.ERC721Token.transferFrom(fromAccountId: string, toAccountId: string,
token: <Instance of Token Class>)
```

Parameters:

- `fromAccountId: string` – The account ID of the sender in the current organization.
- `toAccountId: string` – The account ID of the receiver in the current organization.
- `token: <Instance of Token Class>` – The token asset to transfer.

Returns:

- On success, a promise with a success message that includes account details. Account IDs have the prefix `oaccount~`. On error, a rejection with an error message.

\Example:

```
await this.Ctx.ERC721Token.transferFrom(fromAccountId, toAccountId, tokeAsset,
data);
```

```
{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1) to Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin)"}
```

Token Behavior Management - Burnable Behavior

`burn`
This method deactivates, or burns, the specified NFT from the caller's account. The caller of this method must have an account. A token cannot be burned unless the token specification file includes the `burnable` behavior. If no `burner_role_name` property is specified in the `roles` section of the specification file, then the owner of the token can burn the token. If a `burner_role_name` property is specified in the `roles` section, then the user assigned the burner role who is also the minter (creator) of the token can burn the token. This is an asynchronous function.

```
Ctx.ERC721Token.burn(token: <Instance of Token Class>)
```

Parameters:

- `token: <Instance of Token Class>` – The token asset to burn.

Returns:

- On success, a promise with a success message that includes account details. On error, a rejection with an error message.

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
await this.Ctx.ERC721Token.burn(tokenAsset);

{msg": "Successfully burned NFT token: 'monalisa1' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin)"}
```

TypeScript Methods for ERC-721 NFT Locking

Blockchain App Builder automatically generates methods that you can use to lock non-fungible tokens that use the extended ERC-721 standard.

A locked token cannot be burned or transferred to other users. All other properties, such as the token's state, owner, and history are preserved. You can use the NFT locking functionality when transferring a token to another blockchain network, such as Ethereum or Polygon.

Before you can lock NFTs, you must assign the vault manager role to a user. The vault manager is a special type of role, a `TokenSys` role. `TokenSys` roles are different from asset-based roles such as burner, minter, and notary, and from administrative roles such as `Token Admin` and `Org Admin`. Currently Blockchain App Builder supports the `vault` `TokenSys` role. The single user who has the `vault` role for a chaincode is the vault manager of the chaincode, and can manage locked NFTs.

The typical flow for using the NFT locking functionality follows these steps.

- Create a non-fungible token that has the lockable behavior.
- Use the `addTokenSysRole` method to give the `vault` role to a user, the vault manager.
- Call the `lockNFT` method to lock a non-fungible token, specified by the token ID.

TokenSys Role Management Methods

`addTokenSysRole`
This method adds a `TokenSys` role to a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async addTokenSysRole(role: string, orgId: string, userId: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.addTokenSysRoleMember",
"TOKEN");
    await this.Ctx.ERC721Auth.checkAuthorization('ERC721TOKEN.addRoleMember',
'TOKEN');
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    return await this.Ctx.ERC721Token.addTokenSysRoleMember(role, accountId);
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role to give to the user.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully added role 'vault' to Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

`isInTokenSysRole`
This method returns a Boolean value to indicate if a user has a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async isInTokenSysRole(orgId: string, userId: string, role: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.isInTokenSysRole",
"TOKEN", {orgId: orgId, userId: userId });
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    return await this.Ctx.ERC721Token.isInTokenSysRole(role, accountId);
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role to check.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "result": true,
    "msg": "Account Id
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1) has vault role"
}
```


Copyright © 2025, Oracle and/or its affiliates.

removeTokenSysRole
This method removes a TokenSys role from a specified user. This method can be called only by a Token Admin of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async removeTokenSysRole(role: string, orgId: string, userId: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.removeTokenSysRoleMember",
 "TOKEN");
    const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
    return await this.Ctx.ERC721Token.removeTokenSysRoleMember(role,
accountId);
}
```

Parameters:

- `role: string` – The name of the TokenSys role to remove.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully removed role 'vault' from Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

transferTokenSysRole
This method transfers a TokenSys role from a user to another user. This method can be called only by a Token Admin of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string())
public async transferTokenSysRole(role: string, fromOrgId: string,
fromUserId: string, toOrgId: string, toUserId: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.transferTokenSysRole",
"TOKEN");
    const fromAccountId = await
this.Ctx.ERC721Account.generateAccountId(fromOrgId, fromUserId);
    const toAccountId = await
this.Ctx.ERC721Account.generateAccountId(toOrgId, toUserId);
    return await this.Ctx.ERC721Token.transferTokenSysRole(role,
fromAccountId, toAccountId);
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `role: string` – The name of the `TokenSys` role to transfer.
- `fromOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role from.
- `fromUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role from.
- `toOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role to.
- `toUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role to.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully transfered role 'vault' from Account Id:
ouaccount~f4e311528f03fffa7810753d643f66289ff6c9080fcf839902f28a1d3aff1789
(Org-Id: Org1MSP, User-Id: user1) to Account Id:
ouaccount~ae5be2ae8f98d6d32f5d02b43877d987114e7937c7bacbc30390dcce09996a19
(Org-Id: Org1MSP, User-Id: user2)"
}
```

`getAccountsByTokenSysRole`
This method returns a list of all account IDs for a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async getAccountsByTokenSysRole(role: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.getAccountsByTokenSysRole"
, "TOKEN");
    return await this.Ctx.ERC721Token.getAccountsByTokenSysRole(role);
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "accountIds": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba"
    ]
}
```

`getUsersByTokenSysRole`
This method returns user information for all users with a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async getUsersByTokenSysRole(role: string) {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.getUsersByTokenSysRole",
"TOKEN");
    return await this.Ctx.ERC721Token.getUsersByTokenSysRole(role);
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
 "users":[
      {

"accountId":"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2
caf03eba",
         "orgId":"Org1MSP",
         "userId":"user1"
      }
   ]
}
```

NFT Locking Methods

`lockNFT`
This method locks a specified non-fungible token. To lock a token, there must be a user with the `TokenSys vault` role, who acts as the vault manager. This method can be called only by the owner of the token.

```
@Validator(yup.string())
public async lockNFT(tokenId: string) {
    return await this.Ctx.ERC721Token.lockNFT(tokenId);
}
```

Parameters:

- `tokenID: string` – The ID of the token to lock.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a JSON representation of the token object.

Return Value Example:

```
{
   "assetType":"otoken",
   "tokenId":"token1",
   "tokenName":"artcollection",
   "symbol":"ART",
   "tokenStandard":"erc721+",
   "tokenType":"nonfungible",
   "tokenUnit":"whole",
   "behaviors":[
      "indivisible",
      "singleton",
      "mintable",
      "transferable",
      "lockable",
      "burnable",
      "roles"
   ],
   "roles":{
      "minter_role_name":"minter"
   },
   "mintable":{
      "max_mint_quantity":20000
   },

"createdBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
   "creationDate":"2023-10-20T10:26:29.000Z",

"owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
   "isBurned":false,
   "isLocked":true,
   "tokenUri":"token1.example.com",
   "price":120,
   "on_sale_flag":false
}
```

`isNFTLocked`
This method returns a Boolean value to indicate if a specified token is locked. This method can be called only by the token owner, the vault manager (the user with the `TokenSys vault` role), or a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async isNFTLocked(tokenId: string) {
  try {
    await this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.isNFTLocked",
"TOKEN", { tokenId });
```


Copyright © 2025, Oracle and/or its affiliates.

```
  } catch(err) {
    const isCallerTokenSysRoleHolder = await
this.Ctx.ERC721Token.isCallerTokenSysRoleHolder(TOKEN_SYS_ROLE_TYPE.VAULT);
    if(!isCallerTokenSysRoleHolder)
      throw err;
  }
  const isLocked = await this.Ctx.ERC721Token.isNFTLocked(tokenId);
  return {isLocked: isLocked}
}
```

Parameters:

- `tokenID: string` – The ID of the token.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
   "isNFTLocked":true
}
```

`getAllLockedNFTs`
This method returns a list of all locked non-fungible tokens. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async getAllLockedNFTs() {
  try {
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.getAllLockedNFTs",
"TOKEN");
  } catch(err) {
    const isCallerTokenSysRoleHolder = await
this.Ctx.ERC721Token.isCallerTokenSysRoleHolder(TOKEN_SYS_ROLE_TYPE.VAULT);
    if(!isCallerTokenSysRoleHolder)
      throw err;
  }
 return this.Ctx.ERC721Token.getAllLockedNFTs();
}
```

Parameters:

- None

Returns:

- On success, an array of the locked non-fungible token objects.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
   {
      "key":"token1",
      "valueJson":{
         "assetType":"otoken",
         "tokenId":"token1",
         "tokenName":"artcollection",
         "symbol":"ART",
         "tokenStandard":"erc721+",
         "tokenType":"nonfungible",
         "tokenUnit":"whole",
         "behaviors":[
            "indivisible",
            "singleton",
            "mintable",
            "transferable",
            "lockable",
            "burnable",
            "roles"
         ],
         "roles":{
            "minter_role_name":"minter"
         },
         "mintable":{
            "max_mint_quantity":20000
         },

"createdBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "creationDate":"2023-10-20T10:26:29.000Z",

"owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "isBurned":false,
         "isLocked":true,
         "tokenUri":"token1.example.com",
         "price":120,
         "on_sale_flag":false
      }
   }
]
```

`getAllLockedNFTsByOrg`
This method returns a list of all locked non-fungible tokens for a specified organization and optionally a specified user. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getLockedNFTsByOrg(orgId: string, userId?: string) {
  try {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    await
this.Ctx.ERC721Auth.checkAuthorization("ERC721TOKEN.getLockedNFTsByOrg",
"TOKEN");
  } catch(err) {
    const isCallerTokenSysRoleHolder = await
this.Ctx.ERC721Token.isCallerTokenSysRoleHolder(TOKEN_SYS_ROLE_TYPE.VAULT);
    if(!isCallerTokenSysRoleHolder)
      throw err;
  }
  return await this.Ctx.ERC721Token.getLockedNFTsByOrg(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user (optional).

Returns:

- On success, an array of the locked non-fungible token objects.

Return Value Example:

```
[
   {
      "key":"token1",
      "valueJson":{
         "assetType":"otoken",
         "tokenId":"token1",
         "tokenName":"artcollection",
         "symbol":"ART",
         "tokenStandard":"erc721+",
         "tokenType":"nonfungible",
         "tokenUnit":"whole",
         "behaviors":[
            "indivisible",
            "singleton",
            "mintable",
            "transferable",
            "lockable",
            "burnable",
            "roles"
         ],
         "roles":{
            "minter_role_name":"minter"
         },
         "mintable":{
            "max_mint_quantity":20000
         },

"createdBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "creationDate":"2023-10-20T10:26:29.000Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "isBurned":false,
         "isLocked":true,
         "tokenUri":"token1.examplecom",
         "price":120,
         "on_sale_flag":false
      }
   }
]
```

TypeScript Methods for ERC-721 Token Account Status

Blockchain App Builder automatically generates methods that you can use to manage account status for tokens that use the extended ERC-721 standard.

You can use the following methods to put token user accounts in the active, suspended, or deleted states.

When an account is suspended, the account user cannot complete any write operations, which include minting, burning, and transferring tokens. Additionally, other users cannot transfer tokens to a suspended account. A suspended account can still complete read operations.

An account with a non-zero token balance cannot be deleted. You must transfer or burn all tokens in an account before you can delete the account. After an account is in the deleted state, the account state cannot be changed back to active or suspended.

- Automatically Generated Account Status Methods
- Account Status SDK Methods

Automatically Generated Account Status Methods

`getAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
@Validator(yup.string(), yup.string())
     public async getAccountStatus(orgId: string, userId: string) {
       const accountId = await
this.Ctx.ERC721Account.generateAccountId(orgId, userId);
       await
this.Ctx.ERC721Auth.checkAuthorization("ERC721ACCOUNT_STATUS.get", "TOKEN",
{ accountId });
       try {
         return await
this.Ctx.ERC721AccountStatus.getAccountStatus(accountId);
       } catch (err) {
        return await
this.Ctx.ERC721AccountStatus.getDefaultAccountStatus(accountId);
       }
     }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`getAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
public async getAccountStatusHistory(orgId: string, userId: string) {
       const accountId = await
this.Ctx.ERC721Account.generateAccountId(orgId, userId);
       await this.Ctx.ERC721Account.getAccount(accountId);
       await
this.Ctx.ERC721Auth.checkAuthorization("ERC721ACCOUNT_STATUS.history",
"TOKEN", { accountId });
       const status_id = await
this.Ctx.ERC721AccountStatus.generateAccountStatusId(accountId);
       let accountStatusHistory: any;
       try {
         accountStatusHistory = await
this.Ctx.ERC721AccountStatus.history(status_id);
       } catch (err) {
         return [];
       }
       return accountStatusHistory;
     }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, the account status history in JSON format.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

`activateAccount`
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode. Deleted accounts cannot be activated. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
@Validator(yup.string(), yup.string())
 public async activateAccount(orgId: string, userId: string) {
   await
this.Ctx.ERC721Auth.checkAuthorization("ERC721ACCOUNT_STATUS.activateAccount",
 "TOKEN");
   const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
   return await this.Ctx.ERC721Account.activateAccount(accountId);
 }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

`suspendAccount`
This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
@Validator(yup.string(), yup.string())
 public async suspendAccount(orgId: string, userId: string) {
   await
this.Ctx.ERC721Auth.checkAuthorization("ERC721ACCOUNT_STATUS.suspendAccount",
"TOKEN");
   const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
   return await this.Ctx.ERC721Account.suspendAccount(accountId);
 }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

`deleteAccount`
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance must be zero.

```
@Validator(yup.string(), yup.string())
 public async deleteAccount(orgId: string, userId: string) {
   await
this.Ctx.ERC721Auth.checkAuthorization("ERC721ACCOUNT_STATUS.deleteAccount",
"TOKEN");
   const accountId = await this.Ctx.ERC721Account.generateAccountId(orgId,
userId);
   return await this.Ctx.ERC721Account.deleteAccount(accountId);
 }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Account Status SDK Methods

`getAccountStatus`
This method gets the current status of the token account.

```
Ctx.ERC721AccountStatus.getAccountStatus(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`getAccountStatusHistory`
This method gets the history of the account status.

```
Ctx.ERC721AccountStatus.history(statusId: string)
```

Parameters:

- `statusId: string` – The ID of the account status object.

Returns:

- On success, a JSON representation of the account status history.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
```


Copyright © 2025, Oracle and/or its affiliates.

```
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

activateAccount
This method activates a token account. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
Ctx.ERC721Account.activateAccount(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

suspendAccount
This method suspends a token account.

```
Ctx.ERC721AccountStatus.suspendAccount(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

deleteAccount
This method deletes a token account.

```
Ctx.ERC721Account.deleteAccount(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Scaffolded Go NFT Project for ERC-721

Blockchain App Builder takes the input from your NFT specification file and generates a fully-functional scaffolded chaincode project.

The project automatically generates NFT lifecycle classes and functions, including CRUD and non-CRUD methods. Validation of arguments, marshalling/unmarshalling, and transparent persistence capability are all supported automatically.

For information on the scaffolded project and methods that are not directly related to NFTs, see Scaffolded Go Chaincode Project.

Reference:

- Model
- Controller
  - Automatically Generated NFT Methods
  - Custom Methods
- NFT SDK Methods

Copyright © 2025, Oracle and/or its affiliates.

Model

Transparent Persistence Capability, or simplified ORM, is captured in the `OchainModel` class.

```
package model

type ArtCollection struct {
    AssetType     string `json:"AssetType" final:"otoken"`
    TokenId       string `json:"TokenId" id:"true" mandatory:"true"
validate:"regexp=^[A-Za-z0-9][A-Za-z0-9_-]*$,max=16"`
    TokenName     string `json:"TokenName" final:"artcollection"`
    TokenDesc     string `json:"TokenDesc" validate:"max=256"`
    Symbol        string `json:"Symbol" final:"ART"`
    TokenStandard string `json:"TokenStandard" final:"erc721+"`
    TokenType     string `json:"TokenType" final:"nonfungible"
validate:"regexp=^nonfungible$"`
    TokenUnit     string `json:"TokenUnit" final:"whole"
validate:"regexp=^whole$"`

    Behavior []string `json:"Behavior"
final:"[\"indivisible\",\"singleton\",\"mintable\",\"transferable\",\"burnable
\",\"roles\"]"`

    Roles map[string]interface{} `json:"Roles"
final:"{\"minter_role_name\":\"minter\"}"`

    Mintable map[string]interface{} `json:"Mintable"
final:"{\"Max_mint_quantity\":20000}"`

    Owner           string `json:"Owner,omitempty" validate:"string"`
    CreatedBy       string `json:"CreatedBy,omitempty" validate:"string"`
    TransferredBy   string `json:"TransferredBy,omitempty" validate:"string"`
    CreationDate    string `json:"CreationDate,omitempty" validate:"string"`
    TransferredDate string `json:"TransferredDate,omitempty"
validate:"string"`
    IsBurned        bool   `json:"IsBurned" validate:"bool"`
    BurnedBy        string `json:"BurnedBy,omitempty" validate:"string"`
    BurnedDate      string `json:"BurnedDate,omitempty" validate:"string"`
    TokenUri        string `json:"TokenUri" validate:"string,max=2000"`

    TokenMetadata ArtCollectionMetadata `json:"TokenMetadata"`

    Price        int  `json:"Price" validate:"int"`
    On_sale_flag bool `json:"On_sale_flag" validate:"bool"`
}

type ArtCollectionMetadata struct {
    Painting_name string `json:"Painting_name" validate:"string"`
    Description   string `json:"Description" validate:"string"`
    Image         string `json:"Image" validate:"string"`
    Painter_name  string `json:"Painter_name" validate:"string"`
}
```


Copyright © 2025, Oracle and/or its affiliates.

Controller

There is only one main controller.

```
type Controller struct {
    Ctx trxcontext.TrxContext
}
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable. The other methods are hidden.

You can use the token SDK methods to write custom methods for your business application.

Automatically Generated NFT Methods

Blockchain App Builder automatically generates methods to support NFTs and NFT life cycles. You can use these methods to initialize NFTs, manage roles and accounts, and complete other NFT lifecycle tasks without any additional coding.

Blockchain App Builder automatically generates methods to support NFTs and NFT life cycles. You can use these methods to initialize NFTs, manage roles and accounts, and complete other NFT lifecycle tasks without any additional coding. Controller methods must be public to be invokable. Public method names begin with an upper case character. Method names that begin with a lower case character are private.

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

`AddTokenAdmin`
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenAdmin(orgId string, userId string) (interface{},
error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ADMIN.AddAdmin", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
```


Copyright © 2025, Oracle and/or its affiliates.

```
      return t.Ctx.ERC721Admin.AddAdmin(orgId, userId)
}
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfully added Admin (OrgId: Org1MSP, UserId: user1)"}
```

`RemoveTokenAdmin`
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode. You cannot use this method to remove yourself as a `Token Admin`.

```
func (t *Controller) RemoveTokenAdmin(orgId string, userId string)
(interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ADMIN.RemoveAdmin", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Admin.RemoveAdmin(orgId, userId)
}
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfully removed Admin (OrgId Org1MSP UserId user1)"}
```


Copyright © 2025, Oracle and/or its affiliates.

IsTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. A `Token Admin` can call this function on any other user in the blockchain network. Other users can call this method only on their own accounts.

```
func (t *Controller) IsTokenAdmin(orgId string, userId string) (interface{},
error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ADMIN.IsTokenAdmin", "TOKEN",
map[string]string{"orgId": orgId, "userId": userId})
      if err != nil || !auth {
            return false, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Auth.IsUserTokenAdmin(orgId, userId)
}
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

Return Value Example:

```
{"result":true}
```

GetAllTokenAdmins
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) GetAllTokenAdmins() (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ADMIN.GetAllAdmins", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Admin.GetAllAdminUsers()
}
```

Parameters:

- none

Returns:

- On success, a JSON list of admins that includes `OrgId` and `UserId` objects.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "admins": [
        {
            "OrgId":"Org1MSP",
            "UserId":"admin"
        }
    ]
}
```

Methods for Token Configuration Management

`Init`
This method is called when the chaincode is instantiated. Every `Token Admin` is identified by the `UserId` and `OrgId` information in the `adminList` parameter. The `UserId` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `OrgId` is the membership service provider (MSP) ID of the user in the current network organization. The `adminList` parameter is mandatory the first time you deploy the chaincode. If you are upgrading the chaincode, pass an empty list (`[]`). Any other information in the `adminList` parameter is ignored during upgrades.

```
func (t *Controller) Init(adminList []erc721Admin.ERC721TokenAdminAsset)
(interface{}, error) {
      list, err := t.Ctx.ERC721Admin.InitAdmin(adminList)
      if err != nil {
            return nil, fmt.Errorf("initialising admin list failed %s",
err.Error())
      }
      <Token Name> := <Token Class>{}
      _, err = t.Ctx.ERC721Token.SaveClassInfo(&<Token Name>)
      if err != nil {
            return nil, err
      }
      _, err = t.Ctx.ERC721Token.SaveDeleteTransactionInfo()
      if err != nil {
            fmt.Println("error: ", err)
      }
      return list, nil
}
```

Parameters:

- `adminList array` – An array of {`OrgId, UserId`} information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

`GetAllTokens`
This method returns all of the token assets that are saved in the state database. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL

Copyright © 2025, Oracle and/or its affiliates.

rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAllTokens() (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetAllTokens", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Token.GetAllTokens()
}
```

Parameters:

- none

Returns:

- A list of all token assets in JSON format.

Return Value Example:

```
[
    {
        "key": "monalisa",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "CreationDate": "2022-04-10T11:01:42Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 0,
            "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "token Description",
            "TokenId": "monalisa",
            "TokenMetadata": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "Description": "Mona Lisa Painting",
                    "Image": "monalisa.jpeg",
                    "PainterName": "Leonardo_da_Vinci",
                    "PaintingName": "Mona_Lisa"
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            }
        },
        {
            "key": "monalisa2",
            "valueJson": {
                "AssetType": "otoken",
                "Behavior": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "CreationDate": "2022-04-10T11:04:44Z",
                "IsBurned": false,
                "Mintable": {
                    "Max_mint_quantity": 20000
                },
                "NftBasePrice": 100,
                "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "Roles": {
                    "burner_role_name": "burner",
                    "minter_role_name": "minter"
                },
                "Symbol": "PNT",
                "TokenDesc": "",
                "TokenId": "monalisa1",
                "TokenMetadata": {
                     "Description": "Mona Lisa Painting",
                     "Image": "monalisa.jpeg",
                     "PainterName": "Leonardo_da_Vinci",
                     "PaintingName": "Mona_Lisa"
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "https://
```


Copyright © 2025, Oracle and/or its affiliates.

```
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
        }
        }
    }
]
```

`GetAllTokensByUser`
This method returns all of the token assets that are owned by a specified user. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) GetAllTokensByUser(orgId string, userId string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetAllTokensByUser",
"TOKEN", map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
    return t.Ctx.ERC721Token.GetAllTokensByUser(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- A list of token assets in JSON format.

Return Value Example:

```
[
    {
        "key": "monalisa",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
```


Copyright © 2025, Oracle and/or its affiliates.

```
            ],
            "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "CreationDate": "2022-04-10T11:01:42Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 0,
            "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "token Description",
            "TokenId": "monalisa",
            "TokenMetadata": {
                 "Description": "Mona Lisa Painting",
                 "Image": "monalisa.jpeg",
                 "PainterName": "Leonardo_da_Vinci",
                 "PaintingName": "Mona_Lisa"
            },
            "TokenName": "paintingnft",
            "TokenStandard": "erc721+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
        }
    },
    {
        "key": "monalisa2",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "CreationDate": "2022-04-10T11:04:44Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 100,
            "Owner":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "",
            "TokenId": "monalisa1",
            "TokenMetadata": {
                 "Description": "Mona Lisa Painting",
                 "Image": "monalisa.jpeg",
                 "PainterName": "Leonardo_da_Vinci",
                 "PaintingName": "Mona_Lisa"
            },
            "TokenName": "paintingnft",
            "TokenStandard": "erc721+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
        }
        }
    }
]
```

GetTokenById

This method returns a token object if the token is present in the state database. This method can be called only by a `Token Admin` of the chaincode or the token owner.

```
func (t *Controller) GetTokenById(tokenId string) (interface{}, error) {
      auth, err := t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.Get",
"TOKEN", map[string]string{"tokenId": tokenId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      tokenAsset, err := t.getTokenObject(tokenId)
      if err != nil {
            return nil, err
      }
      return tokenAsset.Interface(), nil
}
```

Parameters:

- `tokenId: string` – The ID of the token to get.

Returns:

- The token asset in JSON format.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2022-04-06T00:35:42Z",
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "NftBasePrice": 200,
    "Owner":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "Symbol": "PNT",
    "TokenDesc": "Token Description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "paintingnft",
    "TokenStandard": "erc721+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
    "TransferredBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "TransferredDate": "2022-04-06T00:51:56Z"
}
```

GetTokenHistory

This method returns the history for a specified token ID. This method can only be called when connected to the remote Oracle Blockchain Platform network. Anyone can call this method.

```
func (t *Controller) GetTokenHistory(tokenId string) (interface{}, error) {
      /*
      auth, err :=
```


Copyright © 2025, Oracle and/or its affiliates.

```
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetTokenHistory", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      */
      return t.Ctx.ERC721Token.History(tokenId)
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value Example:

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T19:22:52z",
        "TxId":
"6b7989be322956164a8d1cd7bf2a7187d59eba73ce756e6bf946ab48b349bbc0",
        "Value": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "CreationDate": "2022-04-06T19:22:23z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 100,
            "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "token Description",
            "TokenId": "monalisa",
            "TokenMetadata": {
               "Description": "Mona Lisa Painting",
               "Image": "monalisa.jpeg",
               "PainterName": "Leonardo_da_Vinci",
               "PaintingName": "Mona_Lisa"
            },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "TokenName": "paintingnft",
             "TokenStandard": "erc721+",
             "TokenType": "nonfungible",
             "TokenUnit": "whole",
             "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg""
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T19:22:23z",
        "TxId":
"e61bcb3cb61c8920f7e6d8f0d19726c7c88d876e0ad6cfb052cfb92d49985c3f",
        "Value": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "CreationDate": "2022-04-06T19:22:23z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 100,
            "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "",
            ""TokenId": "monalisa",
            "TokenMetadata": {
                "Description": "Mona Lisa Painting",
                "Image": "monalisa.jpeg",
                "PainterName": "Leonardo_da_Vinci",
                "PaintingName": "Mona_Lisa"
            },
            "TokenName": "paintingnft",
            "TokenStandard": "erc721+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    }
]
```

`getTokenObject`
This is a utility method that returns an instance of the token for a specified token ID. This method is used by many of the automatically generated methods to fetch token objects. You can call this method as needed from your custom methods. When you create a tokenized asset or class, update the switch case with the corresponding `Token` class to return the correct token object. The `ochain sync` command in Blockchain App Builder automatically creates a switch case when a tokenized asset is created in the specification file. Because this method is private, it is not directly invokable and can only be called from other methods.

```
func (t *Controller) getTokenObject(tokenId string) (reflect.Value, error) {
      if tokenId == "" {
            return reflect.Value{}, fmt.Errorf("error in retrieving token,
token_id cannot be empty")
      }
      tokenAsset, err := t.Ctx.ERC721Token.Get(tokenId)
      if err != nil {
            return reflect.Value{}, fmt.Errorf("no token exists with id %s
%s", tokenId, err.Error())
      }
      tokenName := tokenAsset.(map[string]interface{})["TokenName"].(string)
      switch tokenName {
      case "<Token Name>":
            var asset <Token Class>
            _, err := t.Ctx.ERC721Token.Get(tokenId, &asset)
            if err != nil {
                  return reflect.Value{}, err
            }
            return reflect.ValueOf(&asset), nil
      default:
            return reflect.Value{}, fmt.Errorf("no token exists with token
name %s", tokenName)
      }
}
```

Parameters:

- `tokenId: string` – The ID of the token.

`OwnerOf`
This method returns the account ID of the owner of the specified token ID. Anyone can call this method.

```
func (t *Controller) Ownerof(tokenId string) (interface{}, error) {
      return t.Ctx.ERC721Token.OwnerOf(tokenId)
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- A JSON object of the owner's account ID.

Return Value Example:

```
{"Owner":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba"}
```

Name
This method returns the name of the token class. Anyone can call this method.

```
func (t *Controller) Name() (interface{}, error) {
      return t.Ctx.ERC721Token.Name()
}
```

Parameters:

- none

Returns:

- A JSON object of the token name.

Return Value Example:

```
{"TokenName": "paintingnft"}
```

Symbol
This method returns the symbol of the token class. Anyone can call this method.

```
func (t *Controller) Symbol() (interface{}, error) {
      return t.Ctx.ERC721Token.Symbol()
}
```

Parameters:

- none

Returns:

- A JSON object of the token symbol.

Return Value Example:

```
{"Symbol": "PNT"}
```

TokenURI
This method returns the URI of a specified token. Anyone can call this method.

```
func (t *Controller) TokenURI(tokenId string) (interface{}, error) {
      return t.Ctx.ERC721Token.TokenURI(tokenId)
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object of the token URI.

Return Value Example:

```
{"TokenURI": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"}
```

`TotalSupply`
This method returns the total number of minted tokens. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) TotalSupply() (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.TotalSupply", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Token.TotalSupply()
}
```

Parameters:

- none

Returns:

- On success, a JSON object of the token count.

Return Value Example:

```
{"TotalSupply": 3}
```

`TotalNetSupply`
This method returns the total number of minted tokens minus the number of burned tokens. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) TotalNetSupply() (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.TotalNetSupply", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Token.GetTotalMintedTokens()
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- none

Returns:

- On success, a JSON object of the token count.

Return Value Example:

```
{"TotalNetSupply": 2}
```

Methods for Account Management

`CreateAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track the number of NFTs a user has. Users must have accounts in the network to complete token-related operations. You can create only one NFT account per user.
An account ID is an alphanumeric set of characters, prefixed with `oaccount~` and followed by an SHA-256 hash of the membership service provider ID (`OrgId`) of the user in the current network organization, the user name or email ID (`UserId`) of the instance owner or the user who is logged in to the instance, and the constant string `nft`. This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) CreateAccount(orgId string, userId string, tokenType
string) (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.CreateAccount", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Account.CreateAccount(orgId, userId, tokenType)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenType: string` – The only supported token type is `nonfungible`.

Returns:

- On success, a JSON object of the account that was created.

Return Value Example:

```
{
    "AssetType": "oaccount",
    "BapAccountVersion" : 0,
    "AccountId":
"oaccount~a0a60d54ba9e2ff349737d292ea10ebd9cc8f1991c11443c19d20aea299a9507",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "UserId": "admin",
    "OrgId": "Org1MSP",
    "TokenType": "nonfungible",
    "NoOfNfts": 0
}
```

BalanceOf

This method returns the total number of NFTs that a specified user holds. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) BalanceOf(orgId string, userId string) (interface{},
error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.BalanceOf", "TOKEN",
map[string]string{"orgId": orgId, "userId": userId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil{
            return nil,err
      }
      return t.Ctx.ERC721Account.BalanceOf(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- A JSON object of the current NFT count.

Return Value Example:

```
{"totalNfts": 0}
```

GetAllAccounts

This method returns a list of all accounts. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAllAccounts() (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.GetAllAccounts", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
```


Copyright © 2025, Oracle and/or its affiliates.

```
      return t.Ctx.ERC721Account.GetAllAccounts()
}
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

Return Value Example:

```
[
    {
        "key":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
        "valueJson": {
            "AccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "AssetType": "oaccount",
            "BapAccountVersion" : 0,
            "NoOfNfts": 5,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user1"
        }
    },
    {
        "key":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
        "valueJson": {
            "AccountId":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
            "AssetType": "oaccount",
            "BapAccountVersion" : 0,
            "NoOfNfts": 0,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user_minter"
        }
    },
    {
        "key":
"oaccount~5541fb520058d83664b844e7a55fe98d574ddeda765d0e795d4779e9ccc271ce",
        "valueJson": {
            "AccountId":
"oaccount~5541fb520058d83664b844e7a55fe98d574ddeda765d0e795d4779e9ccc271ce",
            "AssetType": "oaccount",
            "BapAccountVersion" : 0,
            "NoOfNfts": 0,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user_burner"
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
]
```

GetAccountByUser
This method returns account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
func (t *Controller) GetAccountByUser(orgId string, userId string)
(interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.GetAccountByUser",
"TOKEN", map[string]string{"orgId": orgId, "userId": userId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Account.GetAccountWithStatusByUser(orgId, userId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `AccountId` – The ID of the user account.
- `UserId` – The user name or email ID of the user.
- `OrgId` – The membership service provider (MSP) ID of the user in the current organization.
- `TokenType` – The type of token that the account holds.
- `NoOfNfts` – The total number of NFTs held by the account.
- `BapAccountVersion` – An account object parameter for internal use.
- `Status` – The current status of the user account.

Return Value Example:

```
{
  "AccountId":
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
  "AssetType": "oaccount",
  "BapAccountVersion": 0,
  "NoOfNfts": 0,
  "OrgId": "appdev",
  "Status": "active",
  "TokenType": "nonfungible",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "UserId": "idcqa"
}
```

GetUserByAccountId
This method returns the user details of a specified account. This method can be called by any user.

```
func (t *Controller) GetUserByAccountId(accountId string) (interface{},
error) {
      return t.Ctx.ERC721Account.GetUserByAccountById(accountId)
}
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`OrgId` and `UserId`).

Return Value Example:

```
{
    "OrgId": "Org1MSP",
    "UserId": "admin"
}
```

GetAccountHistory
This method returns account history for a specified user. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) GetAccountHistory(orgId string, userId string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.History", "TOKEN",
map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Account.History(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a list of objects.

Return Value Example:

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T08:16:53Z",
        "TxId":
"750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7",
        "Value": {
            "AccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "AssetType": "oaccount",
            "BapAccountVersion" : 1,
            "NoOfNfts": 1,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user1"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T08:15:19Z",
        "TxId":
"49eb84c42d452e5ba0028d8ebb4190454cf9013a11c0bad3e96594af452d4982",
        "Value": {
            "AccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "AssetType": "oaccount",
            "BapAccountVersion" : 0,
            "NoOfNfts": 0,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user1"
        }
    }
]
```

Methods for Role Management

`AddRole`
This method adds a role to a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddRole(role string, orgId string, userId string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
```


Copyright © 2025, Oracle and/or its affiliates.

```
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.AddRoleMember", "TOKEN",
map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Token.AddRoleMember(role, accountId)
}
```

Parameters:

- `userRole: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message with user details.

Return Value Example:

```
{"msg": "Successfully added role minter to
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(orgId : Org1MSP, userId : admin)"}
```

`RemoveRole`
This method removes a role from a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) RemoveRole(userRole string, orgId string, userId string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.RemoveRoleMember", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Token.RemoveRoleMember(userRole, accountId)
}
```

Parameters:

- `userRole: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message with user details.

Return Value Example:

```
{"msg": "successfully removed memberId
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(orgId : Org1MSP, userId : admin) from role minter"}
```

`GetAccountsByRole`
This method returns a list of all account IDs for a specified role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetAccountsByRole(userRole string) (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ROLE.GetAccountsByRole", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Role.GetAccountsByRole(userRole)
}
```

Parameters:

- `userRole: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
    "accounts": [

"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d"
    ]
}
```

`GetUsersByRole`
This method returns a list of all users for a specified role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetUsersByRole(userRole string) (interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ROLE.GetUsersByRole", "TOKEN")
      if err != nil && !auth {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Role.GetUsersByRole(userRole)
}
```

Parameters:

- `userRole: string` – The name of the role to search for.

Returns:

- On success, a JSON array of the user objects (`orgId` and `userId`).

Return Value Example:

```
{
    "Users": [
        {
            "OrgId": "Org1MSP",
            "UserId": "admin"
        }
    ]
}
```

IsInRole
This method returns a Boolean value to indicate if a user has a specified role. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
func (t *Controller) IsInRole(orgId string, userId string, role string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.IsInRole", "TOKEN",
map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      result, err := t.Ctx.ERC721Token.IsInRole(role, accountId)
      if err != nil {
            return nil, fmt.Errorf("error in IsInRole %s", err.Error())
      }
      response := make(map[string]interface{})
      response["result"] = result
      return response, nil
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `userRole: string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Return Value Example:

```
{"result":"true"}
```

Methods for Transaction History Management

`GetAccountTransactionHistory`
This method returns account transaction history for a specified user. This method can be called by the `Token Admin` of the chaincode or the owner of the account.

```
func (t *Controller) GetAccountTransactionHistory(orgId string, userId
string) (interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.GetAccountTransactionHistor
y", "TOKEN", map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }

      transactionArray, err :=
t.Ctx.ERC721Account.GetAccountTransactionHistory(accountId)
      return transactionArray, err
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a list of objects.

Return Value Example:

```
[
     {
        "Timestamp": "2022-04-06T08:31:39Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
        "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
",
        "TransactionType": "DEBIT"
    }
    {
        "Timestamp": "2022-04-06T08:16:53Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
        "TransactionId":
"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
",
        "TransactionType": "MINT"
    }
]
```

`GetAccountTransactionHistoryWithFilters`
This method returns account transaction history for a specified user, filtered by `PageSize`, `Bookmark`, `StartTime` and `EndTime`. This method can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) GetAccountTransactionHistoryWithFilters(orgId string,
userId string, filters ...erc721Account.AccountHistoryFilters) (interface{},
error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, err
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.GetAccountTransactionHistor
yWithFilters", "TOKEN", map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }

      transactionArray, err :=
t.Ctx.ERC721Account.GetAccountTransactionHistoryWithFilters(accountId,
filters...)
      return transactionArray, err
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Returns:

- On success, a list of objects.

Return Value Example:

```
[
     {
        "Timestamp": "2022-04-06T08:31:39Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
        "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
",
        "TransactionType": "DEBIT"
    }
    {
        "Timestamp": "2022-04-06T08:16:53Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
        "TransactionId":
"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
",
        "TransactionType": "MINT"
    }
]
```

`GetTransactionById`
This method returns transaction history for a specified transaction ID. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) GetTransactionById(transactionId string) (interface{},
error) {
      return t.Ctx.ERC721Transaction.GetTransactionById(transactionId)
}
```

Parameters:

- `transactionId: string` – The id of the transaction asset.

Returns:

- On success, a list of objects.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "History": [
        {
            "IsDelete": "false",
            "Timestamp": "2022-04-06T08:31:39Z",
            "TxId":
"5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7",
            "Value": {
                "AssetType": "otransaction",
                "Data": "",
                "FromAccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "Timestamp": "2022-04-06T08:31:39Z",
                "ToAccountId":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
                "TokenId": "monalisa",
                "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
",
                "TransactionType": "TRANSFER",
                "TriggeredByAccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1"
            }
        }
    ],
    "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
"
}
```

`DeleteHistoricalTransactions`
This method deletes transactions older than a specified time stamp in the state database. This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) DeleteHistoricalTransactions(timestamp string)
(interface{}, error) {
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TRANSACTION.DeleteHistoricalTransac
tions", "TOKEN")
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      return t.Ctx.ERC721Transaction.DeleteHistoricalTransactions(timestamp)
}
```

Parameters:

- `timestamp: Date` – A time stamp. All transactions before the time stamp will be deleted.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a list of objects.

Return Value Example:

```
{
    "Transactions": [

"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
"
    ],
    "msg": "Successfuly deleted transaction older than
date:2022-04-06T08:17:53Z"
}
```

Methods for Token Behavior Management - Mintable Behavior

`Create<Token Name>Token`
This method creates (mints) an NFT. The asset and associated properties are saved in the state database. The caller of this transaction must have a token account. The caller of this transaction becomes the owner of the NFT. If the token specification file includes the `roles` section for `behaviors` and the `minter_role_name` property for `roles`, then the caller of the transaction must have the minter role. Otherwise, any caller can mint NFTs.

```
func (t *Controller) Create<Token Name>Token(tokenAsset <Token Class>)
(interface{}, error) {
      return t.Ctx.ERC721Token.CreateToken(&tokenAsset)
}
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset to mint. For more information about the properties of the token asset, see the input specification file.

Returns:

- On success, a JSON token asset object that includes the following properties:
- `Behavior` – A description of all token behaviors.
- `CreatedBy` – The account ID of the user who called the transaction to mint the token.
- `CreationDate` – The time stamp of the transaction.
- `IsBurned` – A Boolean value that indicates if the NFT identified by `tokenId` is burned.
- `Mintable` – A description of the properties of mintable behavior. The `max_mint_quantity` property specifies the maximum number of NFTs of this token class that can be created.
- `Owner` – The account ID of the current owner of the token. During the minting process, the caller of this method becomes the owner of the token.
- `Symbol` – The symbol of the token.
- `TokenDesc` – The description of the token.
- `TokenMetadata` – JSON information that describes the token.
- `TokenName` – The name of the token.

Copyright © 2025, Oracle and/or its affiliates.

- `TokenStandard` – The standard of the token.
- `TokenType` – The type of token held by this account.
- `TokenUnit` – The unit of the token.
- `TokenUri` – The URI of the token.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",

    "CreationDate": "2022-04-06T08:16:53Z",
    "IsBurned": false,
     "Mintable": {
        "Max_mint_quantity": 20000
    },
    "NftBasePrice": 100,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "Symbol": "PNT",
    "TokenDesc": "token Description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "paintingnft",
    "TokenStandard": "erc721+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
}
```

`Update<Token Name>Token`
This method updates token properties. This method can be called only by the user who is the owner or creator of the token. After a token asset is created, only the token owner can update

Copyright © 2025, Oracle and/or its affiliates.

the token custom properties. If the user is both token owner and creator of a token, they can also update the `TokenDesc` property. Token metadata cannot be updated. You must pass all token properties to this method, even if you want to update only certain properties.

```
func (t *Controller) Update<Token Name>Token(tokenAsset <Token Class>)
(interface{}, error) {
      return t.Ctx.ERC721Token.UpdateToken(&tokenAsset)
}
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset to update. For more information about the the properties of the token asset, see the input specification file.

Returns:

- On success, an updated JSON token asset object

Return Value Example:

```
{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",

    "CreationDate": "2022-04-06T08:16:53Z",
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "NftBasePrice": 200,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "Symbol": "PNT",
    "TokenDesc": "Token Description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "TokenName": "paintingnft",
    "TokenStandard": "erc721+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
}
```

Methods for Token Behavior Management - Transferable Behavior

`SafeTransferFrom`
This method transfers ownership of the specified NFT from the caller to another account. This method includes the following validations:

- The token exists and is not burned.
- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.
- The caller of the function is the sender.

```
func (t *Controller) SafeTransferFrom(fromOrgId string, fromUserId string,
toOrgId string, toUserId string, tokenId string, data ...string)
(interface{}, error) {
      tokenAssetValue, err := t.getTokenObject(tokenId)
      if err != nil {
            return nil, err
      }
      fromAccountId, err := t.Ctx.ERC721Account.GenerateAccountId(fromOrgId,
fromUserId)
      if err != nil {
            return nil, err
      }
      toAccountId, err := t.Ctx.ERC721Account.GenerateAccountId(toOrgId,
toUserId)
      if err != nil {
            return nil, err
      }
      return t.Ctx.ERC721Token.SafeTransferFrom(fromAccountId, toAccountId,
tokenAssetValue.Interface(), data...)
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the sender in the current organization.
- `fromUserId: string` – The user name or email ID of the sender.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.
- `tokenId: string` – The ID of the token to transfer.

Copyright © 2025, Oracle and/or its affiliates.

- `data: string` – Optional additional information to store in the transaction record.

Returns:

- On success, a message with the sender and receiver account details.

Return Value Example:

```
{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin) to Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1)"}
```

`TransferFrom`
This method transfers ownership of the specified NFT from a sender account to a receiver account. It is the responsibility of the caller to pass the correct parameters. This method can be called by any user, not only the token owner. This method includes the following validations:

- The token exists and is not burned.
- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.

```
func (t *Controller) TransferFrom(fromOrgId string, fromUserId string,
toOrgId string, toUserId string, tokenId string) (interface{}, error) {
      tokenAssetValue, err := t.getTokenObject(tokenId)
      if err != nil {
            return nil, err
      }
      fromAccountId, err := t.Ctx.ERC721Account.GenerateAccountId(fromOrgId,
fromUserId)
      if err != nil {
            return nil, err
      }
      toAccountId, err := t.Ctx.ERC721Account.GenerateAccountId(toOrgId,
toUserId)
      if err != nil {
            return nil, err
      }
      return t.Ctx.ERC721Token.TransferFrom(fromAccountId, toAccountId,
tokenAssetValue.Interface())
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the sender in the current organization.
- `fromUserId: string` – The user name or email ID of the sender.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenId: string` – The ID of the token to transfer.

Returns:

- On success, a message with the sender and receiver account details.

Return Value Example:

```
{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin) to Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1)"}
```

Methods for Token Behavior Management - Burnable Behavior

`Burn`
This method deactivates, or burns, the specified NFT from the caller's account. The caller of this method must have an account. A token cannot be burned unless the token specification file includes the `burnable` behavior. If no `burner_role_name` property is specified in the `roles` section of the specification file, then the owner of the token can burn the token. If a `burner_role_name` property is specified in the `roles` section, then the user assigned the burner role who is also the minter (creator) of the token can burn the token.

```
func (t *Controller) Burn(tokenId string) (interface{}, error) {
      tokenAssetValue, err := t.getTokenObject(tokenId)
      if err != nil {
            return nil, err
      }
      return t.Ctx.ERC721Token.Burn(tokenAssetValue.Interface())
}
```

Parameters:

- `tokenId: string` – The ID of the token to burn.

Returns:

- On success, a message with the account details.

Return Value Example:

```
{"msg": "Successfully burned NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin"}
```

`BurnNFT`
This method deactivates, or burns, the specified NFT from the caller's account, and returns a token object and token history. The caller of this method must have an account. A token cannot be burned unless the token specification file includes the `burnable` behavior. If no `burner_role_name` property is specified in the `roles` section of the specification file, then the owner of the token can burn the token. If a `burner_role_name` property is specified in the

Copyright © 2025, Oracle and/or its affiliates.

`roles` section, then the user assigned the burner role who is also the minter (creator) or owner of the token can burn the token.

```
func (t *Controller) BurnNFT(tokenId string) (interface{}, error) {
      tokenAsset, err := t.Ctx.ERC721Token.Get(tokenId)
      if err != nil {
            return nil, err
      }
      tokenHistory, err := t.Ctx.ERC721Token.History(tokenId)
      if err != nil {
            return nil, err
      }
      token := tokenAsset.(map[string]interface{})
      if token[constants.IsBurned] == true {
            return nil, fmt.Errorf("token with tokenId %s is already burned",
tokenId)
      }
      token[constants.TokenId], err = strconv.Atoi(token[constants.TokenId].
(string))
      if err != nil {
            return nil, fmt.Errorf("tokenId is expected to be integer but
found %s", tokenId)
      }
      tokenHistoryBytes, err := json.Marshal(tokenHistory)
      if err != nil {
            return nil, err
      }
      var tokenHistoryAsRawJson json.RawMessage
      err = json.Unmarshal(tokenHistoryBytes, &tokenHistoryAsRawJson)
      if err != nil {
            return nil, err
      }
      token[constants.TokenHistory] = string(tokenHistoryAsRawJson)
      token[constants.IsBurned] = true
      _, err = t.Burn(tokenId)
      if err != nil {
            return nil, err
      }
      return token, nil
}
```

Parameters:

- `tokenId: string` – The ID of the token to burn.

Returns:

- On success, a token object that includes token history information.

Return Value Example:

```
{
    "AssetType": "otoken",
    "Behavior": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "indivisible",
            "singleton",
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "CreationDate": "2023-08-22T13:19:33+05:30",
        "IsBurned": true,
        "Mintable": {
            "Max_mint_quantity": 20000
        },
        "On_sale_flag": false,
        "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "Price": 120,
        "Roles": {
            "minter_role_name": "minter"
        },
        "Symbol": "ART",
        "TokenDesc": "",
        "TokenHistory":
"[{\"IsDelete\":\"false\",\"Timestamp\":\"2023-08-22T13:19:33+05:30\",\"TxId\"
:\"0219099bcaaecd5f76f7f08d719384fd5ed34103a55bd8d3754eca0bfc691594\",\"Value\
":{\"AssetType\":\"otoken\",\"Behavior\":
[\"indivisible\",\"singleton\",\"mintable\",\"transferable\",\"burnable\",\"ro
les\"],\"CreatedBy\":\"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd
1aae201d347ad1288d\",\"CreationDate\":\"2023-08-22T13:19:33+05:30\",\"IsBurned
\":false,\"Mintable\":
{\"Max_mint_quantity\":20000},\"On_sale_flag\":false,\"Owner\":\"oaccount~42e8
9f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d\",\"Price\":120,\
"Roles\":
{\"minter_role_name\":\"minter\"},\"Symbol\":\"ART\",\"TokenDesc\":\"\",\"Toke
nId\":\"1\",\"TokenMetadata\":
{\"Description\":\"\",\"Image\":\"\",\"Painter_name\":\"\",\"Painting_name\":\
"\"},\"TokenName\":\"artcollection\",\"TokenStandard\":\"erc721+
\",\"TokenType\":\"nonfungible\",\"TokenUnit\":\"whole\",\"TokenUri\":\"exampl
e.com\"}}]",
        "TokenId": 1,
        "TokenMetadata": {
            "Description": "",
            "Image": "",
            "Painter_name": "",
            "Painting_name": ""
        },
        "TokenName": "artcollection",
        "TokenStandard": "erc721+",
        "TokenType": "nonfungible",
        "TokenUnit": "whole",
        "TokenUri": "example.com"
    }
```


Copyright © 2025, Oracle and/or its affiliates.

Custom Methods

You can use the token SDK methods to write custom methods for your business application.

The following example shows how to use token SDK methods in custom methods. When the `Sell` method is called, it posts a token for sale for a specified price.

```
func (t *Controller) Sell(tokenId string, sellingPrice int) (interface{},
error) {
    var tokenAsset ArtCollection
    _, err := t.Ctx.ERC721Token.Get(tokenId, &tokenAsset)
    if err != nil {
       return nil, err
    }

 /**  * price is a custom asset
      attribute to set the price of a non-fungible token in the
      marketplace  */
    tokenAsset.Price = sellingPrice
 /**  * on_sale_flag is a
      custom asset attribute that maintains non-fungible token selling status
in the
      marketplace  */
    tokenAsset.On_sale_flag = true

    _, err = t.Ctx.ERC721Token.UpdateToken(tokenAsset)
    if err != nil {
        return nil, err
    }

    msg := fmt.Sprintf("Token ID : %s has been posted for selling in the
marketplace", tokenId)
    return msg, nil
}
```

NFT SDK Methods

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

The NFT SDK provides an access control function. Some methods can be called only by a `Token Admin` or `Account Owner` of the token. You can use this feature to ensure that operations

Copyright © 2025, Oracle and/or its affiliates.

are carried out only by the intended users. Any unauthorized access results in an error. To use the access control function, import the `Authorization` class from the `../lib/auth` module.

```
import { ERC721Authorization } from '../lib/erc721-auth';
```

`AddAdmin`
This method adds a user as a `Token Admin` of the token chaincode.

```
Ctx.ERC721Admin.AddAdmin(orgId string, userId string) (interface{}, error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user added as a `Token Admin` of the token chaincode. On error, a non-nil error object that contains an error message.

Example:
```
t.Ctx.ERC721Admin.AddAdmin(orgId, userId)

{"msg":"Successfully added Admin (OrgId: Org1MSP, UserId: user1)"}
```

`RemoveAdmin`
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.ERC721Admin.RemoveAdmin(orgId string, userId string) (interface{}, error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user removed as a `Token Admin` of the token chaincode. On error, a non-nil error object that contains an error message.

Example:
```
t.Ctx.ERC721Admin.RemoveAdmin(orgId, userId)

{"msg":"Successfuly removed Admin (OrgId Org1MSP UserId user1)"}
```


Copyright © 2025, Oracle and/or its affiliates.

IsUserTokenAdmin
This method returns a map with the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`.

```
Ctx.ERC721Auth.IsUserTokenAdmin(orgId string, userId string)  (interface{},
error)
```

Parameters:

- `orgId` – The membership service provider (MSP) ID of the user in the current network organization.
- `userId` – The user name or email ID of the user.

Returns:

- A Boolean response.

Example:
```
t.Ctx.ERC721Auth.IsUserTokenAdmin(orgId, userId)

{"result":true}
```

GetAllAdmins
This method returns a list of all `Token Admin` users.

```
Ctx.ERC721Admin.GetAllAdmins() (Admin[], error)
```

Parameters:

- none

Returns:

- On success, a list of all `Token Admin` users. On error, a non-nil error object that contains an error message.

Example:
```
t.Ctx.ERC721Admin.GetAllAdmins()

{
    "admins": [
        {
            "OrgId":"Org1MSP",
            "UserId":"admin"
        }
    ]
}
```

GetAllAdminUsers
This method returns a list of all `Token Admin` users.

```
Ctx.ERC721Admin.GetAllAdminUsers() (interface{}, error)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- none

Returns:

- On success, a list of all `Token Admin` users in `map[string]interface{}` format. On error, a non-nil error object that contains an error message.

Example:
`t.Ctx.ERC721Admin.GetAllAdminUsers()`

```
{
    "admins": [
        {
            "OrgId":"Org1MSP",
            "UserId":"admin"
        }
    ]
}
```

`CheckAuthorization`
Use this method to add an access control check to an operation. Most automatically generated methods include access control. Certain token methods can be run only by an `ERC721Admin` or the `Account Owner` of the token or by the `MultipleAccountOwner` for users with multiple accounts. The `CheckAuthorization` method is part of the `erc721Auth` package, which you access via the `Ctx` struct (receiver). The access control mapping is described in the `oChainUtil.go` file, as shown in the following text. You can modify access control by editing the `oChainUtil.go` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
  var t TokenAccess
      var r RoleAccess
      var a AccountAccess
      var as AccountStatusAccess
      var h HoldAccess
      var ad AdminAccess
      var trx TransactionAccess
      var tc TokenConversionAccess
      var auth AuthAccess
      var erc721ad ERC721AdminAccess
      var erc721t ERC721TokenAccess
      var erc721r ERC721RoleAccess
      var erc721a ERC721AccountAccess
      var erc721as ERC721AccountStatusAccess
      var erc721trx ERC721TransactionAccess
     auth.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner"}

      trx.DeleteHistoricalTransactions = []string{"Admin"}
      ad.AddAdmin = []string{"Admin"}
      ad.RemoveAdmin = []string{"Admin"}
      ad.GetAllAdmins = []string{"Admin", "OrgAdmin"}
      ad.AddOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
```


Copyright © 2025, Oracle and/or its affiliates.

```
        ad.RemoveOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
        ad.GetOrgAdmins = []string{"Admin", "OrgAdmin"}
        ad.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner", "OrgAdmin"}
        t.Save = []string{"Admin"}
        t.GetAllTokens = []string{"Admin", "OrgAdmin"}
        t.Update = []string{"Admin"}
        t.GetTokenDecimals = []string{"Admin", "OrgAdmin"}
        t.GetTokensByName = []string{"Admin", "OrgAdmin"}
        t.AddRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
        t.RemoveRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
        t.IsInRole = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
        t.GetTotalMintedTokens = []string{"Admin", "OrgAdmin"}
        t.GetNetTokens = []string{"Admin", "OrgAdmin"}
        t.Get = []string{"Admin", "OrgAdmin"}
        t.GetTokenHistory = []string{"Admin", "OrgAdmin"}
        a.CreateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        a.AssociateToken = []string{"Admin", "OrgAdminAccountIdCheck"}
        a.GetAllAccounts = []string{"Admin"}
        a.GetAllOrgAccounts = []string{"Admin", "OrgAdminOrgIdCheck"}
        a.GetAccount = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.History = []string{"Admin", "AccountOwner"}
        a.GetAccountTransactionHistory = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
        a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
        a.GetSubTransactionsById = []string{"Admin", "TransactionInvoker"}
        a.GetSubTransactionsByIdWithFilters = []string{"Admin",
"TransactionInvoker"}
        a.GetAccountBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.GetAccountOnHoldBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.GetOnHoldIds = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.GetAccountsByUser = []string{"Admin", "OrgAdminOrgIdCheck",
"MultipleAccountOwner"}

        as.Get = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
        as.ActivateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        as.SuspendAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        as.DeleteAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        r.GetAccountsByRole = []string{"Admin"}
        r.GetUsersByRole = []string{"Admin"}
        r.GetOrgAccountsByRole = []string{"Admin", "OrgAdminOrgIdCheck"}
        r.GetOrgUsersByRole = []string{"Admin", "OrgAdminOrgIdCheck"}

        tc.InitializeExchangePoolUser = []string{"Admin"}
        tc.AddConversionRate = []string{"Admin"}
        tc.UpdateConversionRate = []string{"Admin"}
        tc.GetConversionRate = []string{"Admin", "OrgAdmin", "AnyAccountOwner"}
        tc.GetConversionRateHistory = []string{"Admin", "OrgAdmin",
"AnyAccountOwner"}
        tc.TokenConversion = []string{"Admin", "AnyAccountOwner"}
```


Copyright © 2025, Oracle and/or its affiliates.

```
        tc.GetExchangePoolUser = []string{"Admin"}
        erc721ad.AddAdmin = []string{"Admin"}
        erc721ad.GetAllAdmins = []string{"Admin"}
        erc721ad.IsTokenAdmin = []string{"Admin"}
        erc721ad.RemoveAdmin = []string{"Admin"}
        erc721trx.DeleteHistoricalTransactions = []string{"Admin"}
        erc721t.Save = []string{"Admin"}
        erc721t.GetAllTokens = []string{"Admin"}
        erc721t.Update = []string{"Admin"}
        erc721t.GetTokensByName = []string{"Admin"}
        erc721t.AddRoleMember = []string{"Admin"}
        erc721t.RemoveRoleMember = []string{"Admin"}
        erc721t.IsInRole = []string{"Admin", "AccountOwner"}
        erc721t.Get = []string{"Admin", "TokenOwner"}
        erc721t.GetAllTokensByUser = []string{"Admin", "AccountOwner"}
        erc721t.TotalSupply = []string{"Admin"}
        erc721t.TotalNetSupply = []string{"Admin"}
        erc721t.History = []string{"Admin"}

        erc721a.CreateAccount = []string{"Admin"}
        erc721a.CreateUserAccount = []string{"Admin"}
        erc721a.CreateTokenAccount = []string{"Admin"}
        erc721a.AssociateFungibleTokenToAccount = []string{"Admin",
"AccountOwner"}
        erc721a.GetAllAccounts = []string{"Admin"}
        erc721a.History = []string{"Admin", "AccountOwner"}
        erc721a.GetAccountTransactionHistory = []string{"Admin", "AccountOwner"}
        erc721a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"AccountOwner"}
        erc721a.GetAccountByUser = []string{"Admin", "MultipleAccountOwner"}
        erc721a.BalanceOf = []string{"Admin", "MultipleAccountOwner"}
        erc721as.Get = []string{"Admin", "AccountOwner"}
        erc721as.ActivateAccount = []string{"Admin"}
        erc721as.SuspendAccount = []string{"Admin"}
        erc721as.DeleteAccount = []string{"Admin"}

        erc721r.GetAccountsByRole = []string{"Admin"}
        erc721r.GetUsersByRole = []string{"Admin"}

        var accessMap TokenAccessControl
        accessMap.Token = t
        accessMap.Account = a
        accessMap.AccountStatus = as
        accessMap.Hold = h
        accessMap.Role = r
        accessMap.Admin = ad
        accessMap.Auth = auth
        accessMap.TokenConversion = tc
        accessMap.ERC721ADMIN = erc721ad
        accessMap.ERC721TOKEN = erc721t
        accessMap.ERC721ACCOUNT = erc721a
        accessMap.ERC721AccountStatus = erc721as
```


Copyright © 2025, Oracle and/or its affiliates.

```
accessMap.ERC721ROLE = erc721r
accessMap.ERC721TRANSACTION = erc721trx
```

```
Ctx.ERC721Auth.CheckAuthorization(funcName string, args []string) (bool,
error)
```

Parameters:

- `funcName: string` – The map value between the receivers and methods as described in the `oChainUtil.go` file.
- `...args` – A variable argument where `args[0]` takes the constant `'TOKEN'` and `args[1]` takes the `accountId` parameter to add an access control check for an `AccountOwner`. To add an access control check for a `MultipleAccountOwner`, `args[1]` takes the `orgId` parameter and `args[2]` takes the `userId` parameter. To add an access control check for a `TokenOwner`, `args[1]` takes the `tokenId` parameter.

Returns:

- A Boolean response and, if required, an error.

Examples:

```
t.Ctx.ERC721Auth.CheckAuthorization(<parameters>)
```

`Admin` access

```
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetAllTokens", "TOKEN")
```

`AccountOwner` access

```
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.History", "TOKEN",
accountId)
```

`MultipleAccountOwner` access

```
t.Ctx.ERC721Auth.CheckAuthorization("ERC721ACCOUNT.BalanceOf", "TOKEN",
orgId, userId)
```

`TokenOwner` access

```
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.Get", "TOKEN", tokenId)
```

Methods for Token Configuration Management

`CreateToken`
This method creates a token and saves its properties in the state database. This method can be called only by users with the minter role.

```
Ctx.ERC721Token.CreateToken(args ...interface{})
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- A variable argument where `args[0]` contains a reference to the token `struct` of the required type.

Returns:

- On success, an `interface[]` with token details. On error, a non-nil object with an error message.

Example:

```
t.Ctx.ERC721Token.CreateToken(&tokenAsset)

{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",

    "CreationDate": "2022-04-06T08:16:53Z",
    "IsBurned": false,
     "Mintable": {
        "Max_mint_quantity": 20000
    },
    "NftBasePrice": 100,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "Symbol": "PNT",
    "TokenDesc": "token Description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "paintingnft",
    "TokenStandard": "erc721+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetAllTokens`
This method returns all of the token assets that are saved in the state database. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC721Token.GetAllTokens()  (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a map of all of the token assets. On error, a non-nil object containing an error message.

Example:
```
t.Ctx.ERC721Token.GetAllTokens()

[
    {
        "key": "monalisa",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintab

le",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "CreationDate": "2022-04-10T11:01:42Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 0,
            "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "token Description",
            "TokenId": "monalisa",
            "TokenMetadata": {
                 "Description": "Mona Lisa Painting",
                 "Image": "monalisa.jpeg",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "PainterName": "Leonardo_da_Vinci",
                    "PaintingName": "Mona_Lisa"
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            }
        },
        {
            "key": "monalisa2",
            "valueJson": {
                "AssetType": "otoken",
                "Behavior": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "CreationDate": "2022-04-10T11:04:44Z",
                "IsBurned": false,
                "Mintable": {
                    "Max_mint_quantity": 20000
                },
                "NftBasePrice": 100,
                "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "Roles": {
                    "burner_role_name": "burner",
                    "minter_role_name": "minter"
                },
                "Symbol": "PNT",
                "TokenDesc": "",
                "TokenId": "monalisa1",
                "TokenMetadata": {
                    "Description": "Mona Lisa Painting",
                    "Image": "monalisa.jpeg",
                    "PainterName": "Leonardo_da_Vinci",
                    "PaintingName": "Mona_Lisa"
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
        }
    }
]
```

GetAllTokensByUser
This method returns all tokens that are owned by a specified account ID. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC721Token.GetAllTokensByUser(accountId string) (interface{}, error)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a map of token assets for the specified account. On error, a non-nil object containing an error message.

Example:
`t.Ctx.ERC721Token.GetAllTokensByUser(accountId)`

```
[
    {
        "key": "monalisa",
        "valueJson": {
            "AssetType": "otoken",
            "Behavior": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "CreationDate": "2022-04-10T11:01:42Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 0,
            "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "token Description",
            "TokenId": "monalisa",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "TokenMetadata": {
                     "Description": "Mona Lisa Painting",
                     "Image": "monalisa.jpeg",
                     "PainterName": "Leonardo_da_Vinci",
                     "PaintingName": "Mona_Lisa"
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
            }
        },
        {
            "key": "monalisa2",
            "valueJson": {
                "AssetType": "otoken",
                "Behavior": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "CreatedBy":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "CreationDate": "2022-04-10T11:04:44Z",
                "IsBurned": false,
                "Mintable": {
                    "Max_mint_quantity": 20000
                },
                "NftBasePrice": 100,
                "Owner":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "Roles": {
                    "burner_role_name": "burner",
                    "minter_role_name": "minter"
                },
                "Symbol": "PNT",
                "TokenDesc": "",
                "TokenId": "monalisa1",
                "TokenMetadata": {
                     "Description": "Mona Lisa Painting",
                     "Image": "monalisa.jpeg",
                     "PainterName": "Leonardo_da_Vinci",
                     "PaintingName": "Mona_Lisa"
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
        }
        }
    }
]
```

GetTokenUri
This method returns the token URI for a specified token.

```
Ctx.ERC721Token.GetTokenURI(tokenId string) (interface{}, error)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, it returns a map of the new token URI in the string data type. On error, a non-nil object with an error message.

Example:
`t.Ctx.ERC721Token.GetTokenURI(tokenId)`

```
{"TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"}
```

History
This method returns history for the specified token.

```
Ctx.ERC721Token.History(tokenId: string)  (interface{}, error)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, an array of maps. On error, a rejection with an error message.

Example:
`t.Ctx.ERC721Token.History(tokenId)`

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T11:34:06z",
        "TxId":
"3184eac8738c73ef45501fe23c9e14517892e04e4eb03ec9be834b89c29ea17b",
        "Value": {
            "AssetType": "otoken",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "Behavior": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "BurnedBy": null,
                "BurnedDate": null,
                "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "CreationDate": "2022-04-06T11:33:40+05:30",
                "IsBurned": null,
                "Mintable": {
                    "Max_mint_quantity": 20000
                },
                "NftBasePrice": 0,
                "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "Roles": {
                    "burner_role_name": "burner",
                    "minter_role_name": "minter"
                },
                "Symbol": "PNT",
                "TokenDesc": "",
                "TokenId": "t1",
                "TokenMetadata": {
                    "Description": "",
                    "Image": "",
                    "PainterName": "",
                    "PaintingName": ""
                },
                "TokenName": "paintingnft",
                "TokenStandard": "erc721+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "",
                "TransferredBy": null,
                "TransferredDate": null
            }
        },
        {
            "IsDelete": "false",
            "Timestamp": "2022-04-06T11:33:40z",
            "TxId":
"d37dba907a849c308b2a38d47cf8a68cdcb4e3d93fa74050774379fccfcd43be",
            "Value": {
                "AssetType": "otoken",
                "Behavior": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "CreationDate": "2022-04-06T11:33:40+05:30",
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "NftBasePrice": 0,
            "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "Symbol": "PNT",
            "TokenDesc": "",
            "TokenId": "t1",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "PainterName": "",
                "PaintingName": ""
            },
            "TokenName": "paintingnft",
            "TokenStandard": "erc721+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": ""
        }
    }
]
```

Get

This method returns the specified token object if it is present in the state database.

```
Ctx.Get(Id string, result ...interface{}) (interface{}, error)
```

Parameters:

- `tokenId: string` – The ID of the token.
- `result` – A variable argument, where the first argument `result[0]` is a reference to an empty `Token` object of the correct type, which will contain the token data after a successful call of the method.

Returns:

- On success, a map with the token asset data. Also, if `result[0]` is passed then the data is assigned to `result[0]`. On error, a non-nil object containing an error message.

Example:

Copyright © 2025, Oracle and/or its affiliates.

```
t.Ctx.ERC721Token.Get(tokenId, &asset)

{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2022-04-06T00:35:42z",
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "NftBasePrice": 200,
    "Owner":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba",
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "Symbol": "PNT",
    "TokenDesc": "Token Description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "paintingnft",
    "TokenStandard": "erc721+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
    "TransferredBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "TransferredDate": "2022-04-06T00:51:56z"
}
```

OwnerOf
This method returns the account ID of the owner of a specified token.

```
Ctx.ERC721Token.OwnerOf(tokenId string) (interface{}, error)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `tokenId: string` – The ID of the token.

Returns:

- On success, a map of the account ID of the owner in the string data type. On error, a non-nil object containing an error message.

Example:

```
t.Ctx.ERC721Token.OwnerOf(tokenId)

{"Owner":
"oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba"}
```

`Name`

This method returns the name of the token class.

```
Ctx.ERC721Token.Name() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a map of the token name in the string data type. On error, a non-nil object containing an error message.

Example:

```
t.Ctx.ERC721Token.Name()

{"TokenName": "paintingnft"}
```

`Symbol`

This method returns the symbol of the token class.

```
Ctx.ERC721Token.Symbol() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a map of the symbol in the string data type. On error, a non-nil object containing an error message.

Example:

```
t.Ctx.ERC721Token.Symbol()

{"Symbol": "PNT"}
```


Copyright © 2025, Oracle and/or its affiliates.

TokenUri
This method returns the token URI for a specified token.

```
Ctx.ERC721Token.TokenURI(tokenId string) (interface{}, error)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, it returns a map of the new token URI in the string data type. On error, a non-nil object with an error message.

Return Value Example:
`t.Ctx.ERC721Token.TokenURI(tokenId)`

```
{"TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"}
```

TotalSupply
This method returns the total number of minted NFTs.

```
Ctx.ERC721Token.TotalSupply() (map[string]interface{}, error)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a map of the total supply of tokens in the number data type. On error, a rejection with an error message.

Example:
`t.Ctx.ERC721Token.TotalSupply();`

```
{"TotalSupply": 3}
```

UpdateToken
This method updates token properties. This method can be called only by the owner or creator of the token. After a token asset is created, only the token owner can update the token custom properties. If the user is both token owner and creator of a token, they can also update the `TokenDesc` property. Token metadata cannot be updated. You must pass all token properties to this method, even if you want to update only certain properties.

```
Ctx.ERC721Token.UpdateToken(asset interface{}) (interface{}, error)
```

Parameters:

- A reference to the token struct data of the required type

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise message with token details. On error, a rejection with an error message.

Example:
`t.Ctx.ERC721Token.UpdateToken(&asset)`

```
{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",

    "CreationDate": "2022-04-06T08:16:53Z",
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "NftBasePrice": 200,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "Symbol": "PNT",
    "TokenDesc": "Token Description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "paintingnft",
    "TokenStandard": "erc721+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg",
}
```

`GetNewCtx`
This method returns a new `TrxContext` object. The `trxcontext` struct holds references to all of the SDK libraries. Access the sdk methods by using only this object. The `trxcontext` object

Copyright © 2025, Oracle and/or its affiliates.

maintains the mutual exclusivity of transaction stubs in SDK libraries when concurrent transactions are running.

```
GetNewCtx(stub shim.ChaincodeStubInterface) TrxContext
```

Parameters:

- `stub` – The transaction stub.

Returns:

- A `trxcontext` struct.

Example:
```
trxcontext.GetNewCtx(stub)

trxcontext object.
```

Methods for Account Management

`GenerateAccountId`
This method returns an account ID, which is formed by concatenating the membership service provider ID (`orgId`) and the user name or email ID (`userId`) and then creating a SHA-256 hash.

```
Ctx.ERC721Account.GenerateAccountId(orgId string, userId string) (string,
error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, the generated account ID. On error, a non-nil object containing an error message.

Example:
```
t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)

oaccount~a0a60d54ba9e2ff349737d292ea10ebd9cc8f1991c11443c19d20aea299a9507
```

`CreateAccount`
This method creates an account for a specified user. An account must be created for any user who will have tokens at any point. Accounts track the number of NFTs a user has. Users must have accounts in the network to complete token-related operations. You can create only one NFT account per user.
An account ID is an alphanumeric set of characters, prefixed with `oaccount~` and followed by an SHA-256 hash of the membership service provider ID (`org_id`) of the user in the current

Copyright © 2025, Oracle and/or its affiliates.

network organization, the user name or email ID (`userId`) of the instance owner or the user who is logged in to the instance, and the constant string `nft`.

```
Ctx.ERC721Account.CreateAccount(orgId string, userId string, tokenType
string) (ERC721Account, error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenType: string` – The only supported token type is `nonfungible`.

Returns:

- On success, the new account object. On error, a non-nil object containing an error message

Example:

```
t.Ctx.ERC721Account.CreateAccount(orgId, userId, tokenType)

{
    "AssetType": "oaccount",
    "AccountId":
"oaccount~a0a60d54ba9e2ff349737d292ea10ebd9cc8f1991c11443c19d20aea299a9507",
    "UserId": "admin",
    "BapAccountVersion" : 0,
    "OrgId": "Org1MSP",
    "TokenType": "nonfungible",
    "NoOfNfts": 0
}
```

`BalanceOf`

This method returns the total number of NFTs the specified user holds.

```
Ctx.ERC721Account.BalanceOf(accountId string) (interface{}, error)
```

Parameters:

- `accountId: string` – The account ID of the user.

Returns:

- On success, an interface with a message and the total number of NFTs. On error, a non-nil error object that contains an error message.

Example:

```
t.Ctx.ERC721Account.BalanceOf(accountId)

{"TotalNfts": 0}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAllAccounts
This method returns a list of all accounts. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC721Account.GetAllAccounts() (interface{}, error)
```

Parameters:

- none

Returns:

- A JSON array of all accounts.

Example:
```
t.Ctx.ERC721Account.GetAllAccounts()

[
    {
        "key":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
        "valueJson": {
            "AccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "BapAccountVersion" : 0,
            "AssetType": "oaccount",
            "NoOfNfts": 5,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user1"
        }
    },
    {
        "key":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
        "valueJson": {
            "AccountId":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
            "AssetType": "oaccount",
            "BapAccountVersion" : 0,
            "NoOfNfts": 0,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user_minter"
        }
    },
    {
        "key":
"oaccount~5541fb520058d83664b844e7a55fe98d574ddeda765d0e795d4779e9ccc271ce",
        "valueJson": {
            "AccountId":
"oaccount~5541fb520058d83664b844e7a55fe98d574ddeda765d0e795d4779e9ccc271ce",
            "AssetType": "oaccount",
            "BapAccountVersion" : 0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "NoOfNfts": 0,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user_burner"
        }
    }
]
```

GetAccountByUser
This method returns account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC721Account.GetAccountByUser(orgId, userId) (ERC721Account, error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `AccountId` – The ID of the user account.
- `UserId` – The user name or email ID of the user.
- `OrgId` – The membership service provider (MSP) ID of the user in the current organization.
- `TokenType` – The type of token that the account holds.
- `NoOfNfts` – The total number of NFTs held by the account.
- On error, a non-nil object that contains an error message.

Example:

```
{
    "AssetType": "oaccount",
    "AccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "BapAccountVersion" : 0,
    "UserId": "admin",
    "OrgId": "Org1MSP",
    "TokenType": "nonfungible",
    "NoOfNfts": 0
}
```

GetUserByAccountId
This method returns the user details for a specified account.

```
Ctx.ERC721Account.GetUserByAccountId(accountId string) (interface{}, error)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `accountId: string` – The ID of the account.

Returns:

- On success, a JSON object that includes user details in the following properties:
  - `OrgId` – The membership service provider (MSP) ID of the user in the current network organization.
  - `UserId` – The user name or email ID of the user.
- On error, a non-nil object containing an error message.

Example:
`t.Ctx.ERC721Account.GetUserByAccountById(accountId)`

```
{
    "OrgId": "Org1MSP",
    "UserId": "admin"
}
```

`History`
This method returns an array of the account history details for a specified account.

`Ctx.ERC721Account.History(accountId string) (interface{}, error)`

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a `map[string]interface{}` array that contains the account history details for the specified account. The account data is shown under the `value` key in the map. On error, a non-nil error object containing an error message.

Example:
`t.Ctx.ERC721Account.History(accountId)`

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T08:16:53Z",
        "TxId":
"750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7",
        "Value": {
            "AccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "AssetType": "oaccount",
            "BapAccountVersion" : 1,
            "NoOfNfts": 1,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user1"
        }
    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    {
        "IsDelete": "false",
        "Timestamp": "2022-04-06T08:15:19Z",
        "TxId":
"49eb84c42d452e5ba0028d8ebb4190454cf9013a11c0bad3e96594af452d4982",
        "Value": {
            "AccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
            "AssetType": "oaccount",
            "NoOfNfts": 0,
            "BapAccountVersion" : 0,
            "OrgId": "apPart",
            "TokenType": "nonfungible",
            "UserId": "user1"
        }
    }
]
```

`GetAccountWithStatusByUser`
This method returns account details for a specified user, including account status. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC721Account.GetAccountWithStatusByUser(orgId, userId) (interface{},
error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes the following properties:
- `AccountId` – The ID of the user account.
- `UserId` – The user name or email ID of the user.
- `OrgId` – The membership service provider (MSP) ID of the user in the current organization.
- `TokenType` – The type of token that the account holds.
- `NoOfNfts` – The total number of NFTs held by the account.
- `BapAccountVersion` – An account object parameter for internal use.
- `Status` – The current status of the user account.
- On error, a non-nil object that contains an error message.

Example:

```
{
  "AccountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
  "AssetType": "oaccount",
  "BapAccountVersion": 0,
  "NoOfNfts": 0,
  "OrgId": "appdev",
  "Status": "active",
  "TokenType": "nonfungible",
  "UserId": "idcqa"
}
```

Methods for Role Management

`AddRoleMember`
This method adds a role to a specified user.

```
Ctx.ERC721Token.AddRoleMember(role string, accountId string) (interface{},
error)
```

Parameters:

- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `accountId: number` – The account ID to operate on.

Returns:

- On success, a map with a success message. On error, a non-nil object containing an error message.

Example:
`t.Ctx.ERC721Token.AddRoleMember(userRole, accountId)`

```
{"msg": "Successfully added role minter to
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(orgId : Org1MSP, userId : admin)"}
```

`RemoveRoleMember`
This method removes a role from a specified user and token. An account ID is formed by creating an SHA-256 hash of the concatenated membership service provider ID (`orgId`) and the user name or email ID (`userId`).

```
Ctx.Token.RemoveRoleMember(role string, accountId string) (interface{}, error)
```

Parameters:

- `role: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `accountId: number` – The account ID to operate on.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a map with a success message. On error, a non-nil object containing an error message.

Example:
```
t.Ctx.ERC721Token.RemoveRoleMember(userRole, accountId)

{"msg": "successfully removed memberId
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(orgId : Org1MSP, userId : admin) from role minter"}
```

`GetAccountsByRole`
This method returns a list of all account IDs for a specified role.

```
Ctx.ERC721Role.GetAccountsByRole(roleName string) (interface{}, error)
```

Parameters:

- `roleName: string` – The name of the role to search for.

Returns:

- On success, a JSON array of account IDs. On error, a non-nil error object that contains an error message.

Example:
```
t.Ctx.ERC721Role.GetAccountsByRole(userRole)

{
    "accounts": [

"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d"
    ]
}
```

`GetUsersByRole`
This method returns a list of all users for a specified role.

```
Ctx.ERC721Role.GetUsersByRole(roleName string) (interface{}, error)
```

Parameters:

- `roleName: string` – The name of the role to search for.

Returns:

- On success, a JSON array of user objects. Each object contains the user ID and organization ID. On error, a non-nil error object that contains an error message.

Example:
```
t.Ctx.ERC721Role.GetUsersByRole(userRole)

{
    "Users": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
        {
            "OrgId": "Org1MSP",
            "UserId": "admin"
        }
    ]
}
```

IsInRole
This method returns a Boolean value to indicate if a user and token has a specified role. An account ID is formed by creating an SHA-256 hash of the concatenated membership service provider ID (`orgId`) and the user name or email ID (`userId`).

```
Ctx.ERC721Token.IsInRole(role string, accountId string) (bool, error)
```

Parameters:

- `role: string` – The name of the role to check for the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file.
- `accountId: number` – The account ID to operate on.

Returns:

- On success, a Boolean value that is true if the role is present for the specified account ID, otherwise false. On error, a non-nil object containing an error message

Example:

```
t.Ctx.ERC721Token.IsInRole(userRole, accountId)

{"result": false}
```

Methods for Transaction History Management

GetAccountTransactionHistory
This method returns an array of the transaction history details for a specified account.

```
Ctx.ERC721Account.GetAccountTransactionHistory(accountId string)
(interface{}, error)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, an array of account transaction objects in JSON format:
  - `TransactionId` – The ID of the transaction.
  - `TransactedAccount` – The account with which the transaction took place.
  - `TransactionType` – The type of transaction.

Copyright © 2025, Oracle and/or its affiliates.

- `Timestamp` – The time of the transaction.
- On error, a non-nil error object that contains an error message.

Example:

```
t.Ctx.ERC721Account.GetAccountTransactionHistory(accountId)

[
     {
        "Timestamp": "2022-04-06T08:31:39Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
        "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
",
        "TransactionType": "DEBIT"
    }
    {
        "Timestamp": "2022-04-06T08:16:53Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
        "TransactionId":
"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
",
        "TransactionType": "MINT"
    }
]
```

`GetAccountTransactionHistoryWithFilters`

This method returns account transaction history for a specified user, filtered by `PageSize`, `Bookmark`, `startTime` and `endTime`. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC721Account.GetAccountTransactionHistoryWithFilters(accountId string,
filters ...erc721Account.AccountHistoryFilters)
```

Parameters:

- `accountId: string` – ID of the account.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Returns:

- On success, an array of account transaction objects in JSON format:
  - `TransactionId` – The ID of the transaction.

Copyright © 2025, Oracle and/or its affiliates.

- `TransactedAccount` – The account with which the transaction took place.
- `TransactionType` – The type of transaction.
- `Timestamp` – The time of the transaction.
- On error, a non-nil error object that contains an error message.

Example:
`t.Ctx.ERC721Account.GetAccountTransactionHistoryWithFilters(accountId, filters...)`

```
[
     {
        "Timestamp": "2022-04-06T08:31:39Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
        "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
",
        "TransactionType": "DEBIT"
    }
    {
        "Timestamp": "2022-04-06T08:16:53Z",
        "TokenId": "monalisa",
        "TransactedAccount":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
        "TransactionId":
"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
",
        "TransactionType": "MINT"
    }
]
```

`GetTransactionById`
This method returns the history of a `Transaction` asset.

```
Ctx.ERC721Transaction.GetTransactionById(trxId string) (interface{}, error)
```

Parameters:

- `trxId: string` – The ID of the transaction asset.

Returns:

- On success, an array of maps of transaction assets. On error, a non-nil error object that contains an error message.

Example:
`t.Ctx.ERC721Transaction.GetTransactionById(transactionId)`

```
{
    "History": [
        {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "IsDelete": "false",
            "Timestamp": "2022-04-06T08:31:39Z",
            "TxId":
"5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7",
            "Value": {
                "AssetType": "otransaction",
                "Data": "",
                "FromAccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1",
                "Timestamp": "2022-04-06T08:31:39Z",
                "ToAccountId":
"oaccount~0829f0996744ca9dc8b4e9165a7a8f5db3fdffdc46c96b94f5d625041502cec4",
                "TokenId": "monalisa",
                "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
",
                "TransactionType": "TRANSFER",
                "TriggeredByAccountId":
"oaccount~543c2258e351c3e7a40ea59b81e62154d38fbfc9d1b5b79f30ac5e08e7d0dfd1"
            }
        }
    ],
    "TransactionId":
"otransaction~5a353e02e657c2c8fddce41dd4e7260025fe7beef634ca3351fc366a440e8ac7
"
}
```

`DeleteHistoricalTransactions`
This method deletes transactions that are older than a specified date from the state database.

```
func (t *Controller) DeleteHistoricalTransactions(referenceTime string)
(interface{}, error)
```

Parameters:

- `referenceTime: string` – Transactions older than the specified time will be deleted.

Returns:

- On success, an array of the deleted transaction IDs and a success message. On error, a non-nil error object that contains an error message.

Example:

```
t.Ctx.ERC721Transaction.DeleteHistoricalTransactions(timestamp)
```

```
{
    "Transactions": [

"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
"
    ],
    "msg": "Successfuly deleted transaction older than
```


Copyright © 2025, Oracle and/or its affiliates.

```
date:2022-04-06T08:17:53Z"
}
```

Token Behavior Management - Mintable Behavior

`GetMaxMintQuantity`
This method returns the maximum mintable quantity of a token. If the `max_mint_quantity` behavior is not configured in the specification file, then the default value is 0 and an infinite number of tokens can be minted.

```
Ctx.ERC721Token.GetMaxMintQuantity(id string) (float64, error)
```

Parameters:

- `id` – The ID of the token to operate on.

Returns:

- On success, the maximum mintable quantity of the token, in the number data type. On error, a non-nil object with an error message.

Example:
```
t.Ctx.ERC721Token.GetMaxMintQuantity(tokenId);

20000
```

`GetTotalMintedTokens`
This method returns the total net number of tokens available in the system for the specified token. The net number of tokens available is the total number of minted tokens minus the number of burned tokens.

```
Ctx.ERC721Token.GetTotalMintedTokens() (map[string]interface{}, error)
```

Parameters:

- none

Returns:

- On success, a map of the total minted tokens, in the number data type, and a success message. On error, a non-nil object with an error message.

Example:
```
t.Ctx.ERC721Token.GetTotalMintedTokens()

{"TotalNetSupply": 5}
```

Token Behavior Management - Transferable Behavior

`SafeTransferFrom`
This method transfers ownership of the specified NFT from the caller to another account. This method includes the following validations:

- The token exists and is not burned.

Copyright © 2025, Oracle and/or its affiliates.

- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.
- The caller of the function is the sender.

```
Ctx.ERC721Token.SafeTransferFrom(fromAccountId string, toAccountId string,
tokenAsset interface{}, data ...string) (interface{}, error)
```

Parameters:

- `fromAccountId: string` – The account ID of the sender in the current organization.
- `toAccountId: string` – The account ID of the receiver in the current organization.
- `tokenAsset` – The reference to the token asset to operate on.
- `data: string` – Optional additional information to store in the transaction.

Returns:

- On success, a promise with a success message that includes account details. On error, a non-nil object with an error message.

Example:

```
t.Ctx.ERC721Token.SafeTransferFrom(fromAccountId, toAccountId,
tokenAssetValue.Interface(), data...)
```

```
{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin) to Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1"}
```

`TransferFrom`

This method transfers ownership of the specified NFT from a sender account to a receiver account. It is the responsibility of the caller to pass the correct parameters. This method can be called by any user, not only the token owner. This method includes the following validations:

- The token exists and is not burned.
- The sender account and receiver account exist and are not the same account.
- The sender account owns the token.

```
Ctx.ERC721Token.TransferFrom(fromAccountId string, toAccountId string,
tokenAsset interface{}) (interface{}, error)
```

Parameters:

- `fromAccountId: string` – The account ID of the sender in the current organization.
- `toAccountId: string` – The account ID of the receiver in the current organization.
- `tokenAsset` – The reference to the token asset to operate on.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a promise with a success message that includes account details. Account IDs have the prefix `oaccount~`. On error, a non-nil object with an error message.

\Example:
```
t.Ctx.ERC721Token.TransferFrom(fromAccountId, toAccountId,
tokenAssetValue.Interface())

{"msg": "Successfully transferred NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin) to Account-Id:
oaccount~ec32cff8635a056f3dda3da70b1d6090d61f66c6a170c4a95fd008181f729dba
(Org-Id: Org1MSP, User-Id: user1"}
```

Token Behavior Management - Burnable Behavior

`Burn`
This method deactivates, or burns, the specified NFT from the caller's account. The caller of this method must have an account. A token cannot be burned unless the token specification file includes the `burnable` behavior. If no `burner_role_name` property is specified in the `roles` section of the specification file, then the owner of the token can burn the token. If a `burner_role_name` property is specified in the `roles` section, then the user assigned the burner role who is also the minter (creator) of the token can burn the token. The `burn` method is part of the `ERC721Token` package, which you access via the receiver of the `Ctx` struct.

```
Ctx.Token.Burn(tokenAsset interface{}) (interface{}, error)
```

Parameters:

- `tokenAsset` – The reference to the token asset to operate on.

Returns:

- On success, a promise with a success message that includes account details. On error, a non-nil object with an error message.

Example:
```
t.Ctx.ERC721Token.Burn(tokenAssetValue.Interface())

{"msg": "Successfully burned NFT token: 'monalisa' from Account-Id:
oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d
(Org-Id: Org1MSP, User-Id: admin"}
```

Go Methods for ERC-721 NFT Locking

Blockchain App Builder automatically generates methods that you can use to lock non-fungible tokens that use the extended ERC-721 standard.

A locked token cannot be burned or transferred to other users. All other properties, such as the token's state, owner, and history are preserved. You can use the NFT locking functionality when transferring a token to another blockchain network, such as Ethereum or Polygon.

Before you can lock NFTs, you must assign the vault manager role to a user. The vault manager is a special type of role, a `TokenSys` role. `TokenSys` roles are different from asset-based roles such as burner, minter, and notary, and from administrative roles such as `Token Admin` and `Org Admin`. Currently Blockchain App Builder supports the `vault TokenSys` role. The

Copyright © 2025, Oracle and/or its affiliates.

single user who has the `vault` role for a chaincode is the vault manager of the chaincode, and can manage locked NFTs.

The typical flow for using the NFT locking functionality follows these steps.

- Create a non-fungible token that has the lockable behavior.
- Use the `AddTokenSysRole` method to give the `vault` role to a user, the vault manager.
- Call the `LockNFT` method to lock a non-fungible token, specified by the token ID.

TokenSys Role Management Methods

`AddTokenSysRole`
This method adds a `TokenSys` role to a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenSysRole(role string, orgId string, userId
string) (interface{}, error) {
    accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.AddTokenSysRole", "TOKEN",
map[string]string{"accountId": accountId})
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Token.AddTokenSysRoleMember(role, accountId)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role to give to the user.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully added role 'vault' to Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

IsInTokenSysRole
This method returns a Boolean value to indicate if a user has a specified TokenSys role. This method can be called only by a Token Admin of the chaincode.

```
func (t *Controller) IsInTokenSysRole(orgId string, userId string, role
string) (interface{}, error) {
    accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.IsInTokenSysRole", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Token.IsInTokenSysRoleMember(role, accountId)
}
```

Parameters:

- role: string – The name of the TokenSys role to check.
- orgId: string – The membership service provider (MSP) ID of the user in the current organization.
- userId: string – The user name or email ID of the user.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "result": true,
    "msg": "Account Id
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1) has vault role"
}
```

RemoveTokenSysRole
This method removes a TokenSys role from a specified user. This method can be called only by a Token Admin of the chaincode.

```
func (t *Controller) RemoveTokenSysRole(role string, orgId string, userId
string) (interface{}, error) {
    accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.RemoveTokenSysRole",
"TOKEN", map[string]string{"accountId": accountId})
```


Copyright © 2025, Oracle and/or its affiliates.

```
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Token.RemoveTokenSysRoleMember(role, accountId)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role to remove.
- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully removed role 'vault' from Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

`TransferTokenSysRole`
This method transfers a `TokenSys` role from a user to another user. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) TransferTokenSysRole(role string, fromOrgId string,
fromUserId string, toOrgId string, toUserId string) (interface{}, error) {
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.TransferTokenSysRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
    }
    fromAccountId, err := t.Ctx.ERC721Account.GenerateAccountId(fromOrgId,
fromUserId)
    if err != nil {
        return nil, fmt.Errorf("error in TransferTokenSysRole. Error: %s",
err)
    }
    toAccountId, err := t.Ctx.ERC721Account.GenerateAccountId(toOrgId,
toUserId)
    if err != nil {
        return nil, fmt.Errorf("error in TransferTokenSysRole. Error: %s",
err)
    }
    return t.Ctx.ERC721Token.TransferTokenSysRole(role, fromAccountId,
```


Copyright © 2025, Oracle and/or its affiliates.

```
toAccountId)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role to transfer.
- `fromOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role from.
- `fromUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role from.
- `toOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role to.
- `toUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role to.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully transfered role 'vault' from Account Id:
ouaccount~f4e311528f03fffa7810753d643f66289ff6c9080fcf839902f28a1d3aff1789
(Org-Id: Org1MSP, User-Id: user1) to Account Id:
ouaccount~ae5be2ae8f98d6d32f5d02b43877d987114e7937c7bacbc30390dcce09996a19
(Org-Id: Org1MSP, User-Id: user2)"
}
```

`GetAccountsByTokenSysRole`
This method returns a list of all account IDs for a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetAccountsByTokenSysRole(role string) (interface{},
error) {
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetAccountsByTokenSysRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Token.GetAccountsByTokenSysRole(role)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "accountIds": [

"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba"
    ]
}
```

GetUsersByTokenSysRole
This method returns user information for all users with a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetUsersByTokenSysRole(role string) (interface{}, error)
{
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetUsersByTokenSysRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Token.GetUsersByTokenSysRole(role)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
   "Users":[
      {

"accountId":"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2
caf03eba",
         "orgId":"Org1MSP",
         "userId":"user1"
      }
   ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

NFT Locking Methods

`LockNFT`
This method locks a specified non-fungible token. To lock a token, there must be a user with the `TokenSys vault` role, who acts as the vault manager. This method can be called only by the owner of the token.

```
func (t *Controller) LockNFT(tokenId string) (interface{}, error) {
    return t.Ctx.ERC721Token.LockNFT(tokenId)
}
```

Parameters:

- `tokenID: string` – The ID of the token to lock.

Returns:

- On success, a JSON representation of the token object.

Return Value Example:

```
{
   "AssetType":"otoken",
   "Behavior":[
      "indivisible",
      "singleton",
      "mintable",
      "transferable",
      "lockable",
      "burnable",
      "roles"
   ],

"CreatedBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
   "CreationDate":"2023-10-20T12:39:50Z",
   "IsBurned":false,
   "IsLocked":true,
   "Mintable":{
      "Max_mint_quantity":20000
   },
   "On_sale_flag":false,

"Owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
   "Price":120,
   "Roles":{
      "minter_role_name":"minter"
   },
   "Symbol":"ART",
   "TokenDesc":"",
   "TokenId":"token1",
   "TokenMetadata":{
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "Description":"",
      "Image":"",
      "Painter_name":"",
      "Painting_name":""
   },
   "TokenName":"artcollection",
   "TokenStandard":"erc721+",
   "TokenType":"nonfungible",
   "TokenUnit":"whole",
   "TokenUri":"token1.example.com"
}
```

IsNFTLocked

This method returns a Boolean value to indicate if a specified token is locked. This method can be called only by the token owner, the vault manager (the user with the `TokenSys vault` role), or a `Token Admin` of the chaincode.

```
func (t *Controller) IsNFTLocked(tokenId string) (interface{}, error) {
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.IsNFTLocked", "TOKEN",
map[string]string{"tokenId": tokenId})
    if err != nil && !auth {
        isCallerTokenSysRoleHolder, error :=
t.Ctx.ERC721Token.IsCallerTokenSysRoleHolder(constants.Vault)
        if error != nil {
            return nil, error
        }
        if !isCallerTokenSysRoleHolder {
            return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
        }
    }
    return t.Ctx.ERC721Token.IsNFTLocked(tokenId)
}
```

Parameters:

- `tokenID: string` – The ID of the token.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
   "isNFTLocked":true
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAllLockedNFTs
This method returns a list of all locked non-fungible tokens. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
func (t *Controller) GetAllLockedNFTs() (interface{}, error) {
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetAllLockedNFTs", "TOKEN")
    if err != nil && !auth {
        isCallerTokenSysRoleHolder, error :=
t.Ctx.ERC721Token.IsCallerTokenSysRoleHolder(constants.Vault)
        if error != nil {
            return nil, error
        }
        if !isCallerTokenSysRoleHolder {
            return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
        }
    }
    return t.Ctx.ERC721Token.GetAllLockedNFTs()
}
```

Parameters:

- None

Returns:

- On success, an array of the locked non-fungible token objects.

Return Value Example:

```
[
   {
      "key":"token1",
      "valueJson":{
         "AssetType":"otoken",
         "Behavior":[
            "indivisible",
            "singleton",
            "mintable",
            "transferable",
            "lockable",
            "burnable",
            "roles"
         ],

"CreatedBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "CreationDate":"2023-10-20T12:39:50Z",
         "IsBurned":false,
         "IsLocked":true,
         "Mintable":{
            "Max_mint_quantity":20000
         },
```


Copyright © 2025, Oracle and/or its affiliates.

```
         "On_sale_flag":false,

"Owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "Price":120,
         "Roles":{
            "minter_role_name":"minter"
         },
         "Symbol":"ART",
         "TokenDesc":"",
         "TokenId":"token1",
         "TokenMetadata":{
            "Description":"",
            "Image":"",
            "Painter_name":"",
            "Painting_name":""
         },
         "TokenName":"artcollection",
         "TokenStandard":"erc721+",
         "TokenType":"nonfungible",
         "TokenUnit":"whole",
         "TokenUri":"token1.example.com"
      }
   }
]
```

`GetAllLockedNFTsByOrg`
This method returns a list of all locked non-fungible tokens for a specified organization and optionally a specified user. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
func (t *Controller) GetLockedNFTsByOrg(orgId string, userId ...string)
(interface{}, error) {
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721TOKEN.GetLockedNFTsByOrg", "TOKEN")
    if err != nil && !auth {
        isCallerTokenSysRoleHolder, error :=
t.Ctx.ERC721Token.IsCallerTokenSysRoleHolder(constants.Vault)
        if error != nil {
            return nil, error
        }
        if !isCallerTokenSysRoleHolder {
            return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
        }
    }
    return t.Ctx.ERC721Token.GetLockedNFTsByOrg(orgId, userId...)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `userId: string` – The user name or email ID of the user (optional).

Returns:

- On success, an array of the locked non-fungible token objects.

Return Value Example:


```
[
   {
      "key":"token1",
      "valueJson":{
         "AssetType":"otoken",
         "Behavior":[
            "indivisible",
            "singleton",
            "mintable",
            "transferable",
            "lockable",
            "burnable",
            "roles"
         ],

"CreatedBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "CreationDate":"2023-10-20T12:39:50Z",
         "IsBurned":false,
         "IsLocked":true,
         "Mintable":{
            "Max_mint_quantity":20000
         },
         "On_sale_flag":false,

"Owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "Price":120,
         "Roles":{
            "minter_role_name":"minter"
         },
         "Symbol":"ART",
         "TokenDesc":"",
         "TokenId":"token1",
         "TokenMetadata":{
            "Description":"",
            "Image":"",
            "Painter_name":"",
            "Painting_name":""
         },
         "TokenName":"artcollection",
         "TokenStandard":"erc721+",
         "TokenType":"nonfungible",
         "TokenUnit":"whole",
         "TokenUri":"token1.example.com"
      }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
]
```

Go Methods for ERC-721 Token Account Status

Blockchain App Builder automatically generates methods that you can use to manage account status for tokens that use the extended ERC-721 standard.

You can use the following methods to put token user accounts in the active, suspended, or deleted states.

When an account is suspended, the account user cannot complete any write operations, which include minting, burning, and transferring tokens. Additionally, other users cannot transfer tokens to a suspended account. A suspended account can still complete read operations.

An account with a non-zero token balance cannot be deleted. You must transfer or burn all tokens in an account before you can delete the account. After an account is in the deleted state, the account state cannot be changed back to active or suspended.

- Automatically Generated Account Status Methods
- Account Status SDK Methods

Automatically Generated Account Status Methods

`GetAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
func (t *Controller) GetAccountStatus(orgId string, userId string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating accountId
of (Org-Id: %s, User-Id: %s)", orgId, userId)
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721AccountStatus.Get", "TOKEN",
map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      accountStatus, err :=
t.Ctx.ERC721AccountStatus.GetAccountStatus(accountId)
      if err != nil {
            return
t.Ctx.ERC721AccountStatus.GetDefaultAccountStatus(accountId)
      }
      return accountStatus, nil
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`GetAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
func (t *Controller) GetAccountStatusHistory(orgId string, userId string)
(interface{}, error) {
      accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating accountId
of (Org-Id: %s, User-Id: %s)", orgId, userId)
      }
      _, err = t.Ctx.ERC721Account.GetAccount(accountId)
      if err != nil {
            return nil, fmt.Errorf("error in GetAccountStatusHistory: %s",
err)
      }
      auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721AccountStatus.Get", "TOKEN",
map[string]string{"accountId": accountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      statusId, err :=
t.Ctx.ERC721AccountStatus.GenerateAccountStatusId(accountId)
      if err != nil {
            return nil, err
      }
      accountStatusHistory, err := t.Ctx.ERC721AccountStatus.History(statusId)
      if err != nil {
            return []map[string]interface{}{}, nil
```


Copyright © 2025, Oracle and/or its affiliates.

```
      }
      return accountStatusHistory, nil
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, the account status history in JSON format.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

`ActivateAccount`
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode. Deleted accounts cannot be activated. For any accounts created prior to the

Copyright © 2025, Oracle and/or its affiliates.

account status functionality being available, you must call this method to set the account status as `active`.

```
func (t *Controller) ActivateAccount(orgId string, userId string)
(interface{}, error) {
    accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
    if err != nil {
        return nil, fmt.Errorf("error in getting the generating accountId of
(Org-Id: %s, User-Id: %s)", orgId, userId)
    }
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721AccountStatus.ActivateAccount",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Account.ActivateAccount(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`SuspendAccount`

This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
func (t *Controller) SuspendAccount(orgId string, userId string)
(interface{}, error) {
    accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
    if err != nil {
        return nil, fmt.Errorf("error in getting the generating accountId of
(Org-Id: %s, User-Id: %s)", orgId, userId)
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721AccountStatus.SuspendAccount",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC721Account.SuspendAccount(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

`DeleteAccount`
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance must be zero.

```
func (t *Controller) DeleteAccount(orgId string, userId string) (interface{},
error) {
    accountId, err := t.Ctx.ERC721Account.GenerateAccountId(orgId, userId)
    if err != nil {
        return nil, fmt.Errorf("error in getting the generating accountId of
(Org-Id: %s, User-Id: %s)", orgId, userId)
    }
    auth, err :=
t.Ctx.ERC721Auth.CheckAuthorization("ERC721AccountStatus.DeleteAccount",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return t.Ctx.ERC721Account.DeleteAccount(accountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```

Account Status SDK Methods

`GetDefaultAccountStatus`
This method gets the current status of a token account, with the status as `active` for any account that does not have account status stored in the ledger (because the account was created prior to the account status functionality).

```
Ctx.ERC721AccountStatus.GetDefaultAccountStatus(accountId string)
(NFTAccountStatus, error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
      "AssetType": "oaccountStatus",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "Status": "active"
}
```

GetAccountStatus
This method gets the current status of the token account.

```
Ctx.ERC721AccountStatus.GetAccountStatus(accountId string) (NFTAccountStatus,
error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

GetAccountStatusHistory
This method gets the history of the account status.

```
Ctx.ERC721AccountStatus.History(statusId string) (interface{}, error)
```

Parameters:

- `statusId: string` – The ID of the account status object.

Returns:

- On success, a JSON representation of the account status history.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

`ActivateAccount`
This method activates a token account. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
Ctx.ERC721Account.ActivateAccount(accountId string) (interface{}, error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```


Copyright © 2025, Oracle and/or its affiliates.

SuspendAccount
This method suspends a token account.

```
Ctx.ERC721Account.SuspendAccount(accountId string) (interface{}, error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

DeleteAccount
This method deletes a token account.

```
Ctx.ERC721Account.DeleteAccount(accountId string) (interface{}, error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```


Copyright © 2025, Oracle and/or its affiliates.

ERC-1155

Blockchain App Builder supports an extended version of the ERC-1155 standard to work with fungible and non-fungible tokens.

- Input Specification File for ERC-1155
- ERC-1155 Tokenization Flow
- Scaffolded TypeScript Token Project for ERC-1155
- Scaffolded Go Token Project for ERC-1155

Input Specification File for ERC-1155

The Blockchain App Builder initialization command reads the input specification file and generates the scaffolded project with several tools to assist in the chaincode development process.

You can define standard assets and both fungible and non-fungible token assets that are based on the ERC-1155 standard in the same specification file. You cannot define token assets based on more than one standard in the same specification file.

For information on including standard assets in the specification file, see Input Specification File.

The following sample specification files for ERC-1155 token assets are available in the Blockchain App Builder package:

- `NFTArtCollectionMarketplacewithERC1155-TypeScript.yml`
- `FractionalNFTinRealEstate-TypeScript.yml`

In addition to the standard properties and sections, token assets support the `behavior` and `anatomy` sections in the specification file. In addition, non-fungible token assets support the `metadata` section. The following example shows the structure of a specification file for two ERC-1155 token assets, a whole non-fungible token and a fractional fungible token:

```
assets:
    - name: ArtCollection #Asset name
      type: token #Asset type
      standard: erc1155+   # Token standard

      anatomy:
          type: nonfungible # Token type
          unit: whole  #Token unit

      behavior:
        - indivisible
        - mintable:
            max_mint_quantity: 20000
        - transferable
        - burnable
        - lockable
        - roles:
            minter_role_name: minter

      properties:  # Custom asset attributes for non-fungible token
```


Copyright © 2025, Oracle and/or its affiliates.

```
          - name: price # Custom asset attribute to set the price of a non-
fungible token in the marketplace
            type: number

          - name: on_sale_flag # Custom asset attribute maintains non-
fungible token selling status in the marketplace
            type: boolean

      metadata: #  Use this section to maintain the metadata on the
blockchain. Only the user creating the non-fungible token can assign metadata
attribute values, which cannot be updated later.

          - name: painting_name
            type: string

          - name: description
            type: string

          - name: image
            type: string

          - name: painter_name
            type: string

    - name: Loyalty # Asset name
      type: token  # Asset type
      standard: erc1155+   # Token standard

      anatomy:
          type: fungible # Token type
          unit: fractional # Token unit

      behavior: # Token behaviors
          - divisible:
                decimal: 2
          - mintable:
                max_mint_quantity: 10000
          - transferable
          - burnable
          - roles:
                minter_role_name: minter

      properties:
          - name: currency_name # Custom attribute to represent the token in
a specific currency.
            type: string

          - name: token_to_currency_ratio # Custom attribute to specify the
token to currency ratio.
            type: number
```


Copyright © 2025, Oracle and/or its affiliates.

The following example shows the structure of a specification file for a fractional non-fungible token:

```
- name: RealEstateProperty #Asset name
      type: token #Asset type
      standard: erc1155+ # Token standard

      anatomy:
          type: nonfungible # Token type
          unit: fractional #Token unit

      behavior:
        - divisible:
        - mintable:
        - transferable
        - roles:
            minter_role_name: minter

      properties: # Custom asset attributes for non-fungible token.

          - name: propertySellingPrice # Custom asset attribute to set the
real estate property selling price
            type: number

          - name: propertyRentingPrice # Custom asset attribute maintains the
renting amount for the real estate property
            type: number

      metadata: # To maintain the metadata on-chain, this tag will be used.
Users won't be able to update the metadata attribute values.

          - name: propertyType
            type: string

          - name: propertyName
            type: string

          - name: propertyAddress
            type: string

          - name: propertyImage
            type: string
```

Table 4-3 Parameter Descriptions and Examples for an ERC-1155 Token Specification File

Entry	Description	Examples

`type:`	You must specify `type: token` in the `assets` section.	`assets:`
		`  - name: ArtCollection #Asset name`
		`  type: token #Asset type`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-3 (Cont.) Parameter Descriptions and Examples for an ERC-1155 Token Specification File

Entry	Description	Examples
`standard:`	The `standard` property is mandatory for ERC-1155 tokens. It represents the token standard to follow during chaincode generation.	`standard: erc1155+ # Token standard`

`anatomy:`	The `anatomy` section has two mandatory parameters:
	• `type: nonfungible` or `type: fungible` A non-fungible token is unique.
	• `unit: whole` or `unit: fractional` A whole token cannot be subdivided into smaller fractional units. A fractional token can be subdivided into smaller units, or shares, based on a specified number of decimal places.
		`anatomy:`
		`  type: nonfungible # Token type`
		`  unit: whole #Token unit`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-3 (Cont.) Parameter Descriptions and Examples for an ERC-1155 Token Specification File

Entry	Description	Examples

`behavior:`	This section describes the capabilities and restrictions of the token. The `mintable`, `transferable` behaviors are mandatory for all tokens. The `indivisible` behavior is mandatory for whole non-fungible tokens.
	• `indivisible`: This behavior supports a restriction so that whole tokens cannot be subdivided into fractions.
	• `divisible`: This behavior describes how tokens can be subdivided. The `decimal` parameter specifies the number of decimal places that can be used. The smallest fraction possible with the number of decimal places is the smallest unit of the token that can be owned. If the `decimal` parameter is not specified, the default is zero decimal places.
	• `mintable`: This mandatory behavior supports minting new token instances. The optional `max_mint_quantity` parameter specifies the total number of tokens that can be minted. If you do not specify the `max_mint_quantity` parameter,
		`behavior:`
		`- indivisible`
		`- mintable:`
		`max_mint_quantity: 20000`
		`- transferable`
		`- burnable`
		`- lockable`
		`- roles:`
		`minter_role_name: minter`

Copyright © 2025, Oracle and/or its affiliates.

Table 4-3 (Cont.) Parameter Descriptions and Examples for an ERC-1155 Token Specification File

Entry	Description	Examples
	any number of tokens can be minted. • `transferable`: This mandatory behavior supports transferring ownership of tokens. • `burnable`: This behavior supports deactivating, or burning, tokens. Burning does not delete a token but instead places it in a permanent state where it cannot be used. Burning is not reversible. • `lockable`: This behavior is optional and is supported only by non-fungible tokens. This behavior allows the token owner to lock a non-fungible token. A locked token cannot be transferred to or burned by any other users. • `roles`: This behavior restricts token behaviors to users with specific roles. Currently two roles are available: `minter_role_name` and `burner_role_name`. If you do not specify roles, then any user can act as a minter or burner. For example, if the burner role is not specified, any account user implicitly has the burner role. If the burner role is	

Copyright © 2025, Oracle and/or its affiliates.

Table 4-3 (Cont.) Parameter Descriptions and Examples for an ERC-1155 Token Specification File

Entry	Description	Examples
	specified, then during the token setup process, the `Token Admin` user must assign the burner role to other users explicitly.	

`metadata:`	The `metadata` property is optional and is supported only by non-fungible tokens. This property specifies metadata information, which is stored on the blockchain, for a non-fungible token. Metadata attribute values can be assigned only by the token owner who mints the token, and cannot be updated after the token is minted.
	In the example, `name` is the name of the metadata attribute and `type` is the type of value that the attribute has.
		`metadata:`
		`- name: painting_name`
		`type: string`
		`- name: description`
		`type: string`
		`- name: image`
		`type: string`
		`- name: painter_name`
		`type: string`

Limitations

Blockchain App Builder provides partial support for the ERC-1155 standard. Currently, the following ERC-1155 events and methods are not supported.

Events:

- `TransferSingle`
- `TransferBatch`
- `ApprovalForAll`
- `URI`

Methods:

- `safeTransferFrom`
- `balanceOf`
- `setApprovalForAll`
- `isApprovedForAll`

Copyright © 2025, Oracle and/or its affiliates.

ERC-1155 Tokenization Flow

After you deploy an ERC-1155 token project, token administrators and token owners follow a typical flow for creating tokens and completing lifecycle operations.

When you deploy a token project, the users in the list passed to the initialization method become token administrators of the chaincode. After deployment, the typical flow for creating tokens and completing lifecycle operations follows these steps:

Token administrator operations:

- Create user accounts for anyone who will possess tokens or complete token-related operations.
- For each user account, create token accounts. Users can have multiple fungible token accounts, but only one non-fungible token (NFT) account. Token administrators can use the `createAccount` method to create user and token accounts simultaneously instead of separately.
- Create fungible tokens, as needed. When you initialize a fungible token, you can assign the associated metadata and behaviors to the token.
- For fungible tokens, associate the fungible token accounts of users to specific fungible tokens.
- Add minter and burner roles to the token accounts of users as needed.

Token owner operations:

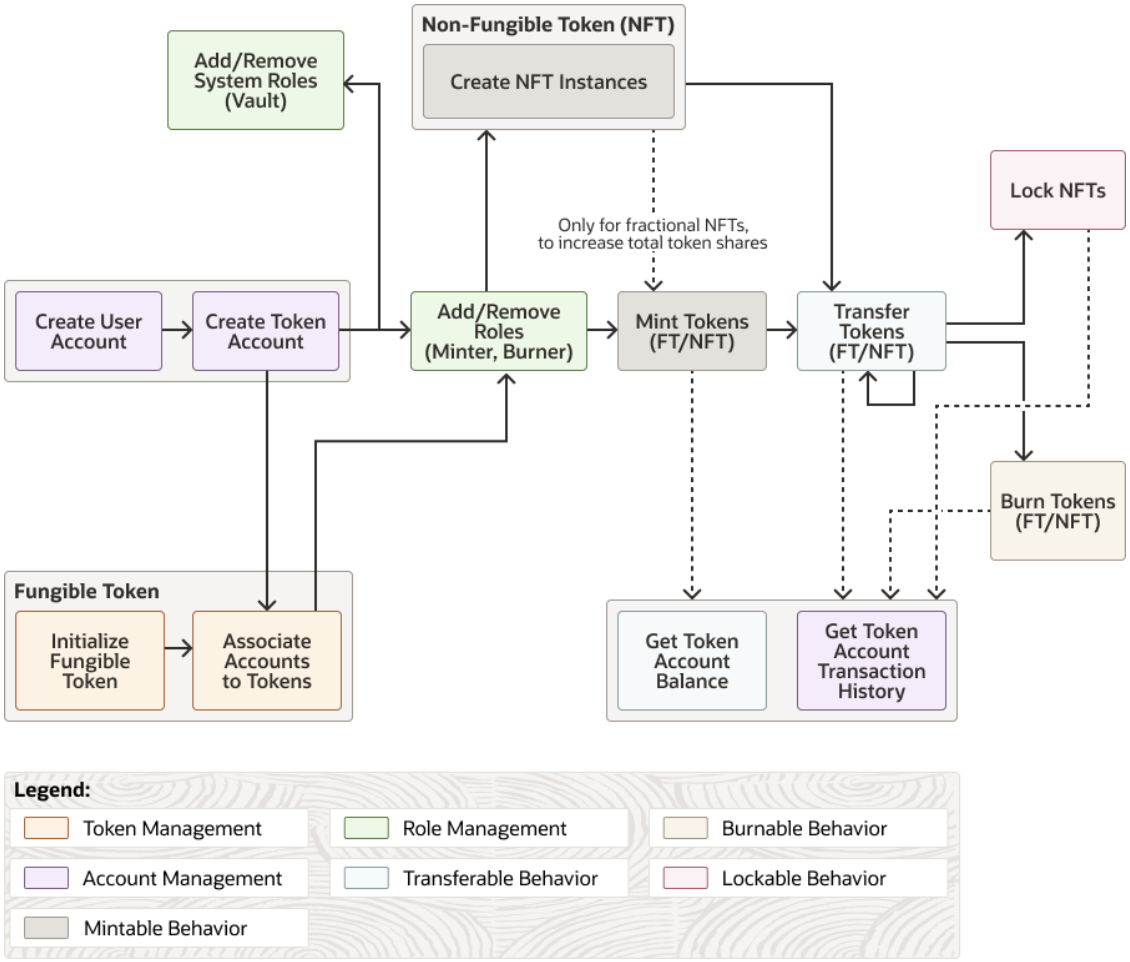
- Users who have the minter role for a specific token can create (mint) NFTs or fungible tokens.
- Users can transfer tokens between accounts, and check account balances.
- Users who have the burner role for a specific token can destroy (burn) NFTs or fungible tokens.

Vault manager operations:

- The user who has the vault role can lock NFTs. A locked NFT cannot be burned or transferred to other users.

The following diagram shows the overall process flow for an ERC-1155 tokenization scenario.

Copyright © 2025, Oracle and/or its affiliates.



The following table summarizes the methods that are automatically generated when you scaffold an ERC-1155 token project.

Method Category	Auto-generated Method (TypeScript)	Auto-generated Method (Go)	Called By	Description
Admin Management	`init`	`Init`	Admin	Initializes the token chaincode
	`isTokenAdmin`	`IsTokenAdmin`	Admin	Returns true if the caller is an Admin
	`addTokenAdmin`	`AddTokenAdmin`	Admin	Adds an Admin
	`removeTokenAdmin`	`RemoveTokenAdmin`	Admin	Removes an Admin
	`getAllTokenAdmins`	`GetAllTokenAdmins`	Admin	Returns all Admins
Account Management	`createAccount`	`CreateAccount`	Admin	Creates a user account and token accounts
	`createUserAccount`	`CreateUserAccount`	Admin	Creates a user account
	`createTokenAccount`	`CreateTokenAccount`	Admin	Creates a token account

Copyright © 2025, Oracle and/or its affiliates.

Method Category	Auto-generated Method (TypeScript)	Auto-generated Method (Go)	Called By	Description
	associateFungibleTokenAccount	AssociateFungibleTokenAccount	Admin	Associates a fungible token account with a fungible token
	getAccountHistory	GetAccountHistory	Admin / Account Owner	Returns history for a token account
	getAccountTransactionHistory	GetAccountTransactionHistory	Admin / Account Owner	Returns transaction history for an account
	getAccount	GetAccount	Admin / Account Owner	Returns details for a token account
	getAllAccounts	GetAllAccounts	Admin	Returns details for all user accounts
	getAccountDetailsByUser	GetAccountDetailsByUser	Admin / Account Owner	Returns details for a user account and all associated tokens
	getUserByAccountId	GetUserByAccountId	Any user	Returns user details for an account ID
Role Management	addRole	AddRole	Admin	Adds a role to a user and token
	isInRole	IsInRole	Admin / Account Owner	Returns whether a user has a specified role for a token
	removeRole	RemoveRole	Admin	Removes a role from a user and token
	getAccountsByRole	GetAccountsByRole	Admin	Returns account IDs for a specified role and token
	getUsersByRole	GetUsersByRole	Admin	Returns a list of users for a specified role and token
Mintable Behavior	mintBatch	MintBatch	Users with the minter role	Mints multiple tokens
Transferable Behavior	batchTransferFrom	BatchTransferFrom	Any user	Transfers tokens between users
	safeBatchTransferFrom	SafeBatchTransferFrom	Any user	Transfers tokens between the method caller and another user
	balanceOfBatch	BalanceOfBatch	Admin / Account Owner	Returns token account balances for multiple users and tokens
	exchangeToken	ExchangeToken	Account Owner	Exchanges tokens between token accounts

Copyright © 2025, Oracle and/or its affiliates.

Method Category	Auto-generated Method (TypeScript)	Auto-generated Method (Go)	Called By	Description
Burnable Behavior	burnBatch	BurnBatch	Users with the burner role	Burns tokens
Token Management	create<TokenName>Token	Create<TokenName>Token	Admin (fungible tokens) / Users with the minter role (NFTs)	Creates tokens
	update<TokenName>Token	Update<TokenName>Token	Admin (fungible tokens) / Token Owner (NFTs)	Updates tokens
	getTokenHistory	GetTokenHistory	Any user	Returns the history of a token
	getTransactionById	GetTransactionById	Any user	Returns the details of a specified transaction
	deleteHistoricalTransactions	DeleteHistoricalTransactions	Admin	Deletes transactions before a specified time
	getAllTokens	GetAllTokens	Admin	Returns all token assets
	getTokenById	GetTokenById	Admin / Token Owner	Returns a token
	getAllTokensByUser	GetAllTokensByUser	Admin / Account Owner	Returns all tokens owned by a specified user
	ownerOf	OwnerOf	Any user	Returns the user details of the owner of a specified token
	URI	URI	Any user	Returns the URI of a specified token
	name	Name	Any user	Returns the name of a specified token
	totalSupply	TotalSupply	Admin	Returns the number of minted tokens for a specified token
	totalNetSupply	TotalNetSupply	Admin	Returns the number of minted tokens minus the number of burned tokens for a specified token
	getTokensByName	getTokensByName	Admin	Returns all token assets for a specified token name
	getTokenDecimal	getTokenDecimal	Admin	Returns the number of decimal places for a specified token

Copyright © 2025, Oracle and/or its affiliates.

Scaffolded TypeScript Token Project for ERC-1155

Blockchain App Builder takes the input from your token specification file and generates a fully-functional scaffolded chaincode project.

The project automatically generates token lifecycle classes and functions, including CRUD and non-CRUD methods. Validation of arguments, marshalling/unmarshalling, and transparent persistence capability are all supported automatically.

For information on the scaffolded project and methods that are not directly related to tokens, see Scaffolded TypeScript Chaincode Project.

Reference:

- Model
- Controller
  - Automatically Generated Token Methods
- SDK Methods

Model

Every tokenized model class extends the `OchainModel` class. Transparent Persistence Capability, or simplified ORM, is captured in the `OchainModel` class. The following model shows a whole non-fungible token.

```
import * as yup from "yup";
import { Id, Mandatory, Validate, Default, Embedded, Derived, ReadOnly } from
"../../lib/decorators";
import { OchainModel } from "../../lib/ochain-model";
import { STRATEGY } from "../../lib/utils";
import { EmbeddedModel } from "../../lib/ochain-embedded-model";

export class ArtCollectionMetadata extends
EmbeddedModel<ArtCollectionMetadata> {
  @Validate(yup.string())
  public painting_name: string;

  @Validate(yup.string())
  public description: string;

  @Validate(yup.string())
  public image: string;

  @Validate(yup.string())
  public painter_name: string;

}

@Id("tokenId")
export class ArtCollection extends OchainModel<ArtCollection> {
  public readonly assetType = "otoken";

  @Mandatory()
  @Validate(
    yup
```


Copyright © 2025, Oracle and/or its affiliates.

```
        .string()
        .required()
        .matches(/^[A-Za-z0-9][A-Za-z0-9_-]*$/)
        .max(16)
    )
    public tokenId: string;

    @ReadOnly("artcollection")
    public tokenName: string;

    @Validate(yup.string().trim().max(256))
    public tokenDesc: string;


    @ReadOnly("erc1155+")
    public tokenStandard: string;

    @ReadOnly("nonfungible")
    public tokenType: string;

    @ReadOnly("whole")
    public tokenUnit: string;


@ReadOnly(["indivisible","singleton","mintable","transferable","burnable","rol
es"])
    public behaviors: string[];

    @ReadOnly({ minter_role_name: "minter" })
    public roles: object;

    @ReadOnly({ max_mint_quantity: 20000 })
    public mintable: object;

    @Validate(yup.string())
    public owner: string;

    @Validate(yup.string())
    public createdBy: string;

    @Validate(yup.string())
    public transferredBy: string;

    @Validate(yup.string())
    public creationDate: string;

    @Validate(yup.string())
    public transferredDate: string;

    @Validate(yup.bool())
    public isBurned: boolean;

    @Validate(yup.string())
    public burnedBy: string;

    @Validate(yup.string())
```


Copyright © 2025, Oracle and/or its affiliates.

```
  public burnedDate: string;

  @Mandatory()
  @Validate(yup.string().required().max(2000))
  public tokenUri: string;

  @Embedded(ArtCollectionMetadata)
  public tokenMetadata: ArtCollectionMetadata;

  @Validate(yup.number())
  public price: number;

  @Validate(yup.boolean())
  public on_sale_flag: boolean;

}

@Id("tokenId")
export class Loyalty extends OchainModel<Loyalty> {
  public readonly assetType = "otoken";

  @Mandatory()
  @Validate(
    yup
      .string()
      .required()
      .matches(/^[A-Za-z0-9][A-Za-z0-9_-]*$/)
      .max(16)
  )
  public tokenId: string;

  @ReadOnly("loyalty")
  public tokenName: string;

  @Validate(yup.string().trim().max(256))
  public tokenDesc: string;

  @ReadOnly("erc1155+")
  public tokenStandard: string;

  @ReadOnly("fungible")
  public tokenType: string;

  @ReadOnly("fractional")
  public tokenUnit: string;

  @ReadOnly(["divisible","mintable","transferable","burnable","roles"])
  public behaviors: string[];

  @ReadOnly({ minter_role_name: "minter" })
  public roles: object;

  @ReadOnly({ max_mint_quantity: 10000 })
  public mintable: object;

  @ReadOnly({ decimal: 2 })
```


Copyright © 2025, Oracle and/or its affiliates.

```
  public divisible: object;

  @Validate(yup.string())
  public currency_name: string;

  @Validate(yup.number())
  public token_to_currency_ratio: number;

}
```

The following model shows a fractional non-fungible token.

```
export class RealEstatePropertyMetadata extends
EmbeddedModel<RealEstatePropertyMetadata> {
  @Validate(yup.string())
  public propertyType: string;

  @Validate(yup.string())
  public propertyName: string;

  @Validate(yup.string())
  public propertyAddress: string;

  @Validate(yup.string())
  public propertyImage: string;

}

@Id("tokenId")
export class RealEstateProperty extends OchainModel<RealEstateProperty> {
  public readonly assetType = "otoken";

  @Mandatory()
  @Validate(
    yup
      .string()
      .required()
      .matches(/^[A-Za-z0-9][A-Za-z0-9_-]*$/)
      .max(16)
  )
  public tokenId: string;

  @ReadOnly("realestateproperty")
  public tokenName: string;

  @Validate(yup.string().trim().max(256))
  public tokenDesc: string;

  @ReadOnly("erc1155+")
  public tokenStandard: string;

  @ReadOnly("nonfungible")
  public tokenType: string;

  @ReadOnly("fractional")
```


Copyright © 2025, Oracle and/or its affiliates.

```
  public tokenUnit: string;

  @ReadOnly(["divisible","mintable","transferable","roles"])
  public behaviors: string[];

  @ReadOnly({ minter_role_name: "minter" })
  public roles: object;

  @ReadOnly({ max_mint_quantity: 0 })
  public mintable: object;

  @Validate(yup.number().positive())
  public quantity: number;

  @Validate(yup.string())
  public createdBy: string;

  @Validate(yup.string())
  public creationDate: string;

  @ReadOnly({ decimal: 0 })
  public divisible: object;

  @Validate(yup.bool())
  public isBurned: boolean;

  @Mandatory()
  @Validate(yup.string().required().max(2000))
  public tokenUri: string;

  @Embedded(RealEstatePropertyMetadata)
  public tokenMetadata: RealEstatePropertyMetadata;

  @Validate(yup.number())
  public propertySellingPrice: number;

  @Validate(yup.number())
  public propertyRentingPrice: number;

}
```

Controller

The main controller class extends the `OchainController` class. There is only one main controller.

```
export class DigiCurrCCController extends OchainController{
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable. The other methods are hidden.

You can use the token SDK methods to write custom methods for your business application.

Copyright © 2025, Oracle and/or its affiliates.

Automatically Generated Token Methods

Blockchain App Builder automatically generates methods to support tokens and token life cycles. You can use these methods to initialize tokens, manage roles and accounts, and complete other token lifecycle tasks without any additional coding. Controller methods must have a `@Validator(...params)` decorator to be invokable.

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

`isTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAccountDetailsByUser(orgId: string, userId: string) {
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.getAccountDetailsByUse
r", "TOKEN", {
    accountId: userAccountId,
  });
  return await this.Ctx.ERC1155Account.getAccountDetailsByUser(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

Return Value Example:

```
{"result": true}
```


Copyright © 2025, Oracle and/or its affiliates.

addTokenAdmin
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
public async addTokenAdmin(orgId: string, userId: string) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ADMIN.addAdmin",
"TOKEN");
  return await this.Ctx.ERC1155Admin.addAdmin(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg":"Successfully added Admin (OrgId: appDev, UserId: user1)"}
```

removeTokenAdmin
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode. You cannot remove yourself as a `Token Admin`.

```
@Validator(yup.string(), yup.string())
public async removeTokenAdmin(orgId: string, userId: string) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ADMIN.removeAdmin",
"TOKEN");
  return await this.Ctx.ERC1155Admin.removeAdmin(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg": "Successfully removed Admin (OrgId: appDev, UserId: user1)"}
```


Copyright © 2025, Oracle and/or its affiliates.

`getAllTokenAdmins`
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async getAllTokenAdmins() {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ADMIN.getAllAdmins",
"TOKEN");
  return await this.Ctx.ERC1155Admin.getAllAdmins();
}
```

Parameters:

- none

Returns:

- On success, an `admins` array in JSON format that contains `orgId` and `userId` objects.

Return Value Example:

```
{
  "admins": [
    {
      "orgId": "appdev",
      "userId": "user2"
    },
    {
      "orgId": "appdev",
      "userId": "user1"
    }
  ]
}
```

Methods for Token Configuration Management

`init`
This method is called when the chaincode is instantiated. Every `Token Admin` is identified by the `userId` and `orgId` information in the `adminList` parameter. The `userId` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `orgId` is the membership service provider (MSP) ID of the user in the current network organization. The `adminList` parameter is mandatory the first time you deploy the chaincode. If you are upgrading the chaincode, pass an empty list (`[]`). If you are the user who initially deployed the chaincode, you can also specify new admins in the `adminList` parameter when you are upgrading the chaincode. Any other information in the `adminList` parameter is ignored during upgrades.

```
@Validator(yup.array().of(yup.object()).nullable())
public async init(adminList: ERC1155TokenAdminAsset[]) {
  await this.Ctx.ERC1155Admin.initAdmin(adminList);
  await this.Ctx.ERC1155Token.saveClassInfo(<1st NFT Token Name>);
  await this.Ctx.ERC1155Token.saveClassInfo(<2nd NFT Token Name>);
```


Copyright © 2025, Oracle and/or its affiliates.

```
  .
  .
  await this.Ctx.ERC1155Token.saveClassInfo(<nth NFT Token Name>);
  // await this.Ctx.ERC1155Token.saveDeleteTransactionInfo();
  return;
}
```

Parameters:

- `adminList array` – An array of `{orgId, userId}` information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

`create<Token Name>Token`
This method creates tokens. Every token that is defined has its own create method. For fungible tokens, this method can be called only by a `Token Admin` of the chaincode. For non-fungible tokens, if the minter role is defined in the specification file, any user with the minter role can call this method to create an NFT. If the minter role is not defined, any user can use this method to create (mint) NFTs. The user who calls this method becomes the owner of the NFT.

Fungible Tokens:

```
@Validator(<Token Class>)
public async create<Token Name>Token(tokenAsset: <Token Class>) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.save", "TOKEN");
  return await this.Ctx.ERC1155Token.save(tokenAsset);
}
```

Non-Fungible Tokens:

```
@Validator(<Token Class>, yup.number())
public async create<Token Name>Token(tokenAsset: <Token Class>, quantity:
number) {
  return await this.Ctx.ERC1155Token.save(tokenAsset, quantity);
}
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset. The properties of the asset are defined in the model file.
- `quantity: number` – For non-fungible tokens only, the number of tokens to mint. The only supported value for this parameter is `1`.

Returns:

- On success, the token asset in JSON format, which includes the following information, depending on the token type.
- `tokenMetadata` – JSON information that describes the token.
- `createdBy` – The account ID of the caller, who is the user minting the token. This property cannot be edited.
- `creationDate` – The time stamp of the minting transaction. This property cannot be edited.

Copyright © 2025, Oracle and/or its affiliates.

- `isBurned` – This property indicates whether the token is burned. This property cannot be edited.
- `tokenName` – The name of the token. This property cannot be edited.
- `tokenDesc` – The description of the token.
- `symbol` – The symbol of the token. This property cannot be edited.
- `tokenStandard` – The standard of the token. This property cannot be edited.
- `tokenType` – The type of the token (fungible or non-fungible). This property cannot be edited.
- `tokenUnit` – The unit of the token (whole or fractional). This property cannot be edited.
- `behaviors` – A list of token behaviors. This property cannot be edited.
- `mintable` – The properties related to minting. The `max_mint_quantity` value defines the maximum number of tokens that can be created for the token class.
- `owner` – The account ID of the current owner, who is the caller of the method.
- `tokenUri` – The URI of the token.
- `quantity` – The quantity of the token.

Return Value Example (Whole NFT):

```
{
    "tokenMetadata": {
        "paintingName": "monalisa",
        "description": "monalisa painting",
        "image": "image link",
        "painterName": "Leonardo da Vinci"
    },
    "assetType": "otoken",
    "quantity": 1,
    "tokenId": "artnft",
    "tokenName": "artcollection",
    "tokenDesc": "artcollection nft",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 500
    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-12-29T04:08:35.000Z",
    "isBurned": false,
    "tokenUri": "tu",
    "price": 10000,
    "onSaleFlag": false
}
```

Return Value Example (Fungible Token):

```
{
    "assetType": "otoken",
    "tokenId": "Loyalty",
    "tokenName": "loyalty",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
    "tokenType": "fungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 10000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "Dollar"
}
```

Return Value Example (Fractional NFT):

```
{
    "tokenMetadata": {
        "painting_name": "paint",
        "description": "Painting Description"
    },
    "assetType": "otoken",
    "tokenId": "realEstate",
    "tokenName": "realestate",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "tokenType": "nonfungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "quantity": 100,
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2023-06-14T04:20:14.000Z",
    "divisible": {
        "decimal": 2
    },
    "isBurned": false,
    "tokenUri": "www.realestate.example.com",
    "price": 1000,
    "on_sale_flag": true
}
```

update<Token Name>Token
This method updates tokens. Every token that is defined has its own update method. You cannot update token metadata or the token URI of non-fungible tokens. For fungible tokens, this method can be called only by a `Token Admin` of the chaincode. For non-fungible tokens, this method can be called only by the token owner.

Fungible Tokens:

```
@Validator(<Token Class>)
public async update<Token Name>Token(tokenAsset: <Token Class>) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.update",
"TOKEN");
  return await this.Ctx.ERC1155Token.update(tokenAsset);
}
```

Non-Fungible Tokens:

```
@Validator(<Token Class>)
public async update<Token Name>Token(tokenAsset: <Token Class>) {
  return await this.Ctx.ERC1155Token.update(tokenAsset);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `tokenAsset: <Token Class>` – The token asset. The properties of the asset are defined in the model file.

Returns:

- On success, the updated token asset in JSON format.

Return Value Example (Whole NFT):


```
{
    "tokenMetadata": {
        "paintingName": "monalisa",
        "description": "monalisa painting",
        "image": "image link",
        "painterName": "Leonardo da Vinci"
    },
    "assetType": "otoken",
    "quantity": 1,
    "tokenId": "artnft",
    "tokenName": "artcollection",
    "tokenDesc": "artcollection nft",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 500
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-12-29T04:08:35.000Z",
    "isBurned": false,
    "tokenUri": "tu",
    "price": 10000,
    "onSaleFlag": false
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getTokenHistory`
This method returns the history for a specified token ID. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async getTokenHistory(tokenId: string) {
  return await this.Ctx.ERC1155Token.getTokenHistory(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON array that contains the token history.

Return Value Example (Fungible Token):

```
[
    {
        "trxId":
"ef4af760c3d7ee5e273196231d59fb91cafe6ca0f78c64747e87bc9bcbb3334b",
        "timeStamp": "2023-09-04T02:36:20.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "LoyaltyToken",
            "tokenName": "loyalty",
            "tokenDesc": "Updated Fungible Whole",
            "tokenStandard": "erc1155+",
            "tokenType": "fungible",
            "tokenUnit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 10000
            },
            "divisible": {
                "decimal": 2
            },
            "currency_name": "Rupees"
        }
    },
    {
        "trxId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"4fb391a8903633a12a545cd2ecfb57f5575241325abf59995e2a4ed96572bb09",
        "timeStamp": "2023-09-04T02:35:07.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "LoyaltyToken",
            "tokenName": "loyalty",
            "tokenDesc": "Fungible Whole",
            "tokenStandard": "erc1155+",
            "tokenType": "fungible",
            "tokenUnit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 10000
            },
            "divisible": {
                "decimal": 2
            },
            "currency_name": "Dollar"
        }
    }
]
```


Return Value Example (Fractional NFT):


```
[
    {
        "txId":
"99bca74f401465206da7499cbf704dd443b3c3d94e348b1d6682ab5ee1864a08",
        "timestamp": "2023-06-20T01:09:18.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "FNFT",
            "tokenName": "realestate",
            "tokenStandard": "erc1155+",
            "tokenType": "nonfungible",
            "tokenUnit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "quantity": 100,
                "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                "creationDate": "2023-06-20T00:53:13.000Z",
                "divisible": {
                    "decimal": 2
                },
                "isBurned": false,
                "tokenUri": "www.FNFT.example.com",
                "price": 2000,
                "on_sale_flag": true,
                "owners": [
                    {
                        "accountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                        "tokenShare": 90
                    },
                    {
                        "accountId":
"oaccount~3cddfdaa855900579d963aa6f755a4aed1f3a474a2462c1b45bd7f36df673224",
                        "tokenShare": 10
                    }
                ]
            }
        },
        {
            "txId":
"d517c61f40e7d6af2f04fe6d337b3e5108eb57030c9dc823793498fd4fed671b",
            "timestamp": "2023-06-20T00:53:13.000Z",
            "value": {
                "assetType": "otoken",
                "tokenId": "FNFT",
                "tokenName": "realestate",
                "tokenStandard": "erc1155+",
                "tokenType": "nonfungible",
                "tokenUnit": "fractional",
                "behaviors": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "mintable": {
                "max_mint_quantity": 20000
            },
            "quantity": 100,
            "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "creationDate": "2023-06-20T00:53:13.000Z",
            "divisible": {
                "decimal": 2
            },
            "isBurned": false,
            "tokenUri": "www.FNFT.example.com",
            "price": 2000,
            "on_sale_flag": true,
            "owners": [
                {
                    "accountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                    "tokenShare": 100
                }
            ]
        }
    }
]
```


Return Value Example (Whole NFT):


```
[
    {
        "trxId":
"92ac6b56112acdba724dd49924d2420a7899c013c61aa40d272e8ab391a65e0f",
        "timeStamp": "2023-09-04T02:28:48.000Z",
        "value": {
            "tokenMetadata": {
                "painting_name": "monalisa",
                "description": "monalisa painting",
                "image": "image link",
                "painter_name": "Leonardo da Vinci"
            },
            "assetType": "otoken",
            "tokenId": "artnft",
            "tokenName": "artcollection",
            "tokenDesc": "Updated Token Description",
            "tokenStandard": "erc1155+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "quantity": 1,
                "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "creationDate": "2023-09-04T02:27:19.000Z",
                "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "isBurned": false,
                "tokenUri": "www.FNFT.example.com",
                "price": 10000,
                "on_sale_flag": true
            }
        },
        {
            "trxId":
"27697dd4a8dba53bad073aa95587cd1ef173b02fd95d771a60273d301fd3bcbe",
            "timeStamp": "2023-09-04T02:27:19.000Z",
            "value": {
                "tokenMetadata": {
                    "painting_name": "monalisa",
                    "description": "monalisa painting",
                    "image": "image link",
                    "painter_name": "Leonardo da Vinci"
                },
                "assetType": "otoken",
                "tokenId": "artnft",
                "tokenName": "artcollection",
                "tokenDesc": "artcollection nft",
                "tokenStandard": "erc1155+",
                "tokenType": "nonfungible",
                "tokenUnit": "whole",
                "behaviors": [
                    "indivisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "quantity": 1,
                "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "creationDate": "2023-09-04T02:27:19.000Z",
            "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "isBurned": false,
            "tokenUri": "www.FNFT.example.com",
            "price": 10000,
            "on_sale_flag": true
        }
    }
]


[
    {
        "trxId":
"ebda1f31543f8906b7ec50a631afff6b8318a3d63c84f3e73be6785cc2ff31ff",
        "timeStamp": "2023-06-20T01:14:08.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "NFT",
            "tokenName": "artcollection",
            "tokenStandard": "erc1155+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 20000
            },
            "quantity": 1,
            "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "creationDate": "2023-06-20T01:14:08.000Z",
            "owner":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "isBurned": false,
            "tokenUri": "www.NFT.example.com",
            "price": 2000,
            "on_sale_flag": true
        }
    }
]
```


`getAllTokens`

This method returns all of the token assets that are saved in the state database. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL

Copyright © 2025, Oracle and/or its affiliates.

rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@GetMethod()
@Validator(yup.string())
public async getTokenHistory(tokenId: string) {
  return await this.Ctx.ERC1155Token.getTokenHistory(tokenId);
}
```

Parameters:

- none

Returns:

- A list of all token assets in JSON format.

Return Value Example:

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
      }
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenTwo",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "tokenType": "fungible",
        "tokenUnit": "fractional",
        "behaviors": [
          "divisible",
          "mintable",
          "transferable",
          "roles"
        ],
        "roles": {
          "minter_role_name": "minter",
          "burner_role_name": "burner"
        },
        "mintable": {
          "max_mint_quantity": 1000
        },
        "divisible": {
          "decimal": 2
        }
      }
    },
    {
      "key": "art",
      "valueJson": {
        "assetType": "otoken",
        "quantity": 1,
        "tokenId": "art",
        "tokenName": "artcollection",
        "tokenStandard": "erc1155+",
        "tokenType": "nonfungible",
        "tokenUnit": "whole",
        "behaviors": [
          "indivisible",
          "singleton",
          "mintable",
          "transferable",
          "burnable",
          "roles"
        ],
        "roles": {
          "minter_role_name": "minter"
        },
        "mintable": {
          "max_mint_quantity": 20000
        },
        "owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
        "createdBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
        "creationDate": "2022-12-08T08:52:57.000Z",
        "isBurned": true,
        "tokenUri": "art.example.com",
        "transferredBy":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
        "transferredDate": "2022-12-08T08:59:17.000Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "burnedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "burnedDate": "2022-12-08T09:01:28.000Z"
    }
  },
  {
    "key": "FNFT",
    "valueJson": {
        "assetType": "otoken",
        "tokenId": "FNFT",
        "tokenName": "realestate",
        "tokenStandard": "erc1155+",
        "tokenType": "nonfungible",
        "tokenUnit": "fractional",
        "behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "roles": {
            "minter_role_name": "minter",
            "burner_role_name": "burner"
        },
        "mintable": {
            "max_mint_quantity": 20000
        },
        "quantity": 100,
        "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
        "creationDate": "2023-06-20T00:53:13.000Z",
        "divisible": {
            "decimal": 2
        },
        "isBurned": false,
        "tokenUri": "www.FNFT.example.com",
        "price": 2000,
        "on_sale_flag": true
    }
  }
]
```

getTokenById

This method returns a token object if the token is present in the state database. For fractional NFTs, the list of owners is also returned. This method can be called only by a `Token Admin` of the chaincode or the token owner.

```
@GetMethod()
@Validator(yup.string())
public async getTokenById(tokenId: string) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getTokenById",
"TOKEN", { tokenId });
```


Copyright © 2025, Oracle and/or its affiliates.

```
  return await this.Ctx.ERC1155Token.getTokenById(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token to get.

Return Value Example (Whole NFT):

```
{
  "assetType": "otoken",
  "quantity": 1,
  "tokenId": "art",
  "tokenName": "artcollection",
  "tokenStandard": "erc1155+",
  "tokenType": "nonfungible",
  "tokenUnit": "whole",
  "behaviors": [
    "indivisible",
    "singleton",
    "mintable",
    "transferable",
    "burnable",
    "roles"
  ],
  "roles": {
    "minter_role_name": "minter"
  },
  "mintable": {
    "max_mint_quantity": 20000
  },
  "owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "createdBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "creationDate": "2022-12-08T08:52:57.000Z",
  "isBurned": true,
  "tokenUri": "art.example.com",
  "transferredBy":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "transferredDate": "2022-12-08T08:59:17.000Z",
  "burnedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "burnedDate": "2022-12-08T09:01:28.000Z"
}
```

Return Value Example (Fungible Token):

```
{
    "assetType": "otoken",
    "tokenId": "Loyalty",
    "tokenName": "loyalty",
    "tokenDesc": "Token Description",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "tokenStandard": "erc1155+",
    "tokenType": "fungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 10000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "Dollar"
}
```

Return Value Example (Fractional NFT):

```
{
    "tokenMetadata": {
        "painting_name": "paint",
        "description": "Painting Description"
    },
    "assetType": "otoken",
    "tokenId": "realEstate",
    "tokenName": "realestate",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "quantity": 100,
    "createdBy":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2023-06-14T04:20:14.000Z",
    "divisible": {
        "decimal": 2
    },
    "isBurned": false,
    "tokenUri": "www.realestate.example.com",
    "price": 1000,
    "on_sale_flag": true,
    "owners": [
        {
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "tokenShare": 100
        }
    ]
}
```

`getAllTokensByUser`
This method returns all of the token assets that are owned by a specified user. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAllTokensByUser(orgId: string, userId: string) {
  const accountId = this.Ctx.ERC1155Account.generateAccountId(orgId, userId,
ACCOUNT_TYPE.USER_ACCOUNT);
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getAllTokensByUser",
"TOKEN", { accountId });
  return await this.Ctx.ERC1155Token.getAllTokensByUser(accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Return Value Example:

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
      }
    }
  },
  {
    "key": "nftToken",
    "valueJson": {
      "assetType": "otoken",
      "quantity": 1,
      "tokenId": "nftToken",
      "tokenName": "artcollection",
      "tokenStandard": "erc1155+",
      "tokenType": "nonfungible",
      "tokenUnit": "whole",
      "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter"
      },
      "mintable": {
        "max_mint_quantity": 20000
      },
      "owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "createdBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "creationDate": "2022-12-08T09:10:21.000Z",
      "isBurned": false,
      "tokenUri": "example.com"
    }
  }
]
```


Copyright © 2025, Oracle and/or its affiliates.

ownerOf
This method returns the account ID, organization ID, and user ID of the owner of the specified token ID. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async ownerOf(tokenId: string) {
  return await this.Ctx.ERC1155Token.ownerOf(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value Example (Whole NFT):

```
{
    "accountId":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
    "orgId": "appdev",
    "userId": "idcqa"
}
```

Return Value Example (Fractional NFT):

```
[
    {
        "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "orgId": "Org1MSP",
        "userId": "admin"
    },
    {
        "accountId":
"oaccount~74108eca702bab6d8548e740254f2cc7955d886885251d52d065042172a59db0",
        "orgId": "Org1MSP",
        "userId": "user"
    }
]
```

URI
This method returns the URI of a specified token. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async URI(tokenId: string) {
  return await this.Ctx.ERC1155Token.tokenURI(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "tokenUri": "example.com"
}
```

`name`
This method returns the name of the token class. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async name(tokenId: string) {
  return await this.Ctx.ERC1155Token.name(tokenId);
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value Example:

```
{"tokenName": "artcollection"}
```

`totalSupply`
This method returns the total number of minted tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.object())
public async totalSupply(tokenDetail: TokenDetail) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.totalSupply",
"TOKEN");
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.totalSupply(token);
}
```

Parameters:

- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"tokenName":"artCollection"}
  ```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{"totalSupply": 110}
```

totalNetSupply
This method returns the total number of minted tokens minus the number of burned tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.object())
public async totalNetSupply(tokenDetail: TokenDetail) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.totalNetSupply",
"TOKEN");
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.totalNetSupply(token);
}
```

Parameters:

- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"tokenName":"artCollection"}
  ```

Return Value Example:

```
{"totalNetSupply": 105}
```

getTokensByName
This method returns all of the token assets for a specified token name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async getTokensByName(tokenName: string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getTokensByName",
"TOKEN");
  return await this.Ctx.ERC1155Token.getTokensByName(tokenName);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `tokenName: string` – The name of the token.

Return Value Example:

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
      }
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenTwo",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "divisible": {
        "decimal": 2
      }
    }
  }
]
```

`getTokenDecimal`
This method returns the number of decimal places for a specified token. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async getTokenDecimal(tokenId: string) {
  const token = await this.Ctx.ERC1155Token.get(tokenId);
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getDecimals",
"TOKEN");
  return {
    msg: `Token Id: ${tokenId} has $
{this.Ctx.ERC1155Token.getDecimals(token)} decimal places.`,
  };
}
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value Example:

```
{
    "msg": "Token Id: tokenOne has 2 decimal places."
}
```

Methods for Account Management

`createAccount`
This method creates an account for a specified user and associated token accounts for fungible or non-fungible tokens. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a user holds. Users must have accounts in the network to complete token-related operations. This method can be called only by a `Token Admin` of the chaincode.
A user account has a unique ID, which is formed by an SHA-256 hash of the `orgId` parameter and the `userId` parameter.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this account.

Copyright © 2025, Oracle and/or its affiliates.

User account IDs start with with `ouaccount~`. Token account IDs start with `oaccount~`.

```
@Validator(yup.string(), yup.string(), yup.boolean(), yup.boolean())
public async createAccount(orgId: string, userId: string, ftAccount: boolean,
nftAccount: boolean) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.createAccount",
"TOKEN");
  return await this.Ctx.ERC1155Account.createAccount(orgId, userId,
ftAccount, nftAccount);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `ftAccount: boolean` – If true, a fungible token account is created and associated with the user account.
- `nftAccount: boolean` – If true, a non-fungible token account is created and associated with the user account.

Returns:

- On success, a JSON object of the account that was created.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~cf20877546f52687f387e7c91d88b9722c97e1a456cc0484f40c747f7804feae",
  "userId": "user1",
  "orgId": "appdev",
  "totalAccounts": 2,
  "totalFtAccounts": 1,
  "associatedFtAccounts": [
    {
      "accountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
      "tokenId": ""
    }
  ],
  "associatedNftAccount":
"oaccount~73c3e835dac6d0a56ca9d8def08269f83cefd59b9d297fe2cdc5a9083828fa58"
}
```

`createUserAccount`
This method creates an account for a specified user. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a user has. Users must have accounts in the network to complete token-related operations.

Copyright © 2025, Oracle and/or its affiliates.

An account ID is an SHA-256 hash of the `orgId` parameter and the `userId` parameter. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
  public async createUserAccount(orgId: string, userId: string) {
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.createUserAccount",
"TOKEN");
    return await this.Ctx.ERC1155Account.createUserAccount(orgId, userId);
  }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object of the user account that was created.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "userId": "user2",
  "orgId": "appdev",
  "totalAccounts": 0,
  "totalFtAccounts": 0,
  "associatedFtAccounts": [],
  "associatedNftAccount": ""
}
```

`createTokenAccount`
This method creates a fungible or non-fungible token account to associate with a user account.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this account.
This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async createTokenAccount(orgId: string, userId: string, tokenType:
TokenType) {
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.createTokenAccount",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"TOKEN");
    return await this.Ctx.ERC1155Account.createTokenAccount(orgId, userId,
tokenType);
  }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenType: TokenType` – The type of token account to create. The only supported token types are `nonfungible` and `fungible`.

Returns:

- On success, a JSON object of the token account that was created.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "userId": "user2",
  "orgId": "appdev",
  "totalAccounts": 1,
  "totalFtAccounts": 1,
  "associatedFtAccounts": [
    {
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "tokenId": ""
    }
  ],
  "associatedNftAccount": ""
}
```

`associateFungibleTokenAccount`
This method associates a user's fungible token account to a particular fungible token. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async associateFungibleTokenToAccount(orgId: string, userId: string,
tokenId: string) {
  const accountId = this.Ctx.ERC1155Account.generateAccountId(orgId, userId,
ACCOUNT_TYPE.USER_ACCOUNT);
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.associateFungibleToken
ToAccount", "TOKEN", { accountId });
  return await this.Ctx.ERC1155Account.associateTokenToToken(accountId,
tokenId);
}
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object of the user account, which shows that the fungible token was associated to the token account. For example, in the following example, the first object in the `associatedFtAccounts` array shows that the fungible token account ID and the token ID are associated.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "userId": "user2",
  "orgId": "appdev",
  "totalAccounts": 1,
  "totalFtAccounts": 1,
  "associatedFtAccounts": [
    {
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "tokenId": "tokenOne"
    }
  ],
  "associatedNftAccount": ""
}
```

`getAccountHistory`
This method returns history for a specified token account. This is an asynchronous method. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async getAccountHistory(orgId: string, userId: string, tokenId?:
string) {
  const userAccountId = await
this.Ctx.ERC1155Account.generateAccountId(orgId, userId,
ACCOUNT_TYPE.USER_ACCOUNT);
  const userAccount = await this.Ctx.ERC1155Account.getAccount(userAccountId,
tokenId);
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.history",
"TOKEN", { accountId: userAccountId });
  return await
this.Ctx.ERC1155Account.getAccountHistory(userAccount.accountId);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, an array of JSON objects that describes the account history.

Return Value Example:


```
[
  {
    "trxId":
"89f462697f3c988024b2c248cbda21f9eb7e96567e56dd8db64ada96a4845a7f",
    "timeStamp": "2022-12-08T07:15:10.000Z",
    "value": {
      "assetType": "oaccount",
      "bapAccountVersion": 1,
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "userId": "user2",
      "orgId": "appdev",
      "tokenType": "fungible",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "balance": 110
    }
  },
  {
    "trxId":
"30dd4fe0746350c85a5000996974487010a0a8fee73d6b2e480c3ca330a6d31f",
    "timeStamp": "2022-12-08T06:43:10.000Z",
    "value": {
      "assetType": "oaccount",
      "bapAccountVersion": 0,
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "userId": "user2",
      "orgId": "appdev",
      "tokenType": "fungible",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "balance": 0
    }
  },
  {
    "trxId":
"6226c0455cc3a4f99c3fd7ed8b1d36b8e93f863e42ab61a9b0d399f2d69d2f3d",
    "timeStamp": "2022-12-08T06:41:51.000Z",
    "value": {
      "assetType": "oaccount",
      "bapAccountVersion": 0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "userId": "user2",
      "orgId": "appdev",
      "tokenType": "fungible",
      "tokenId": "",
      "balance": 0
    }
  }
]
```

`getAccount`
This method returns token account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async getAccount(orgId: string, userId: string, tokenId?: string) {
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.getAccount",
"TOKEN", { accountId: userAccountId });
  return await this.Ctx.ERC1155Account.getAccountWithStatus(userAccountId,
tokenId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON object that includes token account details. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example (Non-Fungible Token Account):

```
{
  "assetType": "oaccount",
  "bapAccountVersion": 1,
  "status": "active",
  "accountId":
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
  "userId": "idcqa",
  "orgId": "appdev",
  "tokenType": "nonfungible",
  "noOfNfts": 1
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fungible Token Account):

```
{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "accountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "userId": "idcqa",
  "orgId": "appdev",
  "tokenType": "fungible",
  "tokenId": "t1",
  "tokenName": "loyalty",
  "balance": 100
}
```

`getAllAccounts`
This method returns details of all user accounts. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async getAllAccounts() {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.getAllAccounts",
"TOKEN");
  return await this.Ctx.ERC1155Account.getAllAccounts();
}
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

Return Value Example:

```
[
        {
            "assetType": "ouaccount",
            "accountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
            "userId": "idcqa",
            "orgId": "appdev",
            "totalAccounts": 2,
            "totalFtAccounts": 1,
            "associatedFtAccounts": [
                {
                    "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
                    "tokenId": "loy1"
                }
```


Copyright © 2025, Oracle and/or its affiliates.

```
            ],
            "associatedNftAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371"
        },
        {
            "assetType": "ouaccount",
            "accountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
            "userId": "user1_minter",
            "orgId": "appdev",
            "totalAccounts": 2,
            "totalFtAccounts": 1,
            "associatedFtAccounts": [
                {
                    "accountId":
"oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c",
                    "tokenId": "loy1"
                }
            ],
            "associatedNftAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446"
        },
    ]
```

getAccountDetailsByUser
This method returns an account summary for a specified user and details of fungible and non-fungible tokens that are associated with the user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getAccountDetailsByUser(orgId: string, userId: string) {
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.getAccountDetailsByUse
r", "TOKEN", {
    accountId: userAccountId,
  });
  return await this.Ctx.ERC1155Account.getAccountDetailsByUser(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes and account summary for the specified user and details of fungible and non-fungible tokens that are associated with the user. For fractional non-fungible tokens, the `tokenShare` property in the `associatedNFTs` section shows the share that the user owns.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "userAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
    "associatedFTAccounts": [
        {
            "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "tokenId": "FT",
            "balance": 50
        }
    ],
    "associatedNFTAccount": {
        "accountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "associatedNFTs": [
            {
                "nftTokenId": "FNFT",
                "tokenShare": 100
            },
            {
                "nftTokenId": "FNFT2",
                "tokenShare": 110
            },
            {
                "nftTokenId": "NFT"
            }
        ]
    }
}
```

`getUserByAccountId`
This method returns the user details of a specified account ID. This method can be called by any user.

```
@GetMethod()
@Validator(yup.string())
public async getUserByAccountId(accountId: string) {
  return await this.Ctx.ERC1155Account.getUserByAccountId(accountId);
}
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`orgId` and `userId`).

Return Value Example:

```
{
  "orgId": "appdev"
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "userId": "user2",
}
```

Methods for Role Management

`addRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. The specified user must have a token account that is associated with the fungible token, or a non-fungible token account for NFT roles. The specified role must exist in the specification file for the token.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.object())
public async addRole(orgId: string, userId: string, role: string,
tokenDetail: TokenDetail) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.addRoleMember",
"TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.addRoleMember(role, userAccountId,
token);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the role to add to the specified user.
- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"tokenName":"artCollection"}
  ```

Returns:

- On success, a message with account details.

Return Value Example:

```
{
  "msg": "Successfully added role 'minter' to Account Id:
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(Org-Id: appdev, User-Id: idcqa)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

isInRole
This method returns a Boolean value to indicate if a user has a specified role. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account. The specified user must have a token account that is associated with the fungible token, or a non-fungible token account for NFT roles. The specified role must exist in the specification file for the token.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string(), yup.object())
public async isInRole(orgId: string, userId: string, role: string,
tokenDetail: TokenDetail) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.removeRoleMember",
"TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.isInRole(role, userAccountId, token);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the role to search for.
- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"tokenName":"artCollection"}
  ```

Return Value Example:

```
{
    "result": true,
    "msg": "Account Id
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(Org-Id: appdev, User-Id: idcqa) has minter role"
}
```

removeRole
This method removes a role from a specified user and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode. The specified user must have a token account that is

Copyright © 2025, Oracle and/or its affiliates.

associated with the fungible token, or a non-fungible token account for NFT roles. The specified role must exist in the specification file for the token.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.object())
public async removeRole(orgId: string, userId: string, role: string,
tokenDetail: TokenDetail) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.removeRoleMember",
"TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.removeRoleMember(role, userAccountId,
token);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the role to remove from the specified user.
- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"tokenName":"artCollection"}
  ```

Return Value Example:

```
{
  "msg": "Successfully removed role 'minter' from Account Id:
oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b
(Org-Id: appdev, User-Id: user1)"
}
```

`getAccountsByRole`
This method returns a list of all account IDs for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string(), yup.object())
public async getAccountsByRole(role: string, tokenDetail: TokenDetail) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ROLE.getAccountsByRole",
"TOKEN");
```


Copyright © 2025, Oracle and/or its affiliates.

```
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.getAccountsByRole(role, token);
}
```

Parameters:

- `role: string` – The name of the role to search for.
- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"tokenName":"artCollection"}
  ```

Return Value Example:

```
{
  "accounts": [

"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",

"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b"
  ]
}
```

`getUsersByRole`
This method returns a list of all users for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string(), yup.object())
public async getUsersByRole(role: string, tokenDetail: TokenDetail) {
  await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ROLE.getUsersByRole",
"TOKEN");
  const token = await this.Ctx.ERC1155Token.getTokenByIdOrName(tokenDetail);
  return await this.Ctx.ERC1155Token.getUsersByRole(role, token);
}
```

Parameters:

- `role: string` – The name of the role to search for.
- `tokenDetails: TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"tokenId":"token1"}
  ```


Copyright © 2025, Oracle and/or its affiliates.

For non-fungible tokens, use the following format:

```
{"tokenName":"artCollection"}
```

Return Value Example:

```
{
    "users": [
        {
            "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "orgId": "appdev",
            "userId": "user2"
        },
        {
            "accountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
            "orgId": "appdev",
            "userId": "user1"
        }
    ]
}
```

Methods for Transaction History Management

`getAccountTransactionHistory`
This method returns account transaction history. This method can be called only by a `Token Admin` of the chaincode or by the account owner. For non-fungible tokens, this method can only be called when connected to the remote Oracle Blockchain Platform network.

```
@GetMethod()
@Validator(yup.string(), yup.string(), yup.string())
public async getAccountTransactionHistory(orgId: string, userId: string,
tokenId?: string) {
  const userAccountId = await
this.Ctx.ERC1155Account.generateAccountId(orgId, userId,
ACCOUNT_TYPE.USER_ACCOUNT);
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.getAccountTransactionH
istory", "TOKEN", {
    accountId: userAccountId,
  });
  const account = await this.Ctx.ERC1155Account.getAccount(userAccountId,
tokenId);
  return await
this.Ctx.ERC1155Account.getAccountTransactionHistory(account.accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `userId: string` – The user name or email ID of the user.
- `tokenId?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Return Value Example:


```
[
    {
        "transactionId":
"otransaction~3a6b23c3003626f3947e990eddbd7ac23398d2200e2eb3eac745e6ddfae140bc
~7c88c736df38d5622512f1e8dcdd50710eb47c953f1ecb24ac44790a9e2f475b",
        "timestamp": "2023-06-06T14:48:08.000Z",
        "tokenId": "FNFT",
        "transactedAmount": 10,
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "transactionType": "DEBIT",
        "balance": 90
    },
    {
        "transactionId":
"otransaction~3a6b23c3003626f3947e990eddbd7ac23398d2200e2eb3eac745e6ddfae140bc
~178e3730bc5bee50d02f1464a4eebf733a051905f651e5789039adb4a3edc114",
        "timestamp": "2023-06-06T14:48:08.000Z",
        "tokenId": "NFT",
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "transactionType": "DEBIT"
    },
    {
        "transactionId":
"otransaction~c369929e28e78de06c72d020f1418c9a154a7dd280b2e22ebb4ea4485e249124
~a7cefb22ff39ee7e36967be71de27da6798548c872061a62dabc56d88d50b930",
        "timestamp": "2023-06-06T14:47:08.000Z",
        "tokenId": "NFT",
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "transactionType": "MINT"
    },
    {
        "transactionId":
"otransaction~114a1bc78d04be48ee6dc140c32c042ee9481cb118959626f090eec744522422
~e4eb15d9354f694230df8835ade012100d82aa43672896a2c7125a86e3048f9f",
        "timestamp": "2023-06-05T17:17:57.000Z",
        "tokenId": "FNFT",
        "transactedAmount": 100,
        "triggeredByUserAccountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "transactionType": "MINT",
        "balance": 100
    }
]
```

getTransactionById
This method returns the transaction details for a specified transaction ID. Anyone can call this method.

```
@GetMethod()
@Validator(yup.string())
public async getTransactionById(transactionId: string) {
  return await this.Ctx.ERC1155Transaction.getTransactionById(transactionId);
}
```

Parameters:

- `transactionId: string` – The ID of the transaction.

Return Value Example:

```
{
  "transactionId":
"otransaction~9ea7b05ab099f7ff4db8342b8c3609031f1d54f11205906e7f1fe88661fe3cbe
~33b59ce0c89e96c1e16449f24301581e8e71954f38ad977c7eb6f065e87f2a53",
  "history": [
    {
      "trxId":
"9ea7b05ab099f7ff4db8342b8c3609031f1d54f11205906e7f1fe88661fe3cbe",
      "timeStamp": "2022-12-08T09:01:28.000Z",
      "value": {
        "assetType": "otransaction",
        "transactionId":
"otransaction~9ea7b05ab099f7ff4db8342b8c3609031f1d54f11205906e7f1fe88661fe3cbe
~33b59ce0c89e96c1e16449f24301581e8e71954f38ad977c7eb6f065e87f2a53",
        "tokenId": "tokenOne",
        "fromAccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
        "toAccountId": "",
        "transactionType": "BURN",
        "amount": 5,
        "timestamp": "2022-12-08T09:01:28.000Z",
        "triggeredByUserAccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc"
      }
    }
  ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`deleteHistoricalTransactions`
This method deletes transactions before a specified time stamp from the state database. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.date())
public async deleteHistoricalTransactions(time_to_expiration: Date) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("TRANSACTION.deleteTransactions",
"TOKEN");
  return await
this.Ctx.ERC1155Transaction.deleteTransactions(time_to_expiration);
}
```

Parameters:

- `timestamp: string` – All transactions before this time stamp will be deleted.

Return Value Example:

```
{
  "msg": "Successfuly deleted transaction older than date: Thu Apr 07 2022
21:18:59 GMT+0000 (Coordinated Universal Time).",
  "transactions": [

"otransaction~30513757d8b647fffaafac440d743635f5c1b2e41b25ebd6b70b5bbf78a2643f
",

"otransaction~ac0e908c735297941ba58bb208ee61ff4816a1e54c090d68024f82adf743892b
"
  ]
}
```

Methods for Token Behavior Management - Mintable Behavior

`mintBatch`
This method creates (mints) multiple tokens in a batch operation. This method creates only fungible tokens or fractional non-fungible tokens.
For fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. You cannot mint more than the `max_mint_quantity` property of the token, if that property was specified when the token was created or updated.
For non-fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. Additionally, the caller must also be the creator of the token. There is no upper limit to the quantity of fractional non-fungible tokens that can be minted.
You cannot use this method to mint a whole non-fungible token.

```
@Validator(yup.string(), yup.string(), yup.array().of(yup.string()),
yup.array().of(yup.number()))
public async mintBatch(orgId: string, userId: string, tokenIds: string[],
quantity: number[]) {
  const accountId = this.Ctx.ERC1155Account.generateAccountId(orgId, userId,
ACCOUNT_TYPE.USER_ACCOUNT);
```


Copyright © 2025, Oracle and/or its affiliates.

```
  return await this.Ctx.ERC1155Token.mintBatch(accountId, tokenIds, quantity);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenIds: string[]` – The list of token IDs to mint tokens for.
- `quantity: number[]` – The list of quantities of tokens to mint, corresponding to the token ID array.

Returns:

- On success, a JSON object that includes details on the minted tokens.

Return Value Example:

```
{
    "msg": "Successfully minted batch of tokens for User-Account-Id
ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38
(Org-Id: appdev, User-Id: idcqa).",
    "details": [
        {
            "msg": "Successfully minted 100 tokens of fractional tokenId:
plot55 to Org-Id: appdev, User-Id: idcqa"
        },
        {
            "msg": "Successfully minted 100 tokens of tokenId: loyalty to
Token-Account-Id
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e"
        }
    ]
}
```

Methods for Token Behavior Management - Transferable Behavior

`batchTransferFrom`
This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.
For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.
For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.
This method does not validate that the caller of the method is the specified sender. This method can be called by any user.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.array().of(yup.string()), yup.array().of(yup.number()))
public async batchTransferFrom(
  fromOrgId: string,
```


Copyright © 2025, Oracle and/or its affiliates.

```
  fromUserId: string,
  toOrgId: string,
  toUserId: string,
  tokenIds: string[],
  quantity: number[]
) {
  const fromAccountId = this.Ctx.ERC1155Account.generateAccountId(fromOrgId,
fromUserId, ACCOUNT_TYPE.USER_ACCOUNT);
  const toAccountId = this.Ctx.ERC1155Account.generateAccountId(toOrgId,
toUserId, ACCOUNT_TYPE.USER_ACCOUNT);
  return await this.Ctx.ERC1155Token.batchTransferFrom(fromAccountId,
toAccountId, tokenIds, quantity);
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the sender and token owner in the current organization.
- `fromUserId: string` – The user name or email ID of the sender and token owner.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.
- `tokenIds: string[]` – A list of token IDs for the tokens to transfer.
- `quantity: number[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

- On success, a message with details for each token transfer.

Return Value Example:

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
```


Copyright © 2025, Oracle and/or its affiliates.

```
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

safeBatchTransferFrom

This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.

For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.

For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.

The caller of the method must be the specified sender. This method can be called by any user.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.array().of(yup.string()), yup.array().of(yup.number()))
public async safeBatchTransferFrom(
  fromOrgId: string,
  fromUserId: string,
  toOrgId: string,
  toUserId: string,
  tokenIds: string[],
  quantity: number[]
) {
  const fromAccountId = this.Ctx.ERC1155Account.generateAccountId(fromOrgId,
fromUserId, ACCOUNT_TYPE.USER_ACCOUNT);
  const toAccountId = this.Ctx.ERC1155Account.generateAccountId(toOrgId,
toUserId, ACCOUNT_TYPE.USER_ACCOUNT);
  return await this.Ctx.ERC1155Token.safeBatchTransferFrom(fromAccountId,
toAccountId, tokenIds, quantity);
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the sender and token owner in the current organization.
- `fromUserId: string` – The user name or email ID of the sender and token owner.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.
- `tokenIds: string[]` – A list of token IDs for the tokens to transfer.
- `quantity: number[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

- On success, a message with details for each token transfer.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`balanceOfBatch`
This method completes a batch operation that gets the balance of token accounts. The account details are specified in three separate lists of organization IDs, user IDs, and token IDs. This method can be called only by a `Token Admin` of the chaincode or by account owners. Account owners can see balance details only for accounts that they own.

```
@GetMethod()
@Validator(yup.array().of(yup.string()), yup.array().of(yup.string()),
yup.array().of(yup.string()))
public async balanceOfBatch(orgIds: string[], userIds: string[], tokenIds:
string[]) {
  let callerAccountCheck = false;
  try {
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.balanceOfBatch",
"TOKEN");
  } catch (err) {
    callerAccountCheck = true;
  }
  const accountIds = await this.Ctx.ERC1155Account.generateAccountIds(orgIds,
userIds, callerAccountCheck);
  return await this.Ctx.ERC1155Account.balanceOfBatch(accountIds, tokenIds);
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgIds: string[]` – A list of the membership service provider (MSP) IDs in the current organization.
- `userIds: string[]` – A list of the user name or email IDs.
- `tokenIds: string[]` – A list of the token IDs.

Return Value Example:
In the following example, the token ID `FNFT` represents a fractional non-fungible token and the token ID `FT` represents a fungible token.

```
[
    {
        "orgId": "appdev",
        "userId": "idcqa",
        "userAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "tokenAccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "tokenId": "FNFT",
        "balance": 100
    },
    {
        "orgId": "appdev",
        "userId": "idcqa",
        "userAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "tokenAccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
        "tokenId": "FT",
        "balance": 50
    },
    {
        "orgId": "appdev",
        "userId": "user1_minter",
        "userAccountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
        "tokenAccountId":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "tokenId": "FNFT",
        "balance": 10
    }
]
```

`exchangeToken`
This method exchanges tokens between specified accounts. This method only supports exchanging between an NFT and a fungible token or a fungible token and an NFT. The NFT can be whole or fractional. This method can be called only by the account owner.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.number(),
yup.string(), yup.string(), yup.string(), yup.number())
  public async exchangeToken(
    fromTokenId: string,
    fromOrgId: string,
```


Copyright © 2025, Oracle and/or its affiliates.

```
    fromUserId: string,
    fromTokenQuantity: number,
    toTokenId: string,
    toOrgId: string,
    toUserId: string,
    toTokenQuantity: number
  ) {
    const fromUserAccountId =
this.Ctx.ERC1155Account.generateAccountId(fromOrgId, fromUserId,
ACCOUNT_TYPE.USER_ACCOUNT);
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT.exchangeToken",
"TOKEN", { accountId: fromUserAccountId });
    const toUserAccountId =
this.Ctx.ERC1155Account.generateAccountId(toOrgId, toUserId,
ACCOUNT_TYPE.USER_ACCOUNT);
    return await this.Ctx.ERC1155Token.exchangeToken(
      fromTokenId,
      fromUserAccountId,
      fromTokenQuantity,
      toTokenId,
      toUserAccountId,
      toTokenQuantity
    );
  }
```

Parameters:

- `fromTokenId: string` – The ID of the token that the sender owns.
- `fromOrgId: string` – The membership service provider (MSP) ID of the sender in the current organization.
- `fromUserId: string` – The user name or email ID of the sender.
- `fromTokenQuantity: number` – The quantity of tokens from the sender to exchange with the receiver.
- `toTokenId: string` – The ID of the token that the receiver owns.
- `toOrgId: string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId: string` – The user name or email ID of the receiver.
- `toTokenQuantity: number` – The quantity of tokens from the receiver to exchange with the sender.

Returns:

- On success, a message with token exchange details.

Return Value Example:

```
{
    "msg": "Succesfully exchanged 10 tokens of type nonfungible with tokenId:
[r1] from Account
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
```


Copyright © 2025, Oracle and/or its affiliates.

```
(OrgId: appdev, UserId: idcqa) to 10 tokens of type fungible with tokenId:
[loy1] from Account
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(OrgId: appdev, UserId: user1_minter)"
}
```

Methods for Token Behavior Management - Burnable Behavior

`burnBatch`
This method deactivates, or burns, the specified fungible and non-fungible tokens. Any user with the burner role can call this method.

```
@Validator(yup.string(), yup.string(), yup.array().of(yup.string()),
yup.array().of(yup.number()))
public async burnBatch(orgId: string, userId: string, tokenIds: string[],
quantity: number[]) {
    const accountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
    return await this.Ctx.ERC1155Token.burn(accountId, tokenIds, quantity);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID in the current organization.
- `userId: string` – The user name or email ID.
- `tokenIds: string[]` – The list of the token IDs to burn
- `quantity: number[]` – The list of quantities of tokens to burn, corresponding to the token ID array..

Returns:

- On success, a message with details about the burn operations.

Return Value Example:

```
[
  {
    "msg": "Successfully burned NFT token: 'art' from Account-Id:
oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6
(Org-Id: appdev, User-Id: idcqa)"
  },
  {
    "msg": "Successfully burned 5 tokens of tokenId: tokenOne from Account-ID
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(Org-Id: appdev, User-Id: idcqa)"
  },
  {
    "msg": "Successfully burned 2 token share of tokenId: FNFT from Account-
ID oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a
(Org-Id: AutoF1377358917, User-Id: idcqa)"
  }
]
```


Copyright © 2025, Oracle and/or its affiliates.

burnNFT
This method deactivates, or burns, the specified non-fungible token, and returns a token object and token history. Any user with the burner role can call this method.

```
@Validator(yup.string(), yup.string(), yup.string())
public async burnNFT(orgId: string, userId: string, tokenId: string):
Promise<any> {
    const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId);
    const token = await this.Ctx.ERC1155Token.get(tokenId);
    if (token.tokenType !== TOKEN_TYPE.NON_FUNGIBLE) {

      throw new Error(`The Token with id ${tokenId} is not a nonfungible
token`);
    }
    if (token.isBurned === true) {
      throw new Error(`token with tokenId ${tokenId} is already burned`);
    }
    let tokenBurnQuantity = 1;
    const tokenUnit = token.tokenUnit;
    if (tokenUnit === TOKEN_UNIT.FRACTIONAL) {
       const owners = await
this.Ctx.ERC1155Token.getFractionalNFTOwners(tokenId);
       if (owners.length !== 1) {
         throw new Error(`Token with tokenId ${tokenId} has multiple owners`);
       }
      tokenBurnQuantity = token.quantity;
    }
    const tokenHistory = await this.Ctx.ERC1155Token.getTokenHistory(tokenId);
    await this.Ctx.ERC1155Token.burn(userAccountId, [tokenId],
[tokenBurnQuantity]);
    token.tokenId = parseInt(token.tokenId);
    if(Number.isNaN(token.tokenId)) {
      throw new Error(`tokenId is expected to be integer but found $
{tokenId}`)
    }
    token.isBurned = true;
    return {...token, tokenHistory: JSON.stringify(tokenHistory)};
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID in the current organization.
- `userId: string` – The user name or email ID.
- `tokenId: string` – The ID of the non-fungible token to burn

Returns:

- On success, a token object in JSON format that includes token history information.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
{
    "assetType": "otoken",
    "tokenId": 1,
    "tokenName": "artcollection",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "quantity": 1,
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2023-08-22T07:32:40.000Z",
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "isBurned": true,
    "tokenUri": "example.com",
    "price": 120,
    "onSaleFlag": false,
    "tokenHistory":
"[{\"trxId\":\"21a932750f2d4ccffd62eda5678a577cadde0513ed7c7a307f24cd7713a1818
b\",\"timeStamp\":\"2023-08-22T07:32:40.000Z\",\"value\":
{\"assetType\":\"otoken\",\"tokenId\":\"1\",\"tokenName\":\"artcollection\",\"
tokenStandard\":\"erc1155+
\",\"tokenType\":\"nonfungible\",\"tokenUnit\":\"whole\",\"behaviors\":
[\"indivisible\",\"singleton\",\"mintable\",\"transferable\",\"burnable\",\"ro
les\"],\"roles\":{\"minter_role_name\":\"minter\"},\"mintable\":
{\"max_mint_quantity\":20000},\"quantity\":1,\"createdBy\":\"oaccount~42e89f4c
72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d\",\"creationDate\":\"
2023-08-22T07:32:40.000Z\",\"owner\":\"oaccount~42e89f4c72dfde9502814876423c6d
a630d466e87436dd1aae201d347ad1288d\",\"isBurned\":false,\"tokenUri\":\"example
.com\",\"price\":120,\"onSaleFlag\":false}}]"
}
```


SDK Methods

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management

Copyright © 2025, Oracle and/or its affiliates.

- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

`isUserTokenAdmin`
This method returns the Boolean value `true` if the specified user is a `Token Admin`, and `false` otherwise. The method can be called only by a `Token Admin` of the token chaincode.

```
Ctx.ERC1155Auth.isUserTokenAdmin(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Return Value Example:

```
{
  "result": true
}
```

`addAdmin`
This method adds a user as a `Token Admin` of the token chaincode. The method can be called only by a `Token Admin` of the token chaincode.

```
Ctx.ERC1155Admin.addAdmin(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user added as a `Token Admin` of the token chaincode.

Return Value Example:

```
{
  "msg": "Successfully added Admin (OrgId: appDev, UserId: user1)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

removeAdmin
This method removes a user as a `Token Admin` of the token chaincode. The method can be called only by a `Token Admin` of the token chaincode. You cannot remove yourself as a `Token Admin`.

```
Ctx.ERC1155Admin.removeAdmin(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user removed as a `Token Admin` of the token chaincode.

Return Value Example:

```
{
  "msg": "Successfully removed Admin (OrgId: appDev, UserId: user1)"
}
```

getAllAdmins
This method returns a list of all `Token Admin` users.

```
Ctx.ERC1155Admin.getAllAdmins()
```

Parameters:

- none

Returns:

- On success, a list of all `Token Admin` users, identifed by organization ID and user ID.

Return Value Example:

```
{
  "admins": [
    {
      "orgId": "appdev",
      "userId": "idcqa"
    },
    {
      "orgId": "appdev",
      "userId": "user1"
    }
  ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

checkAuthorization
Use this method to add an access control check to an operation. This is an asynchronous function. Certain token methods can be run only by the `Token Admin` or `AccountOwner` of the token or by the `MultipleAccountOwner` for users with multiple accounts. The access control mapping is described in the `../lib/constant.ts` file. You can modify access control by editing the `../lib/constant.ts` file. To use your own access control or to disable access control, remove the access control code from the automatically generated controller methods and custom methods.

```
ADMIN: {
   isUserTokenAdmin: ["Admin"],
   addAdmin: ["Admin"],
   removeAdmin: ["Admin"],
   getAllAdmins: ["Admin"],
 },
 TOKEN: {
   save: ["Admin"],
   getAllTokens: ["Admin"],
   get: ["Admin"],
   update: ["Admin"],
   getDecimals: ["Admin"],
   getTokensByName: ["Admin"],
   addRoleMember: ["Admin"],
   removeRoleMember: ["Admin"],
   isInRole: ["Admin", "AccountOwner"],
   getTotalMintedTokens: ["Admin"],
   getNetTokens: ["Admin"],
   getTokenHistory: ["Admin"],
 },
 ROLE: {
   getAccountsByRole: ["Admin"],
   getUsersByRole: ["Admin"],
 },
 TRANSACTION: {
   deleteTransactions: ["Admin"],
 },
 ACCOUNT: {
   createAccount: ["Admin"],
   associateToken: ["Admin"],
   getAllAccounts: ["Admin"],
   getAccountsByUser: ["Admin", "MultipleAccountOwner"],
   getAccount: ["Admin", "AccountOwner"],
   history: ["Admin", "AccountOwner"],
   getAccountTransactionHistory: ["Admin", "AccountOwner"],
   getAccountTransactionHistoryWithFilters: ["Admin", "AccountOwner"],
   getSubTransactionsById: ["Admin", TRANSACTION_INVOKER],
   getSubTransactionsByIdWithFilters: ["Admin", TRANSACTION_INVOKER],
   getAccountBalance: ["Admin", "AccountOwner"],
   getAccountOnHoldBalance: ["Admin", "AccountOwner"],
   getOnHoldIds: ["Admin", "AccountOwner"],
   getConversionHistory: ["Admin", "AccountOwner"],
 },
 ACCOUNT_STATUS: {
   get: ["Admin", "AccountOwner"],
```


Copyright © 2025, Oracle and/or its affiliates.

```
  history: ["Admin", "AccountOwner"],
  activateAccount: ["Admin"],
  suspendAccount: ["Admin"],
  deleteAccount: ["Admin"],
},
TOKEN_CONVERSION: {
  initializeExchangePoolUser: ["Admin"],
  addConversionRate: ["Admin"],
  updateConversionRate: ["Admin"],
  getConversionRate: ["Admin", "AnyAccountOwner"],
  getConversionRateHistory: ["Admin", "AnyAccountOwner"],
  tokenConversion: ["Admin", "AnyAccountOwner"],
  getExchangePoolUser: ["Admin"],
},
ERC721ADMIN: {
  isUserTokenAdmin: ["Admin"],
  addAdmin: ["Admin"],
  removeAdmin: ["Admin"],
  getAllAdmins: ["Admin"],
},
ERC721TOKEN: {
  getAllTokens: ["Admin"],
  getAllTokensByUser: ["Admin", "AccountOwner"],
  get: ["Admin", TOKEN_OWNER],
  getTokensByName: ["Admin"],
  addRoleMember: ["Admin"],
  removeRoleMember: ["Admin"],
  isInRole: ["Admin", "AccountOwner"],
  totalSupply: ["Admin"],
  totalNetSupply: ["Admin"],
  history: ["Admin"],
},
ERC721ROLE: {
  getAccountsByRole: ["Admin"],
  getUsersByRole: ["Admin"],
},
ERC721TRANSACTION: {
  deleteTransactions: ["Admin"],
},
ERC721ACCOUNT: {
  createAccount: ["Admin"],
  getAllAccounts: ["Admin"],
  getAccountByUser: ["Admin", "MultipleAccountOwner"],
  history: ["Admin", "AccountOwner"],
  getAccountTransactionHistory: ["Admin", "AccountOwner"],
  getAccountTransactionHistoryWithFilters: ["Admin", "AccountOwner"],
  balanceOf: ["Admin", "MultipleAccountOwner"],
},
ERC1155ADMIN: {
  isUserTokenAdmin: ["Admin"],
  addAdmin: ["Admin"],
  removeAdmin: ["Admin"],
  getAllAdmins: ["Admin"],
},
```


Copyright © 2025, Oracle and/or its affiliates.

```
ERC1155TOKEN: {
  getAllTokens: ["Admin"],
  get: ["Admin", TOKEN_OWNER],
  getAllTokensByUser: ["Admin", "AccountOwner"],
  totalSupply: ["Admin"],
  totalNetSupply: ["Admin"],
  getTokensByName: ["Admin"],
  getDecimals: ["Admin"],
  addRoleMember: ["Admin"],
  removeRoleMember: ["Admin"],
  isInRole: ["Admin", "AccountOwner"],
  save: ["Admin"],
  update: ["Admin"],
},
ERC1155ACCOUNT: {
  createAccount: ["Admin"],
  createUserAccount: ["Admin"],
  createTokenAccount: ["Admin"],
  associateFungibleTokenToAccount: ["Admin", "AccountOwner"],
  getAccountsByUser: ["Admin", "AccountOwner"],
  getAccount: ["Admin", "AccountOwner"],
  history: ["Admin", "AccountOwner"],
  getAllAccounts: ["Admin"],
  balanceOfBatch: ["Admin"],
  getAccountTransactionHistory: ["Admin", "AccountOwner"],
  getAccountTransactionHistoryWithFilters: ["Admin", "AccountOwner"],
  exchangeToken: ["AccountOwner"],
  getAccountDetailsByUser: ["Admin", "AccountOwner"],
},
ERC1155ROLE: {
  getAccountsByRole: ["Admin"],
  getUsersByRole: ["Admin"],
},
```

```
Ctx.ERC1155Auth.checkAuthorization(classFuncName: string, ...args)
```

Parameters:

- `classFuncName: string` – The map value between the class and methods as described in the `../lib/constant.ts` file.
- `...args` – A variable argument where `args[0]` takes the constant `'TOKEN'` and `args[1]` takes the `account_id` to add an access control check for an `AccountOwner`. To add an access control check for a `MultipleAccountOwner`, `args[1]` takes the `org_id` and `args[2]` takes the `user_id`.

Returns:

- On success, a promise. On error, a rejection with an error message.

Methods for Token Configuration Management

save
This method creates tokens. Every token that is defined has its own create method. For non-fungible tokens, if the minter role is defined in the specification file, then any user with the

Copyright © 2025, Oracle and/or its affiliates.

minter role can call this method to create an NFT. If not, any user can use this method to create (mint) NFTs. The user who calls this method becomes the owner of the NFT (whole or fractional).

```
Ctx.ERC1155Token.save(tokenAsset: <Token Class>, quantity?: number);
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset. The properties of the asset are defined in the model file.
- `quantity: number` – For non-fungible tokens only, the number of tokens to mint. The only supported value for this parameter is `1`.

Returns:

- On success, the token asset in JSON format, which can include the following information.
- `tokenMetadata` – JSON information that describes the token.
- `createdBy` – The account ID of the caller, who is the user minting the token. This property cannot be edited.
- `creationDate` – The time stamp of the minting transaction. This property cannot be edited.
- `isBurned` – This property indicates whether the token is burned. This property cannot be edited.
- `tokenName` – The name of the token. This property cannot be edited.
- `tokenDesc` – The description of the token.
- `symbol` – The symbol of the token. This property cannot be edited.
- `tokenStandard` – The standard of the token. This property cannot be edited.
- `tokenType` – The type of the token (fungible or non-fungible). This property cannot be edited.
- `tokenUnit` – The unit of the token (whole or fractional). This property cannot be edited.
- `behaviors` – A list of token behaviors. This property cannot be edited.
- `mintable` – The properties related to minting. The `max_mint_quantity` value defines the maximum number of tokens that can be created for the token class.
- `owner` – The account ID of the current owner, who is the caller of the method.
- `tokenUri` – The URI of the token.
- `quantity` – The quantity of the token.

Return Value Example (Whole NFT):

```
{
    "tokenMetadata": {
        "paintingName": "monalisa",
        "description": "monalisa painting",
        "image": "image link",
        "painterName": "Leonardo da Vinci"
    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "assetType": "otoken",
    "quantity": 1,
    "tokenId": "artnft",
    "tokenName": "artcollection",
    "tokenDesc": "artcollection nft",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 500
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-12-29T04:08:35.000Z",
    "isBurned": false,
    "tokenUri": "tu",
    "price": 10000,
    "onSaleFlag": false
}
```


Return Value Example (Fungible Token):


```
{
    "assetType": "otoken",
    "tokenId": "Loyalty",
    "tokenName": "loyalty",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
    "tokenType": "fungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 10000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "Dollar"
}
```

Return Value Example (Fractional NFT):

```
{
    "tokenMetadata": {
        "painting_name": "paint",
        "description": "Painting Description"
    },
    "assetType": "otoken",
    "tokenId": "realEstate",
    "tokenName": "realestate",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 20000
    },
    "quantity": 100,
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2023-06-14T04:20:14.000Z",
    "divisible": {
        "decimal": 2
    },
    "isBurned": false,
    "tokenUri": "www.realestate.example.com",
    "price": 1000,
    "on_sale_flag": true
}
```


Copyright © 2025, Oracle and/or its affiliates.

update
This method updates tokens. You cannot update token metadata or the token URI of non-fungible tokens.

```
Ctx.ERC1155Token.update(tokenAsset: any);
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset. The properties of the asset are defined in the model file.

Returns:

- On success, the updated token asset in JSON format.

Return Value Example (Whole NFT):

```
{
    "tokenMetadata": {
        "paintingName": "monalisa",
        "description": "monalisa painting",
        "image": "image link",
        "painterName": "Leonardo da Vinci"
    },
    "assetType": "otoken",
    "quantity": 1,
    "tokenId": "artnft",
    "tokenName": "artcollection",
    "tokenDesc": "artcollection nft",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 500
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-12-29T04:08:35.000Z",
    "isBurned": false,
    "tokenUri": "tu",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "price": 10000,
    "onSaleFlag": false
}
```

history (Token)
This method returns the history for a specified token ID.

```
Ctx.ERC1155Token.history(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON array that contains the token history.

Return Value Example (Fungible Token):

```
[
    {
        "trxId":
"ef4af760c3d7ee5e273196231d59fb91cafe6ca0f78c64747e87bc9bcbb3334b",
        "timeStamp": "2023-09-04T02:36:20.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "LoyaltyToken",
            "tokenName": "loyalty",
            "tokenDesc": "Updated Fungible Whole",
            "tokenStandard": "erc1155+",
            "tokenType": "fungible",
            "tokenUnit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 10000
            },
            "divisible": {
                "decimal": 2
            },
            "currency_name": "Rupees"
        }
    },
    {
        "trxId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"4fb391a8903633a12a545cd2ecfb57f5575241325abf59995e2a4ed96572bb09",
        "timeStamp": "2023-09-04T02:35:07.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "LoyaltyToken",
            "tokenName": "loyalty",
            "tokenDesc": "Fungible Whole",
            "tokenStandard": "erc1155+",
            "tokenType": "fungible",
            "tokenUnit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter",
                "burner_role_name": "burner"
            },
            "mintable": {
                "max_mint_quantity": 10000
            },
            "divisible": {
                "decimal": 2
            },
            "currency_name": "Dollar"
        }
    }
]
```


Return Value Example (Fractional NFT):


```
[
    {
        "txId":
"99bca74f401465206da7499cbf704dd443b3c3d94e348b1d6682ab5ee1864a08",
        "timestamp": "2023-06-20T01:09:18.000Z",
        "value": {
            "assetType": "otoken",
            "tokenId": "FNFT",
            "tokenName": "realestate",
            "tokenStandard": "erc1155+",
            "tokenType": "nonfungible",
            "tokenUnit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "quantity": 100,
                "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                "creationDate": "2023-06-20T00:53:13.000Z",
                "divisible": {
                    "decimal": 2
                },
                "isBurned": false,
                "tokenUri": "www.FNFT.example.com",
                "price": 2000,
                "on_sale_flag": true,
                "owners": [
                    {
                        "accountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                        "tokenShare": 90
                    },
                    {
                        "accountId":
"oaccount~3cddfdaa855900579d963aa6f755a4aed1f3a474a2462c1b45bd7f36df673224",
                        "tokenShare": 10
                    }
                ]
            }
        },
        {
            "txId":
"d517c61f40e7d6af2f04fe6d337b3e5108eb57030c9dc823793498fd4fed671b",
            "timestamp": "2023-06-20T00:53:13.000Z",
            "value": {
                "assetType": "otoken",
                "tokenId": "FNFT",
                "tokenName": "realestate",
                "tokenStandard": "erc1155+",
                "tokenType": "nonfungible",
                "tokenUnit": "fractional",
                "behaviors": [
                    "divisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "mintable": {
                "max_mint_quantity": 20000
            },
            "quantity": 100,
            "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "creationDate": "2023-06-20T00:53:13.000Z",
            "divisible": {
                "decimal": 2
            },
            "isBurned": false,
            "tokenUri": "www.FNFT.example.com",
            "price": 2000,
            "on_sale_flag": true,
            "owners": [
                {
                    "accountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                    "tokenShare": 100
                }
            ]
        }
    }
]
```


Return Value Example (Whole NFT):


```
[
    {
        "trxId":
"92ac6b56112acdba724dd49924d2420a7899c013c61aa40d272e8ab391a65e0f",
        "timeStamp": "2023-09-04T02:28:48.000Z",
        "value": {
            "tokenMetadata": {
                "painting_name": "monalisa",
                "description": "monalisa painting",
                "image": "image link",
                "painter_name": "Leonardo da Vinci"
            },
            "assetType": "otoken",
            "tokenId": "artnft",
            "tokenName": "artcollection",
            "tokenDesc": "Updated Token Description",
            "tokenStandard": "erc1155+",
            "tokenType": "nonfungible",
            "tokenUnit": "whole",
            "behaviors": [
                "indivisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "quantity": 1,
                "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "creationDate": "2023-09-04T02:27:19.000Z",
                "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "isBurned": false,
                "tokenUri": "www.FNFT.example.com",
                "price": 10000,
                "on_sale_flag": true
            }
        },
        {
            "trxId":
"27697dd4a8dba53bad073aa95587cd1ef173b02fd95d771a60273d301fd3bcbe",
            "timeStamp": "2023-09-04T02:27:19.000Z",
            "value": {
                "tokenMetadata": {
                    "painting_name": "monalisa",
                    "description": "monalisa painting",
                    "image": "image link",
                    "painter_name": "Leonardo da Vinci"
                },
                "assetType": "otoken",
                "tokenId": "artnft",
                "tokenName": "artcollection",
                "tokenDesc": "artcollection nft",
                "tokenStandard": "erc1155+",
                "tokenType": "nonfungible",
                "tokenUnit": "whole",
                "behaviors": [
                    "indivisible",
                    "mintable",
                    "transferable",
                    "burnable",
                    "roles"
                ],
                "roles": {
                    "minter_role_name": "minter",
                    "burner_role_name": "burner"
                },
                "mintable": {
                    "max_mint_quantity": 20000
                },
                "quantity": 1,
                "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "creationDate": "2023-09-04T02:27:19.000Z",
            "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "isBurned": false,
            "tokenUri": "www.FNFT.example.com",
            "price": 10000,
            "on_sale_flag": true
        }
    }
]
```

`getAllTokens`
This method returns all of the token assets that are saved in the state database. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Token.getAllTokens()
```

Parameters:

- none

Return Value Example:

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
      }
    }
  },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    {
      "key": "tokenTwo",
      "valueJson": {
        "assetType": "otoken",
        "tokenId": "tokenTwo",
        "tokenName": "moneytok",
        "tokenStandard": "erc1155+",
        "tokenType": "fungible",
        "tokenUnit": "fractional",
        "behaviors": [
          "divisible",
          "mintable",
          "transferable",
          "roles"
        ],
        "roles": {
          "minter_role_name": "minter",
          "burner_role_name": "burner"
        },
        "mintable": {
          "max_mint_quantity": 1000
        },
        "divisible": {
          "decimal": 2
        }
      }
    },
    {
      "key": "art",
      "valueJson": {
        "assetType": "otoken",
        "quantity": 1,
        "tokenId": "art",
        "tokenName": "artcollection",
        "tokenStandard": "erc1155+",
        "tokenType": "nonfungible",
        "tokenUnit": "whole",
        "behaviors": [
          "indivisible",
          "singleton",
          "mintable",
          "transferable",
          "burnable",
          "roles"
        ],
        "roles": {
          "minter_role_name": "minter"
        },
        "mintable": {
          "max_mint_quantity": 20000
        },
        "owner":
  "oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
        "createdBy":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "creationDate": "2022-12-08T08:52:57.000Z",
      "isBurned": true,
      "tokenUri": "art.example.com",
      "transferredBy":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
      "transferredDate": "2022-12-08T08:59:17.000Z",
      "burnedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "burnedDate": "2022-12-08T09:01:28.000Z"
    }
  },
  {
    "key": "FNFT",
    "valueJson": {
        "assetType": "otoken",
        "tokenId": "FNFT",
        "tokenName": "realestate",
        "tokenStandard": "erc1155+",
        "tokenType": "nonfungible",
        "tokenUnit": "fractional",
        "behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "roles": {
            "minter_role_name": "minter",
            "burner_role_name": "burner"
        },
        "mintable": {
            "max_mint_quantity": 20000
        },
        "quantity": 100,
        "createdBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
        "creationDate": "2023-06-20T00:53:13.000Z",
        "divisible": {
            "decimal": 2
        },
        "isBurned": false,
        "tokenUri": "www.FNFT.example.com",
        "price": 2000,
        "on_sale_flag": true
    }
  },
]
```


Copyright © 2025, Oracle and/or its affiliates.

`get (Token)`
This method returns a token object if the token is present in the state database. This method can be called only by a `Token Admin` of the chaincode or the token owner.

```
this.Ctx.ERC1155Token.get(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token to get.

Return Value Example (Whole NFT):

```
{
  "assetType": "otoken",
  "quantity": 1,
  "tokenId": "art",
  "tokenName": "artcollection",
  "tokenStandard": "erc1155+",
  "tokenType": "nonfungible",
  "tokenUnit": "whole",
  "behaviors": [
    "indivisible",
    "singleton",
    "mintable",
    "transferable",
    "burnable",
    "roles"
  ],
  "roles": {
    "minter_role_name": "minter"
  },
  "mintable": {
    "max_mint_quantity": 20000
  },
  "owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "createdBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "creationDate": "2022-12-08T08:52:57.000Z",
  "isBurned": true,
  "tokenUri": "example.com",
  "transferredBy":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "transferredDate": "2022-12-08T08:59:17.000Z",
  "burnedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "burnedDate": "2022-12-08T09:01:28.000Z"
}
```

Return Value Example (Fungible Token):

```
{
    "assetType": "otoken",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "tokenId": "Loyalty",
    "tokenName": "loyalty",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
    "tokenType": "fungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 10000
    },
    "divisible": {
        "decimal": 2
    },
    "currency_name": "Dollar"
}
```

Return Value Example (Fractional NFT):

```
{
    "tokenMetadata": {
        "painting_name": "paint",
        "description": "Painting Description"
    },
    "assetType": "otoken",
    "tokenId": "realEstate",
    "tokenName": "realestate",
    "tokenDesc": "Token Description",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "fractional",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 20000
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    "quantity": 100,
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2023-06-14T04:20:14.000Z",
    "divisible": {
        "decimal": 2
    },
    "isBurned": false,
    "tokenUri": "www.realestate.example.com",
    "price": 1000,
    "on_sale_flag": true,
    "owners": [
        {
            "accountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "tokenShare": 100
        }
    ]
}
```

`getAllTokensByUser`
This method returns all of the token assets that are owned by a specified user. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Token.getAllTokensByUser(accountId: string)
```

Parameters:

- `accountId: string` – The account ID of the user.

Return Value Example:

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
```


Copyright © 2025, Oracle and/or its affiliates.

```
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
      }
    }
  },
  {
    "key": "nftToken",
    "valueJson": {
      "assetType": "otoken",
      "quantity": 1,
      "tokenId": "nftToken",
      "tokenName": "artcollection",
      "tokenStandard": "erc1155+",
      "tokenType": "nonfungible",
      "tokenUnit": "whole",
      "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter"
      },
      "mintable": {
        "max_mint_quantity": 20000
      },
      "owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "createdBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "creationDate": "2022-12-08T09:10:21.000Z",
      "isBurned": false,
      "tokenUri": "example.com"
    }
  }
]
```

`ownerOf`
This method returns the account ID, organization ID, and user ID of the owner of the specified token ID.

```
Ctx.ERC1155Token.ownerOf(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "accountId":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
    "orgId": "appdev",
    "userId": "idcqa"
}
```

tokenURI
This method returns the URI of a specified token. Anyone can call this method.

```
Ctx.ERC1155Token.tokenURI(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value Example:

```
{
    "tokenUri": "example.com"
}
```

name
This method returns the name of the token class. Anyone can call this method.

```
Ctx.ERC1155Token.name(tokenId: string)
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value Example:

```
{"tokenName": "artcollection"}
```

totalSupply
This method returns the total number of minted tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.totalSupply(token: any)
```

Parameters:

- `token: any` – The token asset.

Return Value Example:

```
{"totalSupply": 110}
```


Copyright © 2025, Oracle and/or its affiliates.

totalNetSupply
This method returns the total number of minted tokens minus the number of burned tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.totalNetSupply(token: any)
```

Parameters:

- `token: any` – The token asset.

Return Value Example:

```
{"totalNetSupply": 105}
```

getTokensByName
This method returns all of the token assets for a specified token name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Token.getTokensByName(tokenName: string)
```

Parameters:

- `tokenName: string` – The name of the token.

Return Value Example:

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenOne",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
```


Copyright © 2025, Oracle and/or its affiliates.

```
      }
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "assetType": "otoken",
      "tokenId": "tokenTwo",
      "tokenName": "moneytok",
      "tokenStandard": "erc1155+",
      "tokenType": "fungible",
      "tokenUnit": "fractional",
      "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
      },
      "mintable": {
        "max_mint_quantity": 1000
      },
      "divisible": {
        "decimal": 2
      }
    }
  }
]
```

`getDecimals`
This method returns the number of decimal places for a specified token. If the divisible behavior is not specified for the token, then the default value of zero decimal places is returned.

```
Ctx.ERC1155Token.getDecimals(token)
```

Parameters:

- `token: any` – The token asset.

Return Value Example:

```
2
```

Methods for Account Management

`createAccount`
This method creates an account for a specified user and associated token accounts for fungible or non-fungible tokens. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a

Copyright © 2025, Oracle and/or its affiliates.

user has. Users must have accounts in the network to complete token-related operations. This method can be called only by a `Token Admin` of the chaincode.
A user account has a unique ID, which is formed by an SHA-256 hash of the `orgId` parameter and the `userId` parameter.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this single non-fungible token account.

```
Ctx.ERC1155Account.createAccount(orgId: string, userId: string, ftAccount:
boolean, nftAccount: boolean)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `ftAccount: boolean` – If true, a fungible token account is created and associated with the user account.
- `nftAccount: boolean` – If true, a non-fungible token account is created and associated with the user account.

Returns:

- On success, a JSON object of the account that was created.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~cf20877546f52687f387e7c91d88b9722c97e1a456cc0484f40c747f7804feae",
  "userId": "user1",
  "orgId": "appdev",
  "totalAccounts": 2,
  "totalFtAccounts": 1,
  "associatedFtAccounts": [
    {
      "accountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
      "tokenId": ""
    }
  ],
  "associatedNftAccount":
"oaccount~73c3e835dac6d0a56ca9d8def08269f83cefd59b9d297fe2cdc5a9083828fa58"
}
```


Copyright © 2025, Oracle and/or its affiliates.

createUserAccount
This method creates an account for a specified user. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a user has. Users must have accounts in the network to complete token-related operations.
An account ID is an SHA-256 hash of the `orgId` parameter and the `userId`parameter. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.ERC1155Account.createUserAccount(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON object of the user account that was created.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "userId": "idcqa",
  "orgId": "appdev",
  "totalAccounts": 0,
  "totalFtAccounts": 0,
  "associatedFtAccounts": [],
  "associatedNftAccount": ""
}
```

createTokenAccount
This method creates a fungible or non-fungible token account to associate with a user account.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this single non-fungible token account.
This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.ERC1155Account.createTokenAccount(orgId: string, userId: string,
tokenType: string)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenType: TokenType` – The type of token account to create. The only supported token types are `nonfungible` and `fungible`.

Returns:

- On success, a JSON object of the token account that was created.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "userId": "idcqa",
  "orgId": "appdev",
  "totalAccounts": 1,
  "totalFtAccounts": 1,
  "associatedFtAccounts": [
    {
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "tokenId": ""
    }
  ],
  "associatedNftAccount": ""
}
```

`associateTokenToToken`
This method associates a user's fungible token account to a particular fungible token.

```
Ctx.ERC1155Account.associateTokenToToken(accountId: string, tokenId: string)
```

Parameters:

- `accountId: string` – The user account ID.
- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON object of the user account, which shows that the fungible token was associated to the token account. For example, in the following example, the first object in the `associatedFtAccounts` array shows that the fungible token account ID and the token ID are associated.

Return Value Example:

```
{
  "assetType": "ouaccount",
  "accountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "userId": "idcqa",
  "orgId": "appdev",
  "totalAccounts": 1,
  "totalFtAccounts": 1,
  "associatedFtAccounts": [
    {
      "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
      "tokenId": "tokenOne"
    }
  ],
  "associatedNftAccount": ""
}
```

getAccountHistory
This method returns history for a specified token account.

```
Ctx.ERC1155Account.getAccountHistory(accountId: string)
```

Parameters:

- `accountId: string` – The user account ID.

Returns:

- On success, an array of JSON objects that describes the account history.

Return Value Example:

```
[
    {
        "trxId":
"a2cfc6fc064334d6b9931cdf67193711ec2ff5c50a4714f11855fe7384f00e35",
        "timeStamp": "2023-06-06T14:44:31.000Z",
        "value": {
            "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "assetType": "oaccount",
            "bapAccountVersion": 1,
            "balance": 100,
            "orgId": "appdev",
            "tokenId": "loy1",
            "tokenName": "loyalty",
            "tokenType": "fungible",
            "userId": "idcqa"
        }
    },
    {
        "trxId":
"de483cf7505ae4e7018c4b604c3ab9327c2fb1f802d9408e22735667c1d6997f",
        "timeStamp": "2023-06-06T14:43:23.000Z",
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion": 0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "userId": "idcqa",
            "orgId": "appdev",
            "tokenType": "fungible",
            "tokenId": "loy1",
            "tokenName": "loyalty",
            "balance": 0
        }
    },
    {
        "trxId":
"db053e653d3ad9aa5b7b6e04b7cd51aacfbb413272d857a155b60d2a6a12bf4d",
        "timeStamp": "2023-06-05T16:59:08.000Z",
        "value": {
            "assetType": "oaccount",
            "bapAccountVersion": 0,
            "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "userId": "idcqa",
            "orgId": "appdev",
            "tokenType": "fungible",
            "tokenId": "",
            "balance": 0
        }
    }
]
```

`getAccountWithStatus`
This method returns token account details, including account status, for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC1155Account.GetAccountWithStatus(accountId, tokenId...)
```

Parameters:

- `userAccountId: string` – The account ID of the user.
- `tokenId?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON object that includes token account details, including the account status.

Return Value Example (Non-Fungible Token Account):

```
{
    "assetType": "oaccount",
    "bapAccountVersion": 1,
    "status": "active",
    "accountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
    "userId": "idcqa",
    "orgId": "appdev",
    "tokenType": "nonfungible",
    "noOfNfts": 1
  }
```

Return Value Example (Fungible Token Account):

```
{
  "bapAccountVersion": 0,
  "assetType": "oaccount",
  "status": "active",
  "accountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "userId": "idcqa",
  "orgId": "appdev",
  "tokenType": "fungible",
  "tokenId": "t1",
  "tokenName": "loyalty",
  "balance": 0
}
```

`getAccount`
This method returns token account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC1155Account.getAccount(userAccountId: string, tokenId: string)
```

Parameters:

- `userAccountId: string` – The account ID of the user.
- `tokenId?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON object that includes token account details. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example (Non-Fungible Token Account):

```
{
    "assetType": "oaccount",
    "bapAccountVersion": 0,
    "accountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
    "userId": "idcqa",
    "orgId": "appdev",
    "tokenType": "nonfungible",
    "noOfNfts": 3
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fungible Token Account):

```
{
    "assetType": "oaccount",
    "bapAccountVersion": 0,
    "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
    "userId": "idcqa",
    "orgId": "appdev",
    "tokenType": "fungible",
    "tokenId": "loy1",
    "tokenName": "loyalty",
    "balance": 50
}
```

getAllAccounts
This method returns details of all user accounts.

```
Ctx.ERC1155Account.getAllAccounts()
```

Parameters:

- none

Return Value Example:

```
[
        {
            "assetType": "ouaccount",
            "accountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
            "userId": "idcqa",
            "orgId": "appdev",
            "totalAccounts": 2,
            "totalFtAccounts": 1,
            "associatedFtAccounts": [
                {
                    "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
                    "tokenId": "loy1"
                }
            ],
            "associatedNftAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371"
        },
        {
            "assetType": "ouaccount",
            "accountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
            "userId": "user1_minter",
            "orgId": "appdev",
            "totalAccounts": 2,
            "totalFtAccounts": 1,
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "associatedFtAccounts": [
                {
                    "accountId":
"oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c",
                    "tokenId": "loy1"
                }
            ],
            "associatedNftAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446"
        },
    ]
```

`getAccountDetailsByUser`
This method returns an account summary for a specified user and details of fungible and non-fungible tokens that are associated with the user..

```
Ctx.ERC1155Account.getAccountDetailsByUser(orgId: string, userId: string)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes and account summary for the specified user and details of fungible and non-fungible tokens that are associated with the user. For fractional non-fungible tokens, the `tokenShare` property in the `associatedNFTs` section shows the share that the user owns

Return Value Example:

```
{
    "userAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
    "associatedFTAccounts": [
        {
            "accountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "tokenId": "FT",
            "balance": 50
        }
    ],
    "associatedNFTAccount": {
        "accountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "associatedNFTs": [
            {
                "nftTokenId": "FNFT",
                "tokenShare": 100
            },
            {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "nftTokenId": "FNFT2",
                "tokenShare": 110
            },
            {
                "nftTokenId": "NFT"
            }
        ]
    }
}
```

`getUserByAccountId`
This method returns the user details of a specified account ID.

```
Ctx.ERC1155Account.getUserByAccountId(accountId: string)
```

Parameters:

- `accountId: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`orgId` and `userId`).

Return Value Example:

```
{
    "orgId": "appdev",
    "userId": "idcqa"
}
```

Methods for Role Management

`AddRoleMember`
This method adds a role to a specified user and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.addRoleMember(role: string, userAccountId: string, token:
any)
```

Parameters:

- `userAccountId: string` – The account ID of the user.
- `role: string` – The name of the role to add to the specified user.
- `token: any` – The token asset.

Returns:

- On success, a message with account details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "msg": "Successfully added role 'minter' to Account Id:
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(Org-Id: appdev, User-Id: idcqa)"
}
```

`isInRole`
This method returns a Boolean value to indicate if a user has a specified role. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.isInRole(role: string, userAccountId: string, token: any)
```

Parameters:

- `userAccountId: string` – The account ID of the user.
- `role: string` – The name of the role to search for.
- `token: any` – The token asset.

Return Value Example:

```
{
    "result": true,
    "msg": "Account Id
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(Org-Id: appdev, User-Id: idcqa) has minter role"
}
```

`removeRoleMember`
This method removes a role from a specified user and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.removeRoleMember(role: string, userAccountId: string, token:
any)
```

Parameters:

- `userAccountId: string` – The account ID of the user.
- `role: string` – The name of the role to remove from the specified user.
- `token: any` – The token asset.

Return Value Example:

```
{
  "msg": "Successfully removed role 'minter' from Account Id:
oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b
(Org-Id: appdev, User-Id: user1)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

getAccountsByRole
This method returns a list of all account IDs for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.getAccountsByRole(role: string, token: any)
```

Parameters:

- `role: string` – The name of the role to search for.
- `token: any` – The token asset.

Return Value Example:

```
{
  "accounts": [

"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",

"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b"
  ]
}
```

getUsersByRole
This method returns a list of all users for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.getUsersByRole(role: string, token: any)
```

Parameters:

- `role: string` – The name of the role to search for.
- `token: any` – The token asset.

Return Value Example:

```
{
    "users": [
        {
            "accountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "orgId": "appdev",
            "userId": "idcqa"
        },
        {
            "accountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
            "orgId": "appdev",
            "userId": "user1"
        }
    ]
}
```

Methods for Transaction History Management

Copyright © 2025, Oracle and/or its affiliates.

`getAccountTransactionHistory`
This method returns account transaction history. This method can be called only by a `Token Admin` of the chaincode or by the account owner. For non-fungible tokens, this method can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Account.getAccountTransactionHistory(accountId: string)
```

Parameters:

- `accountId: string` – The token account ID.

Return Value Example:


```
[
    {
        "transactionId":
"otransaction~3a6b23c3003626f3947e990eddbd7ac23398d2200e2eb3eac745e6ddfae140bc
~7c88c736df38d5622512f1e8dcdd50710eb47c953f1ecb24ac44790a9e2f475b",
        "timestamp": "2023-06-06T14:48:08.000Z",
        "tokenId": "FNFT",
        "transactedAmount": 10,
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "transactionType": "DEBIT",
        "balance": 90
    },
    {
        "transactionId":
"otransaction~3a6b23c3003626f3947e990eddbd7ac23398d2200e2eb3eac745e6ddfae140bc
~178e3730bc5bee50d02f1464a4eebf733a051905f651e5789039adb4a3edc114",
        "timestamp": "2023-06-06T14:48:08.000Z",
        "tokenId": "NFT",
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "transactionType": "DEBIT"
    },
    {
        "transactionId":
"otransaction~c369929e28e78de06c72d020f1418c9a154a7dd280b2e22ebb4ea4485e249124
~a7cefb22ff39ee7e36967be71de27da6798548c872061a62dabc56d88d50b930",
        "timestamp": "2023-06-06T14:47:08.000Z",
        "tokenId": "NFT",
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "transactionType": "MINT"
    },
    {
        "transactionId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~114a1bc78d04be48ee6dc140c32c042ee9481cb118959626f090eec744522422
~e4eb15d9354f694230df8835ade012100d82aa43672896a2c7125a86e3048f9f",
        "timestamp": "2023-06-05T17:17:57.000Z",
        "tokenId": "FNFT",
        "transactedAmount": 100,
        "triggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "transactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "transactionType": "MINT",
        "balance": 100
    }
]
```

getTransactionById
This method returns the transaction details for a specified transaction ID.

```
Ctx.ERC1155Transaction.getTransactionById(transactionId: string)
```

Parameters:

- `transactionId: string` – The ID of the transaction.

Return Value Example:

```
{
  "transactionId":
"otransaction~9ea7b05ab099f7ff4db8342b8c3609031f1d54f11205906e7f1fe88661fe3cbe
~33b59ce0c89e96c1e16449f24301581e8e71954f38ad977c7eb6f065e87f2a53",
  "history": [
    {
      "trxId":
"9ea7b05ab099f7ff4db8342b8c3609031f1d54f11205906e7f1fe88661fe3cbe",
      "timeStamp": "2022-12-08T09:01:28.000Z",
      "value": {
        "assetType": "otransaction",
        "transactionId":
"otransaction~9ea7b05ab099f7ff4db8342b8c3609031f1d54f11205906e7f1fe88661fe3cbe
~33b59ce0c89e96c1e16449f24301581e8e71954f38ad977c7eb6f065e87f2a53",
        "tokenId": "tokenOne",
        "fromAccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
        "toAccountId": "",
        "transactionType": "BURN",
        "amount": 5,
        "timestamp": "2022-12-08T09:01:28.000Z",
        "triggeredByUserAccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc"
      }
    }
  ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`deleteTransactions`
This method deletes transactions before a specified time stamp from the state database.

```
Ctx.ERC1155Transaction.deleteTransactions(referenceTime: Date)
```

Parameters:

- `referenceTime: Date` – All transactions before this time stamp will be deleted.

Return Value Example:

```
{
  "msg": "Successfuly deleted transaction older than date: Thu Apr 07 2022
21:18:59 GMT+0000 (Coordinated Universal Time).",
  "transactions": [

"otransaction~30513757d8b647fffaafac440d743635f5c1b2e41b25ebd6b70b5bbf78a2643f
",

"otransaction~ac0e908c735297941ba58bb208ee61ff4816a1e54c090d68024f82adf743892b
"
  ]
}
```

Methods for Token Behavior Management - Mintable Behavior

`mintBatch`
This method creates (mints) multiple tokens in a batch operation. This method creates only fungible tokens or fractional non-fungible tokens.
For fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. You cannot mint more than the `max_mint_quantity` property of the token, if that property was specified when the token was created or updated.
For non-fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. Additionally, the caller must also be the creator of the token. There is no upper limit to the quantity of fractional non-fungible tokens that can be minted.
You cannot use this method to mint a whole non-fungible token.

```
Ctx.ERC1155Token.mintBatch(accountId: string, tokenIds: string[], quantities:
number[])
```

Parameters:

- `accountId: string` – The account ID of the user.
- `tokenIds: string[]` – The list of token IDs to mint tokens for.
- `quantity: number[]` – The list of quantities of tokens to mint, corresponding to the token ID array.

Returns:

- On success, a JSON object that includes details on the minted tokens.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Successfully minted batch of tokens for User-Account-Id
ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38
(Org-Id: appdev, User-Id: idcqa).",
    "details": [
        {
            "msg": "Successfully minted 100 tokens of fractional tokenId:
plot55 to Org-Id: appdev, User-Id: idcqa"
        },
        {
            "msg": "Successfully minted 100 tokens of tokenId: loyalty to
Token-Account-Id
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e"
        }
    ]
}
```

Methods for Token Behavior Management - Transferable Behavior

`batchTransferFrom`
This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.
For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.
For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.
This method does not validate that the caller of the method is the specified sender.

```
Ctx.ERC1155Token.batchTransferFrom(fromUserAccountId: string,
toUserAccountId: string, tokenIds: string[], quantities: number[])
```

Parameters:

- `fromUserAccountId: string` – The account ID of the sender and token owner in the current organization.
- `toUserAccountId: string` – The account ID of the receiver.
- `tokenIds: string[]` – A list of token IDs for the tokens to transfer.
- `quantity: number[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

- On success, a message with details for each token transfer.

Return Value Example:

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
```


Copyright © 2025, Oracle and/or its affiliates.

```
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`safeBatchtransferFrom`

This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.

For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.

For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.

The caller of the method must be the specified sender.

```
Ctx.ERC1155Token.safeBatchTransferFrom(fromUserAccountId: string,
toUserAccountId: string, tokenIds: string[], quantities: number[])
```

Parameters:

- `fromUserAccountId: string` – The account ID of the sender and token owner in the current organization.
- `toUserAccountId: string` – The account ID of the receiver.
- `tokenIds: string[]` – A list of token IDs for the tokens to transfer.
- `quantity: number[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

- On success, a message with details for each token transfer.

Return Value Example:

```
[
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`balanceOfBatch`

This method completes a batch operation that gets the balance of token accounts. The account details are specified in three separate lists of organization IDs, user IDs, and token IDs. This method can be called only by a `Token Admin` of the chaincode or by account owners. Account owners can see balance details only for accounts that they own.

```
Ctx.ERC1155Account.balanceOfBatch(accountIds: string[], tokenIds: string[])
```

Parameters:

- `accountIds: string[]` – A list of the user account IDs.
- `tokenIds: string[]` – A list of the token IDs.

Return Value Example:

```
[
    {
        "orgId": "appdev",
        "userId": "idcqa",
        "userAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "tokenAccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "tokenId": "FNFT",
        "balance": 100
    },
    {
        "orgId": "appdev",
        "userId": "idcqa",
        "userAccountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "tokenAccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
        "tokenId": "FT",
        "balance": 50
    },
    {
        "orgId": "appdev",
        "userId": "user1_minter",
        "userAccountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
        "tokenAccountId":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "tokenId": "FNFT",
        "balance": 10
    }
]
```

exchangeToken
This method exchanges tokens between specified accounts. This method only supports exchanging between an NFT (whole or fractional) and a fungible token or a fungible token and an NFT (whole or fractional). This method can be called only by the account owner.

```
Ctx.ERC1155Token.exchangeToken( fromTokenId: string, fromUserAccountId:
string, fromTokenQuantity: number, toTokenId: string, toUserAccountId:
string, toTokenQuantity: number)
```

Parameters:

- `fromTokenId: string` – The ID of the token that the sender owns.
- `fromUserAccountId: string` – The account ID of the sender.
- `fromTokenQuantity: number` – The quantity of tokens from the sender to exchange with the receiver.
- `toTokenId: string` – The ID of the token that the receiver owns.
- `toUserAccountId: string` – The account ID of the receiver.
- `toTokenQuantity: number` – The quantity of tokens from the receiver to exchange with the sender.

Returns:

- On success, a message with token exchange details.

Return Value Example:

```
{
    "msg": "Succesfully exchanged 10 tokens of type nonfungible with tokenId:
[r1] from Account
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(OrgId: appdev, UserId: idcqa) to 10 tokens of type fungible with tokenId:
[loy1] from Account
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
```


Copyright © 2025, Oracle and/or its affiliates.

```
(OrgId: appdev, UserId: user1_minter)"
}
```

Methods for Token Behavior Management - Burnable Behavior

`burn`
This method deactivates, or burns, the specified fungible and non-fungible tokens.

```
Ctx.ERC1155Token.burn(accountId: string, tokenIds: string[], quantities:
number[])
```

Parameters:

- `accountId: string` – The account ID of the user.
- `tokenIds: string[]` – The list of token IDs to burn.
- `quantity: number[]` – The list of quantities of tokens to burn, corresponding to the token ID array.

Returns:

- On success, a message with details about the burn operations.

Return Value Example:

```
[
  {
    "msg": "Successfully burned NFT token: 'art' from Account-Id:
oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6
(Org-Id: appdev, User-Id: idcqa)"
  },
  {
    "msg": "Successfully burned 5 tokens of tokenId: tokenOne from Account-ID
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(Org-Id: appdev, User-Id: idcqa)"
  },
  {
    "msg": "Successfully burned 2 token share of tokenId: FNFT from Account-
ID oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a
(Org-Id: AutoF1377358917, User-Id: idcqa)"
  }
]
```

TypeScript Methods for ERC-1155 NFT Locking

Blockchain App Builder automatically generates methods that you can use to lock non-fungible tokens that use the extended ERC-1155 standard.

A locked token cannot be burned or transferred to other users. All other properties, such as the token's state, owner, and history are preserved. You can use the NFT locking functionality when transferring a token to another blockchain network, such as Ethereum or Polygon.

Before you can lock NFTs, you must assign the vault manager role to a user. The vault manager is a special type of role, a `TokenSys` role. `TokenSys` roles are different from asset-based roles such as burner, minter, and notary, and from administrative roles such as `Token Admin` and `Org Admin`. Currently Blockchain App Builder supports the `vault TokenSys` role. The

Copyright © 2025, Oracle and/or its affiliates.

single user who has the `vault` role for a chaincode is the vault manager of the chaincode, and can manage locked NFTs.

The typical flow for using the NFT locking functionality follows these steps.

- Create a non-fungible token that has the lockable behavior.
- Use the `addTokenSysRole` method to give the `vault` role to a user, the vault manager.
- Call the `lockNFT` method to lock a non-fungible token, specified by the token ID.

TokenSys Role Management Methods

`addTokenSysRole`
This method adds a `TokenSys` role to a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async addTokenSysRole(orgId: string, userId: string, role: string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.addTokenSysRoleMember",
"TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId);
  return await this.Ctx.ERC1155Token.addTokenSysRoleMember(role,
userAccountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the `TokenSys` role to give to the user.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully added role 'vault' to Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

`isInTokenSysRole`
This method returns a Boolean value to indicate if a user has a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async isInTokenSysRole(orgId: string, userId: string, role: string) {
  await
```


Copyright © 2025, Oracle and/or its affiliates.

```
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.isInTokenSysRole",
"TOKEN", {orgId: orgId, userId: userId });
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId);
  return await this.Ctx.ERC1155Token.isInTokenSysRole(role, userAccountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the `TokenSys` role to check.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "result": true,
    "msg": "Account Id
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1) has vault role"
}
```

`removeTokenSysRole`
This method removes a `TokenSys` role from a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string())
public async removeTokenSysRole(orgId: string, userId: string, role: string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.removeTokenSysRoleMember
", "TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId);
  return await this.Ctx.ERC1155Token.removeTokenSysRoleMember(role,
userAccountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the `TokenSys` role to remove.

Returns:

- On success, a message that contains relevant details of the operation.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Successfully removed role 'vault' from Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

`transferTokenSysRole`
This method transfers a `TokenSys` role from a user to another user. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.string())
public async transferTokenSysRole(fromOrgId: string, fromUserId: string,
toOrgId: string, toUserId: string, role: string) {
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.transferTokenSysRole",
"TOKEN");
    const fromUserAccountId = await
this.Ctx.ERC1155Account.generateAccountId(fromOrgId, fromUserId);
    const toUserAccountId = await
this.Ctx.ERC1155Account.generateAccountId(toOrgId, toUserId);
    return await this.Ctx.ERC1155Token.transferTokenSysRole(role,
fromUserAccountId, toUserAccountId);
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role from.
- `fromUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role from.
- `toOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role to.
- `toUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role to.
- `role: string` – The name of the `TokenSys` role to transfer.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully transfered role 'vault' from Account Id:
ouaccount~f4e311528f03fffa7810753d643f66289ff6c9080fcf839902f28a1d3aff1789
(Org-Id: Org1MSP, User-Id: user1) to Account Id:
ouaccount~ae5be2ae8f98d6d32f5d02b43877d987114e7937c7bacbc30390dcce09996a19
```


Copyright © 2025, Oracle and/or its affiliates.

```
(Org-Id: Org1MSP, User-Id: user2)"
}
```

getAccountsByTokenSysRole
This method returns a list of all account IDs for a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string())
public async getAccountsByTokenSysRole(role: string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getAccountsByTokenSysRol
e", "TOKEN");
  return await this.Ctx.ERC1155Token.getAccountsByTokenSysRole(role);
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "accountIds": [

"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba"
    ]
}
```

getUsersByTokenSysRole
This method returns user information for all users with a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string())
public async getUsersByTokenSysRole(role: string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getUsersByTokenSysRole",
 "TOKEN");
  return await this.Ctx.ERC1155Token.getUsersByTokenSysRole(role);
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
   "users":[
      {

"accountId":"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2
caf03eba",
         "orgId":"Org1MSP",
         "userId":"user1"
      }
   ]
}
```

NFT Locking Methods

`lockNFT`
This method locks a specified non-fungible token. To lock a token, there must be a user with the `TokenSys vault` role, who acts as the vault manager. This method can be called only by the owner of the token.

```
@Validator(yup.string())
public async lockNFT(orgId: string, userId: string, tokenId: string) {
  return await this.Ctx.ERC1155Token.lockNFT(orgId, userId, tokenId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user (optional).
- `tokenID: string` – The ID of the token to lock.

Returns:

- On success, a JSON representation of the token object.

Return Value Example:

```
{
   "assetType":"otoken",
   "tokenId":"token1",
   "tokenName":"artcollection",
   "tokenStandard":"erc1155+",
   "tokenType":"nonfungible",
   "tokenUnit":"whole",
   "behaviors":[
      "indivisible",
      "mintable",
      "transferable",
      "burnable",
      "lockable",
      "roles"
```


Copyright © 2025, Oracle and/or its affiliates.

```
   ],
   "roles":{
      "minter_role_name":"minter"
   },
   "mintable":{
      "max_mint_quantity":20000
   },
   "quantity":1,

"createdBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
   "creationDate":"2023-10-20T09:16:29.000Z",

"owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
   "isBurned":false,
   "isLocked":true,
   "tokenUri":"token1.example.com",
   "price":120,
   "on_sale_flag":false
}
```

isNFTLocked

This method returns a Boolean value to indicate if a specified token is locked. This method can be called only by the token owner, the vault manager (the user with the `TokenSys vault` role), or a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string())
public async isNFTLocked(tokenId: string) {
  try {
    await this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.isNFTLocked",
"TOKEN", { tokenId });
  } catch(err) {
    const isCallerTokenSysRoleHolder = await
this.Ctx.ERC1155Token.isCallerTokenSysRoleHolder(TOKEN_SYS_ROLE_TYPE.VAULT);
    if(!isCallerTokenSysRoleHolder)
      throw err;
  }
  const isNFTLocked = await this.Ctx.ERC1155Token.isNFTLocked(tokenId);
  return {isNFTLocked};
}
```

Parameters:

- `tokenID: string` – The ID of the token.

Returns:

- On success, a message that contains relevant details of the operation.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
   "isNFTLocked":true
}
```

`getAllLockedNFTs`
This method returns a list of all locked non-fungible tokens. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator()
public async getAllLockedNFTs() {
  try {
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getAllLockedNFTs",
"TOKEN");
  } catch(err) {
    const isCallerTokenSysRoleHolder = await
this.Ctx.ERC1155Token.isCallerTokenSysRoleHolder(TOKEN_SYS_ROLE_TYPE.VAULT);
    if(!isCallerTokenSysRoleHolder)
      throw err;
  }
  return this.Ctx.ERC1155Token.getAllLockedNFTs();
}
```

Parameters:

- None

Returns:

- On success, an array of the locked non-fungible token objects.

Return Value Example:

```
[
   {
      "key":"token1",
      "valueJson":{
         "assetType":"otoken",
         "tokenId":"token1",
         "tokenName":"artcollection",
         "tokenStandard":"erc1155+",
         "tokenType":"nonfungible",
         "tokenUnit":"whole",
         "behaviors":[
            "indivisible",
            "mintable",
            "transferable",
            "burnable",
            "lockable",
            "roles"
         ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "roles":{
               "minter_role_name":"minter"
            },
            "mintable":{
               "max_mint_quantity":20000
            },
            "quantity":1,

"createdBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
            "creationDate":"2023-10-20T09:16:29.000Z",

"owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
            "isBurned":false,
            "isLocked":true,
            "tokenUri":"token1.example.com",
            "price":120,
            "on_sale_flag":false
         }
      }
]
```

`getAllLockedNFTsByOrg`
This method returns a list of all locked non-fungible tokens for a specified organization and optionally a specified user. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
@GetMethod()
@Validator(yup.string(), yup.string())
public async getLockedNFTsByOrg(orgId: string, userId?: string) {
  try {
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155TOKEN.getLockedNFTsByOrg",
"TOKEN");
  } catch(err) {
    const isCallerTokenSysRoleHolder = await
this.Ctx.ERC1155Token.isCallerTokenSysRoleHolder(TOKEN_SYS_ROLE_TYPE.VAULT);
    if(!isCallerTokenSysRoleHolder)
      throw err;
  }
  return await this.Ctx.ERC1155Token.getLockedNFTsByOrg(orgId, userId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user (optional).

Returns:

- On success, an array of the locked non-fungible token objects.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
[
   {
      "key":"token1",
      "valueJson":{
         "assetType":"otoken",
         "tokenId":"token1",
         "tokenName":"artcollection",
         "tokenStandard":"erc1155+",
         "tokenType":"nonfungible",
         "tokenUnit":"whole",
         "behaviors":[
            "indivisible",
            "mintable",
            "transferable",
            "burnable",
            "lockable",
            "roles"
         ],
         "roles":{
            "minter_role_name":"minter"
         },
         "mintable":{
            "max_mint_quantity":20000
         },
         "quantity":1,

"createdBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "creationDate":"2023-10-20T09:16:29.000Z",

"owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "isBurned":false,
         "isLocked":true,
         "tokenUri":"token1.example.com",
         "price":120,
         "on_sale_flag":false
      }
   }
]
```


TypeScript Methods for ERC-1155 Token Account Status

Blockchain App Builder automatically generates methods that you can use to manage account status for tokens that use the extended ERC-1155 standard.

You can use the following methods to put token user accounts in the active, suspended, or deleted states.

When an account is suspended, the account user cannot complete any write operations, which include minting, burning, and transferring tokens. Additionally, other users cannot transfer tokens to a suspended account. A suspended account can still complete read operations.

Copyright © 2025, Oracle and/or its affiliates.

An account with a non-zero token balance cannot be deleted. You must transfer or burn all tokens in an account before you can delete the account. After an account is in the deleted state, the account state cannot be changed back to active or suspended.

- Automatically Generated Account Status Methods
- Account Status SDK Methods

Automatically Generated Account Status Methods

`getAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
@Validator(yup.string(), yup.string(), yup.string())
  public async getAccountStatus(orgId: string, userId: string, tokenId ?:
string) {
    const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
    let tokenAccount = await
this.Ctx.ERC1155Account.getAccount(userAccountId, tokenId);
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT_STATUS.get", "TOKEN",
{ accountId: tokenAccount.accountId });
    try {
      return await
this.Ctx.ERC1155AccountStatus.getAccountStatus(tokenAccount.accountId);
    } catch (err) {
      return await
this.Ctx.ERC1155AccountStatus.getDefaultAccountStatus(tokenAccount.accountId);
    }
  }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

getAccountStatusHistory
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
public async getAccountStatusHistory(orgId: string, userId: string,
tokenId ?: string) {
    const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
    let tokenAccount = await
this.Ctx.ERC1155Account.getAccount(userAccountId, tokenId);
    await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT_STATUS.history",
"TOKEN", { accountId: tokenAccount.accountId });
    const status_id = await
this.Ctx.ERC1155AccountStatus.generateAccountStatusId(tokenAccount.accountId);
    let accountStatusHistory: any;
    try {
      accountStatusHistory = await
this.Ctx.ERC1155AccountStatus.history(status_id);
    } catch (err) {
      return [];
    }
    return accountStatusHistory;
  }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, the account status history in JSON format.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
```


Copyright © 2025, Oracle and/or its affiliates.

```
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

activateAccount
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode. Deleted accounts cannot be activated. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
@Validator(yup.string(), yup.string(), yup.string())
public async activateAccount(orgId: string, userId: string, tokenId ?:
string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT_STATUS.activateAccount
", "TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  let tokenAccount = await this.Ctx.ERC1155Account.getAccount(userAccountId,
tokenId);
  return await
this.Ctx.ERC1155Account.activateAccount(tokenAccount.accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

`suspendAccount`
This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
@Validator(yup.string(), yup.string(), yup.string())
public async suspendAccount(orgId: string, userId: string, tokenId ?: string)
{
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT_STATUS.suspendAccount"
, "TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  let tokenAccount = await this.Ctx.ERC1155Account.getAccount(userAccountId,
tokenId);
  return await this.Ctx.ERC1155Account.suspendAccount(tokenAccount.accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "status": "suspended"
}
```

deleteAccount
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance must be zero.

```
@Validator(yup.string(), yup.string(), yup.string())
public async deleteAccount(orgId: string, userId: string, tokenId ?: string) {
  await
this.Ctx.ERC1155Auth.checkAuthorization("ERC1155ACCOUNT_STATUS.deleteAccount",
 "TOKEN");
  const userAccountId = this.Ctx.ERC1155Account.generateAccountId(orgId,
userId, ACCOUNT_TYPE.USER_ACCOUNT);
  let tokenAccount = await this.Ctx.ERC1155Account.getAccount(userAccountId,
tokenId);
  return await this.Ctx.ERC1155Account.deleteAccount(tokenAccount.accountId);
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ?: string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Account Status SDK Methods

getAccountStatus
This method gets the current status of the token account.

```
Ctx.ERC1155AccountStatus.getAccountStatus(accountId: string)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

`getAccountStatusHistory`
This method gets the history of the account status.

```
Ctx.ERC1155AccountStatus.history(statusId: string)
```

Parameters:

- `statusId: string` – The ID of the account status object.

Returns:

- On success, a JSON representation of the account status history.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2022-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "timeStamp": "2022-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

`activateAccount`
This method activates a token account. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
Ctx.ERC1155Account.activateAccount(tokenAccountId: string)
```

Parameters:

- `tokenAccountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

`suspendAccount`
This method suspends a token account.

```
Ctx.ERC1155Account.suspendAccount(tokenAccountId: string)
```

Parameters:

- `tokenAccountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

`deleteAccount`
This method deletes a token account.

```
Ctx.ERC1155Account.deleteAccount(tokenAccountId: string)
```

Parameters:

- `tokenAccountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "statusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "accountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

Scaffolded Go Token Project for ERC-1155

Blockchain App Builder takes the input from your token specification file and generates a fully-functional scaffolded chaincode project.

The project automatically generates token lifecycle classes and functions, including CRUD and non-CRUD methods. Validation of arguments, marshalling/unmarshalling, and transparent persistence capability are all supported automatically.

For information on the scaffolded project and methods that are not directly related to tokens, see Scaffolded Go Chaincode Project.

Reference:

- Model
- Controller
  - Automatically Generated Token Methods

Copyright © 2025, Oracle and/or its affiliates.

- SDK Methods

Model

Transparent Persistence Capability, or simplified ORM, is captured in the `OchainModel` class. The following model shows a whole non-fungible token.

```
package model

type ArtCollection struct {
    AssetType     string `json:"AssetType" final:"otoken"`
    TokenId       string `json:"TokenId" id:"true" mandatory:"true"
validate:"regexp=^[A-Za-z0-9][A-Za-z0-9_-]*$,max=16"`
    TokenName     string `json:"TokenName" final:"artcollection"`
    TokenDesc     string `json:"TokenDesc" validate:"max=256"`
    TokenStandard string `json:"TokenStandard" final:"erc1155+"`
    TokenType     string `json:"TokenType" final:"nonfungible"
validate:"regexp=^nonfungible$"`
    TokenUnit     string `json:"TokenUnit" final:"whole"
validate:"regexp=^whole$"`

    Mintable map[string]interface{} `json:"Mintable"
final:"{\"Max_mint_quantity\":20000}"`

    Behaviors []string `json:"Behaviors"
final:"[\"indivisible\",\"singleton\",\"mintable\",\"transferable\",\"burnable
\",\"roles\"]"`

    Roles map[string]interface{} `json:"Roles"
final:"{\"minter_role_name\":\"minter\"}"`

    Owner           string `json:"Owner,omitempty" validate:"string"`
    CreatedBy       string `json:"CreatedBy,omitempty" validate:"string"`
    TransferredBy   string `json:"TransferredBy,omitempty" validate:"string"`
    CreationDate    string `json:"CreationDate,omitempty" validate:"string"`
    TransferredDate string `json:"TransferredDate,omitempty"
validate:"string"`
    IsBurned        bool   `json:"IsBurned" validate:"bool"`
    BurnedBy        string `json:"BurnedBy,omitempty" validate:"string"`
    BurnedDate      string `json:"BurnedDate,omitempty" validate:"string"`
    TokenUri        string `json:"TokenUri" mandatory:"true"
validate:"string,max=2000"`

    TokenMetadata ArtCollectionMetadata `json:"TokenMetadata"`

    Price        int  `json:"Price" validate:"int"`
    On_sale_flag bool `json:"On_sale_flag" validate:"bool"`
}

type ArtCollectionMetadata struct {
    Painting_name string `json:"Painting_name" validate:"string"`
    Description   string `json:"Description" validate:"string"`
    Image         string `json:"Image" validate:"string"`
    Painter_name  string `json:"Painter_name" validate:"string"`
}
```


Copyright © 2025, Oracle and/or its affiliates.

```
type Loyalty struct {
    AssetType     string `json:"AssetType" final:"otoken"`
    TokenId       string `json:"TokenId" id:"true" mandatory:"true"
validate:"regexp=^[A-Za-z0-9][A-Za-z0-9_-]*$,max=16"`
    TokenName     string `json:"TokenName" final:"loyalty"`
    TokenDesc     string `json:"TokenDesc" validate:"max=256"`
    TokenStandard string `json:"TokenStandard" final:"erc1155+"`
    TokenType     string `json:"TokenType" final:"fungible"
validate:"regexp=^fungible$"`
    TokenUnit     string `json:"TokenUnit" final:"fractional"
validate:"regexp=^fractional$"`

    Mintable map[string]interface{} `json:"Mintable"
final:"{\"Max_mint_quantity\":10000}"`

    Divisible map[string]interface{} `json:"Divisible"
final:"{\"Decimal\":2}"`

    Behaviors []string `json:"Behaviors"
final:"[\"divisible\",\"mintable\",\"transferable\",\"burnable\",\"roles\"]"`

    Roles map[string]interface{} `json:"Roles"
final:"{\"minter_role_name\":\"minter\"}"`

    Currency_name           string      `json:"Currency_name"
validate:"string"`
    Token_to_currency_ratio int         `json:"Token_to_currency_ratio"
validate:"int"`
    Metadata                interface{} `json:"Metadata,omitempty"`
}
```

The following model shows a fractional non-fungible token.

```
type RealEstateProperty struct {
      AssetType string `json:"AssetType" final:"otoken"`
      TokenId string `json:"TokenId" id:"true" mandatory:"true"
validate:"regexp=^[A-Za-z0-9][A-Za-z0-9_-]*$,max=16"`
      TokenName string `json:"TokenName" final:"realestateproperty"`
      TokenDesc string `json:"TokenDesc" validate:"max=256"`
      TokenStandard string `json:"TokenStandard" final:"erc1155+"`
      TokenType string `json:"TokenType" final:"nonfungible"
validate:"regexp=^nonfungible$"`
      TokenUnit string `json:"TokenUnit" final:"fractional"
validate:"regexp=^fractional$"`

      Mintable map[string]interface{} `json:"Mintable"
final:"{\"Max_mint_quantity\":0}"`
      Behaviors []string `json:"Behaviors"
final:"[\"divisible\",\"mintable\",\"transferable\",\"roles\"]"`

      Divisible map[string]interface{} `json:"Divisible"
final:"{\"Decimal\":0}"`

      Roles map[string]interface{} `json:"Roles"
final:"{\"minter_role_name\":\"minter\"}"`
```


Copyright © 2025, Oracle and/or its affiliates.

```
      CreatedBy string `json:"CreatedBy,omitempty" validate:"string"`
      CreationDate string `json:"CreationDate,omitempty" validate:"string"`
      IsBurned bool `json:"IsBurned" validate:"bool"`
      TokenUri string `json:"TokenUri" mandatory:"true"
validate:"string,max=2000"`
      Quantity float64 `json:"Quantity,omitempty"`
      TokenMetadata RealEstatePropertyMetadata `json:"TokenMetadata"`

      PropertySellingPrice int `json:"PropertySellingPrice" validate:"int"`
      PropertyRentingPrice int `json:"PropertyRentingPrice" validate:"int"`
}

type RealEstatePropertyMetadata struct {
      PropertyType string `json:"PropertyType" validate:"string"`
      PropertyName string `json:"PropertyName" validate:"string"`
      PropertyAddress string `json:"PropertyAddress" validate:"string"`
      PropertyImage string `json:"PropertyImage" validate:"string"`
}
```

Controller

There is only one main controller.

```
type Controller struct {
    Ctx trxcontext.TrxContext
}
```

You can create any number of classes, functions, or files, but only those methods that are defined in the main controller class are invokable. The other methods are hidden.

You can use the token SDK methods to write custom methods for your business application.

Automatically Generated Token Methods

Blockchain App Builder automatically generates methods to support tokens and token life cycles. You can use these methods to initialize tokens, manage roles and accounts, and complete other token lifecycle tasks without any additional coding. Controller methods must be public to be invokable. Public method names begin with an upper case character. Method names that begin with a lower case character are private.

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

Copyright © 2025, Oracle and/or its affiliates.

IsTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`.This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) IsTokenAdmin(orgId string, userId string) (interface{},
error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Admin.IsUserTokenAdmin",
"TOKEN", map[string]string{"orgId": orgId, "userId": userId})
            if err != nil || !auth {
                  return false, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Auth.IsUserTokenAdmin(orgId, userId)
      }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

Return Value Example:

```
{"result": true}
```

AddTokenAdmin
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenAdmin(orgId string, userId string) (interface{},
error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Admin.AddAdmin", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Admin.AddAdmin(orgId, userId)
      }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value

```
{"msg":"Successfully added Admin (orgId: appDev, userId: user1)"}
```

`RemoveTokenAdmin`
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode. You cannot remove yourself as a `Token Admin`.

```
func (t *Controller) RemoveTokenAdmin(orgId string, userId string)
(interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Admin.RemoveAdmin", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Admin.RemoveAdmin(orgId, userId)
      }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value

```
{
    "msg": "Successfully removed Admin (orgId appdev userId user1)"
}
```

`GetAllTokenAdmins`
This method returns a list of all users who are a `Token Admin` of the chaincode. This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) GetAllTokenAdmins() (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Admin.GetAllAdmins", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Admin.GetAllAdminUsers()
      }
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- none

Returns:

- On success, an `admins` array in JSON format.

Return Value

```
{
  "admins": [
    {
      "OrgId": "appdev",
      "UserId": "idcqa"
    },
    {
      "OrgId": "appdev",
      "UserId": "user1"
    }
  ]
}
```

Methods for Token Configuration Management

`Init`
This method is called when the chaincode is instantiated. Every `Token Admin` is identified by the `userId` and `orgId` information in the `adminList` parameter. The `userId` is the user name or email ID of the instance owner or the user who is logged in to the instance. The `orgId` is the membership service provider (MSP) ID of the user in the current network organization. The `adminList` parameter is mandatory the first time you deploy the chaincode. If you are upgrading the chaincode, pass an empty list (`[]`). If you are the user who initially deployed the chaincode, you can also specify new admins in the `adminList` parameter when you are upgrading the chaincode. Any other information in the `adminList` parameter is ignored during upgrades.

```
func (t *Controller) Init(adminList []erc1155Admin.ERC1155TokenAdminAsset)
(interface{}, error) {
      list, err := t.Ctx.ERC1155Admin.InitAdmin(adminList)
      if err != nil {
            return nil, fmt.Errorf("initialising admin list failed %s",
err.Error())
      }
      <1st Token Name> := <1st TokenClassName>{}
      _, err = t.Ctx.ERC1155Token.SaveClassInfo(&<1st Token Name>)
      if err != nil {
            return nil, err
      }
         .
         .
      <nth Token Name> := <nth TokenClassName>{}
      _, err = t.Ctx.ERC1155Token.SaveClassInfo(&<nth Token Name>)
      if err != nil {
            return nil, err
```


Copyright © 2025, Oracle and/or its affiliates.

```
      }
      _, err = t.Ctx.ERC1155Token.SaveDeleteTransactionInfo()
      if err != nil {
            fmt.Println("error: ", err)
      }
      return list, err
}
```

Parameters:

- `adminList array` – An array of `{OrgId, UserId}` information that specifies the list of token admins. The `adminList` array is a mandatory parameter.

`Create<Token Name>Token`
This method creates tokens. Every token that is defined has its own create method. For fungible tokens, this method can be called only by a `Token Admin` of the chaincode. For non-fungible tokens, if the minter role is defined in the specification file, any user with the minter role can call this method to create an NFT. If the minter role is not defined, any user can use this method to create (mint) NFTs. The user who calls this method becomes the owner of the NFT.

Fungible Tokens:

```
func (t *Controller) Create<Token Name>Token(tokenAsset <Token Class>)
(interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.Save", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Token.Save(&tokenAsset)
    }
```

Non-Fungible Tokens:

```
func (t *Controller) Create<Token Name>Token(tokenAsset <Token Class>,
quantity float64) (interface{}, error) {
         quantityToPass := []float64{quantity}
          return t.Ctx.ERC1155Token.Save(&tokenAsset, quantityToPass...)
    }
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset. The properties of the asset are defined in the model file.
- `quantity: number` – For non-fungible tokens only, the number of tokens to mint. The only supported value for this parameter is `1`.

Returns:

- On success, the token asset in JSON format, which includes the following information, depending on the token type.

Copyright © 2025, Oracle and/or its affiliates.

- `Behaviors` – A list of token behaviors. This property cannot be edited.
- `CreatedBy` – The account ID of the caller, who is the user minting the token. This property cannot be edited.
- `CreationDate` – The time stamp of the minting transaction. This property cannot be edited.
- `IsBurned` – This property indicates whether the token is burned. This property cannot be edited.
- `Mintable` – The properties related to minting. The `max_mint_quantity` value defines the maximum number of tokens that can be created for the token class.
- `Owner` – The account ID of the current owner, who is the caller of the method.
- `Symbol` – The symbol of the token. This property cannot be edited.
- `TokenDesc` – The description of the token.
- `TokenMetadata` – JSON information that describes the token.
- `TokenName` – The name of the token. This property cannot be edited.
- `TokenStandard` – The standard of the token. This property cannot be edited.
- `TokenType` – The type of the token (fungible or non-fungible). This property cannot be edited.
- `TokenUnit` – The unit of the token (whole or fractional). This property cannot be edited.
- `TokenUri` – The URI of the token.
- `Quantity` – The quantity of the token.

Return Value Example (Whole NFT):


```
{
    "AssetType": "otoken",
    "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2022-12-29T09:57:03+05:30",
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 500
    },
    "OnSaleFlag": false,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Price": 100,
    "Quantity": 1,
    "Roles": {
        "burner_role_name": "burner",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "minter_role_name": "minter"
    },
    "TokenDesc": "token description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "artcollection",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
}
```

Return Value Example (Fungible Token):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "Currency_name": "Dollar",
    "Divisible": {
        "Decimal": 2
    },
    "Mintable": {
        "Max_mint_quantity": 10000
    },
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "Loyalty",
    "TokenName": "loyalty",
    "TokenStandard": "erc1155+",
    "TokenType": "fungible",
    "TokenUnit": "fractional",
    "Token_to_currency_ratio": 0
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fractional NFT):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2023-06-14T09:53:53+05:30",
    "Divisible": {
        "Decimal": 2
    },
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "On_sale_flag": false,
    "Price": 1000,
    "Quantity": 100,
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "realEstate",
    "TokenMetadata": {
        "Description": "Painting Description",
        "Image": "",
        "Painter_name": "",
        "Painting_name": "Paint"
    },
    "TokenName": "realestate",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "fractional",
    "TokenUri": "www.realestate.example.com"
}
```

`Update<Token Name>Token`
This method updates tokens. Every token that is defined has its own update method. You cannot update token metadata or the token URI of non-fungible tokens. For fungible tokens, this method can be called only by a `Token Admin` of the chaincode. For non-fungible tokens, this method can be called only by the token owner.

Fungible Tokens:

```
func (t *Controller) Update<%=tokenModelName%>Token(tokenAsset
<%=tokenModelName%>) (interface{}, error) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.Update", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Token.Update(&tokenAsset)
    }
```

Non-Fungible Tokens:

```
func (t *Controller) Update<%=tokenModelName%>Token(tokenAsset
<%=tokenModelName%>) (interface{}, error) {
            return t.Ctx.ERC1155Token.Update(&tokenAsset)
    }
```

Parameters:

- `tokenAsset: <Token Class>` – The token asset. The properties of the asset are defined in the model file.

Returns:

- On success, the updated token asset in JSON format.

Return Value Example (Whole NFT):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2022-12-29T09:57:03+05:30",
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 500
    },
    "OnSaleFlag": false,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Price": 100,
    "Quantity": 1,
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "token description",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "artcollection",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
}
```

GetTokenHistory
This method returns the history for a specified token ID. Anyone can call this method.

```
func (t *Controller) GetTokenHistory(tokenId string) (interface{}, error) {
            return t.Ctx.ERC1155Token.GetTokenHistory(tokenId)
      }
```

Parameters:

- `tokenId: string` – The ID of the token.

Returns:

- On success, a JSON array that contains the token history.

Return Value Example (Fungible Token):

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-08T09:54:11Z",
    "TxId":
"823sa7c7a00941c62285c86f922bc4d3f5326a20f4bf2f82daa5bc661e4682e8",
    "Value": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "Rupees",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "Updated Token Description",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-08T09:54:11Z",
    "TxId":
"711bb7c7a00941c62285c86f922bc3b3f5326a20f4bf2f82daa5bc661e4682e8",
    "Value": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "Dollar",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  }
]
```


Return Value Example (Fractional NFT):


```
[
    {
        "Timestamp": "2023-06-20T01:06:33Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "TrxId":
"16e53db4f8107f9634b7c3a0a2a81a00f69b634f2a52902b809e544d07f272b1",
        "Value": {
            "AssetType": "otoken",
            "Behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:02:27Z",
            "Divisible": {
                "Decimal": 2
            },
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "On_sale_flag": true,
            "Owners": [
                {
                    "AccountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                    "TokenShare": 10
                },
                {
                    "AccountId":
"oaccount~3cddfdaa855900579d963aa6f755a4aed1f3a474a2462c1b45bd7f36df673224",
                    "TokenShare": 10
                }
            ],
            "Price": 2000,
            "Quantity": 20,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "FNFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "realestate",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "fractional",
            "TokenUri": "www.FNFT.example.com"
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "Timestamp": "2023-06-20T01:02:27Z",
        "TrxId":
"cec80910d087682554048f911d2cf98b66382bbcf1615483fa1c96c7ea08077c",
        "Value": {
            "AssetType": "otoken",
            "Behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:02:27Z",
            "Divisible": {
                "Decimal": 2
            },
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "On_sale_flag": true,
            "Owners": [
                {
                    "AccountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                    "TokenShare": 20
                }
            ],
            "Price": 2000,
            "Quantity": 20,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "FNFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "realestate",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "fractional",
            "TokenUri": "www.FNFT.example.com"
        }
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Whole NFT):


```
[
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-20T01:15:56Z",
        "TxId":
"89a3df3ebbe6dca2bcfbd51fc7dca9aab818a2af746b79a92dc8155b729ab22d",
        "Value": {
            "AssetType": "otoken",
            "Behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:15:56Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "On_sale_flag": true,
            "Owner":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "Price": 2000,
            "Quantity": 1,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "Updated Token Description",
            "TokenId": "NFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "artcollection",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": "www.NFT.example.com"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-20T01:15:56Z",
        "TxId":
"90d6af3ebbe6dca2bcfbd51fc7dca9aab818a2af746b79a92dc8155b729ab22d",
        "Value": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "AssetType": "otoken",
            "Behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:15:56Z",
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "On_sale_flag": true,
            "Owner":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "Price": 2000,
            "Quantity": 1,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "NFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "artcollection",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": "www.NFT.example.com"
        }
    }
]
```

GetAllTokens

This method returns all of the token assets that are saved in the state database. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAllTokens() (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.GetAllTokens", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
```


Copyright © 2025, Oracle and/or its affiliates.

```
            }
            return t.Ctx.ERC1155Token.GetAllTokens()
        }
```

Parameters:

- none

Returns:

- A list of all token assets in JSON format.

Return Value

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenTwo",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "art",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
      ],
      "BurnedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "BurnedDate": "2022-12-08T10:49:37Z",
      "CreatedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "CreationDate": "2022-12-08T10:45:10Z",
      "IsBurned": true,
      "Mintable": {
        "Max_mint_quantity": 20000
      },
      "OnSaleFlag": false,
      "Owner": "",
      "Price": 0,
      "Roles": {
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "art",
      "TokenMetadata": {
        "Description": "",
        "Image": "",
        "PainterName": "",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "PaintingName": ""
      },
      "TokenName": "artcollection",
      "TokenStandard": "erc1155+",
      "TokenType": "nonfungible",
      "TokenUnit": "whole",
      "TokenUri": "art.example.com",
      "TransferredBy":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
      "TransferredDate": "2022-12-08T10:47:04Z"
    }
  },
  {
    "key": "FNFT",
    "valueJson": {
        "AssetType": "otoken",
        "Behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
        "CreationDate": "2023-06-20T01:02:27Z",
        "Divisible": {
            "Decimal": 2
        },
        "IsBurned": false,
        "Mintable": {
            "Max_mint_quantity": 20000
        },
        "On_sale_flag": true,
        "Price": 2000,
        "Quantity": 20,
        "Roles": {
            "burner_role_name": "burner",
            "minter_role_name": "minter"
        },
        "TokenDesc": "",
        "TokenId": "FNFT",
        "TokenMetadata": {
            "Description": "",
            "Image": "",
            "Painter_name": "",
            "Painting_name": ""
        },
        "TokenName": "realestate",
        "TokenStandard": "erc1155+",
        "TokenType": "nonfungible",
        "TokenUnit": "fractional",
        "TokenUri": "www.FNFT.example.com"
    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
  },
  {
    "key": "FNFT",
    "valueJson": {
        "AssetType": "otoken",
        "Behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
        "CreationDate": "2023-06-20T01:02:27Z",
        "Divisible": {
            "Decimal": 2
        },
        "IsBurned": false,
        "Mintable": {
            "Max_mint_quantity": 20000
        },
        "On_sale_flag": true,
        "Price": 2000,
        "Quantity": 20,
        "Roles": {
            "burner_role_name": "burner",
            "minter_role_name": "minter"
        },
        "TokenDesc": "",
        "TokenId": "FNFT",
        "TokenMetadata": {
            "Description": "",
            "Image": "",
            "Painter_name": "",
            "Painting_name": ""
        },
        "TokenName": "realestate",
        "TokenStandard": "erc1155+",
        "TokenType": "nonfungible",
        "TokenUnit": "fractional",
        "TokenUri": "www.FNFT.example.com"
    }
  }
]
```

`GetTokenById`

This method returns a token object if the token is present in the state database. For fractional NFTs, the list of owners is also returned. This method can be called only by a `Token Admin` of the chaincode or the token owner.

```
func (t *Controller) GetTokenById(tokenId string) (interface{}, error) {
            auth, err :=
```


Copyright © 2025, Oracle and/or its affiliates.

```
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.Get", "TOKEN",
map[string]string{"tokenId": tokenId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Token.GetTokenById(tokenId)
      }
```

Parameters:

- `tokenId string` – The ID of the token to get.

Return Value Example (Whole NFT):

```
{
  "AssetType": "otoken",
  "Behaviors": [
    "indivisible",
    "singleton",
    "mintable",
    "transferable",
    "burnable",
    "roles"
  ],
  "CreatedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "CreationDate": "2022-12-08T10:55:29Z",
  "IsBurned": false,
  "Mintable": {
    "Max_mint_quantity": 20000
  },
  "OnSaleFlag": false,
  "Owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "Price": 0,
  "Quantity": 1,
  "Roles": {
    "minter_role_name": "minter"
  },
  "TokenDesc": "",
  "TokenId": "nftToken",
  "TokenMetadata": {
    "Description": "",
    "Image": "",
    "PainterName": "",
    "PaintingName": ""
  },
  "TokenName": "artcollection",
  "TokenStandard": "erc1155+",
  "TokenType": "nonfungible",
  "TokenUnit": "whole",
  "TokenUri": "nftToken.example.com"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fungible Token):


```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "Currency_name": "Dollar",
    "Divisible": {
        "Decimal": 2
    },
    "Mintable": {
        "Max_mint_quantity": 10000
    },
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "Loyalty",
    "TokenName": "loyalty",
    "TokenStandard": "erc1155+",
    "TokenType": "fungible",
    "TokenUnit": "fractional",
    "Token_to_currency_ratio": 0
}
```


Return Value Example (Fractional NFT):


```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2023-06-14T09:53:53+05:30",
    "Divisible": {
        "Decimal": 2
    },
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "On_sale_flag": false,
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "Owners": [
        {
            "AccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
            "TokenShare": 100
        }
    ],
    "Price": 1000,
    "Quantity": 100,
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "realEstate",
    "TokenMetadata": {
        "Description": "Painting Description",
        "Image": "",
        "Painter_name": "",
        "Painting_name": "Paint"
    },
    "TokenName": "realestate",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "fractional",
    "TokenUri": "www.realestate.example.com"
}
```

GetAllTokensByUser

This method returns all of the token assets that are owned by a specified user. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) GetAllTokensByUser(orgId string, userId string)
(interface{}, error) {
            accountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in GetAllTokensByUser. Error:
%s", err)
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.GetAllTokensByUser",
"TOKEN", map[string]string{"accountId": accountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Token.GetAllTokensByUser(accountId)
      }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Return Value


```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "nftToken",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
      ],
      "CreatedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "CreationDate": "2022-12-08T10:55:29Z",
      "IsBurned": false,
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "Mintable": {
        "Max_mint_quantity": 20000
      },
      "OnSaleFlag": false,
      "Owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "Price": 0,
      "Quantity": 1,
      "Roles": {
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "nftToken",
      "TokenMetadata": {
        "Description": "",
        "Image": "",
        "PainterName": "",
        "PaintingName": ""
      },
      "TokenName": "artcollection",
      "TokenStandard": "erc1155+",
      "TokenType": "nonfungible",
      "TokenUnit": "whole",
      "TokenUri": "example.com"
    }
  }
]
```

OwnerOf
This method returns the account ID, organization ID, and user ID of the owner of the specified token ID. Anyone can call this method.

```
func (t *Controller) OwnerOf(tokenId string) (interface{}, error) {
            return t.Ctx.ERC1155Token.OwnerOf(tokenId)
      }
```

Parameters:

- `tokenId string` – The ID of the token.

Return Value Example (Whole NFT):

```
{
  "AccountId":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "OrgId": "appdev",
  "UserId": "idcqa"
}
```

Return Value Example (Fractional NFT):

```
[
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "AccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "OrgId": "Org1MSP",
        "UserId": "admin"
    },
    {
        "AccountId":
"oaccount~74108eca702bab6d8548e740254f2cc7955d886885251d52d065042172a59db0",
        "OrgId": "Org1MSP",
        "UserId": "user"
    }
]
```

URI
This method returns the URI of a specified token. Anyone can call this method.

```
func (t *Controller) URI(tokenId string) (interface{}, error) {
            return t.Ctx.ERC1155Token.TokenURI(tokenId)
      }
```

Parameters:

- `tokenId string` – The ID of the token.

Return Value

```
{
    "TokenUri": "example.com"
}
```

Name
This method returns the name of the token class. Anyone can call this method.

```
func (t *Controller) Name(tokenId string) (interface{}, error) {
            return t.Ctx.ERC1155Token.Name(tokenId)
      }
```

Parameters:

- `tokenId string` – The ID of the token.

Return Value

```
{"TokenName": "artcollection"}
```


Copyright © 2025, Oracle and/or its affiliates.

TotalSupply
This method returns the total number of minted tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) TotalSupply(tokenDetail erc1155Role.TokenDetail)
(interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.TotalSupply", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
            if err != nil {
                  return nil, err
            }
            return t.Ctx.ERC1155Token.TotalSupply(token)
      }
```

Parameters:

- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"TokenName":"artCollection"}
  ```

Return Value

```
{
    "TotalSupply": 100
}
```

TotalNetSupply
This method returns the total number of minted tokens minus the number of burned tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) TotalNetSupply(tokenDetail erc1155Role.TokenDetail)
(interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.TotalNetSupply", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
```


Copyright © 2025, Oracle and/or its affiliates.

```
            if err != nil {
                  return nil, err
            }
            return t.Ctx.ERC1155Token.TotalNetSupply(token)
      }
```

Parameters:

- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"TokenName":"artCollection"}
  ```

Return Value

```
{
   "TotalNetSupply": 900
}
```

`GetTokensByName`
This method returns all of the token assets for a specified token name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetTokensByName(tokenName string) (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.GetTokensByName", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Token.GetTokensByName(tokenName)
      }
```

Parameters:

- `tokenName: string` – The name of the token.

Return Value

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenTwo",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
```


Copyright © 2025, Oracle and/or its affiliates.

```
  }
]
```

GetTokenDecimal
This method returns the number of decimal places for a specified token. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetTokenDecimal(tokenId string) (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.GetTokenDecimal", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            tokenDecimal, err := t.Ctx.ERC1155Token.GetDecimals(tokenId)
            if err != nil {
                  return nil, fmt.Errorf("error in GetTokenDecimal: %s",
err.Error())
            }
            response := make(map[string]interface{})
            response["msg"] = fmt.Sprintf("Token Id: %s has %d decimal
places.", tokenId, tokenDecimal)
            return response, nil
      }
```

Parameters:

- `tokenId string` – The ID of the token.

Return Value

```
{
    "msg": "Token Id: tokenOne has 2 decimal places."
}
```

Methods for Account Management

CreateAccount
This method creates an account for a specified user and associated token accounts for fungible or non-fungible tokens. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a user has. Users must have accounts in the network to complete token-related operations. This method can be called only by a `Token Admin` of the chaincode.
A user account has a unique ID, which is formed by an SHA-256 hash of the `orgId` parameter and the `userId` parameter.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this account.

Copyright © 2025, Oracle and/or its affiliates.

User account IDs start with with `ouaccount~`. Token account IDs start with `oaccount~`.

```
func (t *Controller) CreateAccount(orgId string, userId string, ftAccount
bool, nftAccount bool) (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.CreateAccount", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Account.CreateAccount(orgId, userId,
ftAccount, nftAccount)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `ftAccount bool` – If true, a fungible token account is created and associated with the user account.
- `nftAccount bool` – If true, a non-fungible token account is created and associated with the user account.

Returns:

- On success, a JSON object of the account that was created.

Return Value

```
{
  "AssetType": "ouaccount",
  "AccountId":
"ouaccount~cf20877546f52687f387e7c91d88b9722c97e1a456cc0484f40c747f7804feae",
  "UserId": "user1",
  "OrgId": "appdev",
  "TotalAccounts": 2,
  "TotalFtAccounts": 1,
  "AssociatedFtAccounts": [
    {
      "AccountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
      "TokenId": ""
    }
  ],
  "AssociatedNftAccount":
"oaccount~73c3e835dac6d0a56ca9d8def08269f83cefd59b9d297fe2cdc5a9083828fa58"
}
```

`CreateUserAccount`
This method creates an account for a specified user. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible

Copyright © 2025, Oracle and/or its affiliates.

token accounts that a user has. Users must have accounts in the network to complete token-related operations.
An account ID is an SHA-256 hash of the `orgId` parameter and the `userId` parameter. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) CreateUserAccount(orgId string, userId string)
(interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.CreateUserAccount",
"TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Account.CreateUserAccount(orgId, userId)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- On success, a JSON object of the user account that was created.

Return Value

```
{
  "AssetType": "ouaccount",
  "AccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "UserId": "idcqa",
  "OrgId": "appdev",
  "TotalAccounts": 0,
  "TotalFtAccounts": 0,
  "AssociatedFtAccounts": [],
  "AssociatedNftAccount": ""
}
```

`CreateTokenAccount`
This method creates a fungible or non-fungible token account to associate with a user account.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this account.

Copyright © 2025, Oracle and/or its affiliates.

This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) CreateTokenAccount(orgId string, userId string,
tokenType erc1155Token.TokenType) (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.CreateTokenAccount",
"TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            typeOfToken, err := tokenType.GetTokenType()
          if err != nil {
                  return nil, err
            }
            return t.Ctx.ERC1155Account.CreateTokenAccount(orgId, userId,
typeOfToken)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenType erc1155Token.TokenType` – The type of token account to create. The only supported token types are `nonfungible` and `fungible`.

Returns:

- On success, a JSON object of the token account that was created.

Return Value

```
{
    "AssetType": "ouaccount",
    "AccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
    "UserId": "idcqa",
    "OrgId": "appdev",
    "TotalAccounts": 1,
    "TotalFtAccounts": 1,
    "AssociatedFtAccounts": [
        {
            "AccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "TokenId": ""
        }
    ],
    "AssociatedNftAccount": ""
}
```

`AssociateFungibleTokenAccount`
This method associates a user's fungible token account to a particular fungible token.

Copyright © 2025, Oracle and/or its affiliates.

This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) AssociateFungibleTokenToAccount(orgId string, userId
string, tokenId string) (interface{}, error) {
            accountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in generating accountId.
Error: %s", err)
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.AssociateFungibleTokenToA
ccount", "TOKEN", map[string]string{"accountId": accountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Account.AssociateTokenToToken(accountId,
tokenId)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenId string` – The ID of the token.

Returns:

- On success, a JSON object of the user account, which shows that the fungible token was associated to the token account. For example, in the following example, the first object in the `associatedFtAccounts` array shows that the fungible token account ID and the token ID are associated.

Return Value

```
{
    "AssetType": "ouaccount",
    "AccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
    "UserId": "idcqa",
    "OrgId": "appdev",
    "TotalAccounts": 1,
    "TotalFtAccounts": 1,
    "AssociatedFtAccounts": [
        {
            "AccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "TokenId": "tokenOne"
        }
    ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "AssociatedNftAccount": ""
}
```

GetAccountHistory
This method returns history for a specified token account. This is an asynchronous method. This method can be called only by the `Token Admin` of the chaincode or by the account owner.

```
func (t *Controller) GetAccountHistory(orgId string, userId string,
tokenId ...string) (interface{}, error) {
            userAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(orgId, userId, constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in GetAccountHistory. Error:
%s", err)
            }
            _, err =
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.History", "TOKEN",
map[string]string{"accountId": userAccountId})
            if err != nil {
                  return nil, fmt.Errorf("error in GetAccountHistory. Error:
%s", err)
            }
            tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId,
tokenId...)
            if err != nil {
                  return nil, fmt.Errorf("error in GetAccountHistory. Error:
%s", err)
            }
            tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
            if err != nil {
                  return nil, fmt.Errorf("error in GetAccountHistory. Error:
%s", err)
            }
            return t.Ctx.ERC1155Account.GetAccountHistory(tokenAccountId)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, an array of JSON objects that describes the account history.

Return Value

```
[
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "IsDelete": "false",
        "Timestamp": "2023-06-06T11:03:48Z",
        "TxId":
"c5180f3be3d9130f25a4b4e866f38a4283117dcbfbffb4f55e2c5b03dba0dd29",
        "Value": {
            "AccountCategory": "",
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "AssetType": "oaccount",
            "Balance": 100,
            "BapAccountVersion": 1
            "OrgId": "appdev",
            "TokenId": "loy1",
            "TokenName": "loyalty",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-06T11:02:39Z",
        "TxId":
"6f81b0c94b451d375a3892446aefbdf78d9fd1ac43699daa89f0dff10db5fd22",
        "Value": {
            "AccountCategory": "",
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "AssetType": "oaccount",
            "Balance": 0,
            "BapAccountVersion": 0
            "OrgId": "appdev",
            "TokenId": "loy1",
            "TokenName": "loyalty",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-05T16:28:46Z",
        "TxId":
"8185af648546e909488e72149be497b210f74f95ada252c42da9c35cb9d98691",
        "Value": {
            "AccountCategory": "",
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "AssetType": "oaccount",
            "Balance": 0,
            "BapAccountVersion": 0
            "OrgId": "appdev",
            "TokenId": "",
            "TokenName": "",
            "TokenType": "fungible",
            "UserId": "idcqa"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    }
]
```

GetAccount
This method returns token account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
func (t *Controller) GetAccount(orgId string, userId string,
tokenId ...string) (interface{}, error) {
            accountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
            if err != nil {
                  return nil, err
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.GetAccount", "TOKEN",
map[string]string{"accountId": accountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Account.GetAccountWithStatus(accountId,
tokenId...)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON object that includes token account details. The `BapAccountVersion` and `AccountCategory` parameters are defined in the account object for internal use.

Return Value Example (Non-Fungible Token Account):

```
{
  "AccountId":
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
  "AssetType": "oaccount",
  "BapAccountVersion": 1,
  "NoOfNfts": 1,
  "OrgId": "appdev",
  "Status": "active",
  "TokenType": "nonfungible",
  "UserId": "idcqa"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fungible Token Account):

```
{
  "AccountCategory": "",
  "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
  "AssetType": "oaccount",
  "Balance": 0,
  "BapAccountVersion": 0,
  "OrgId": "appdev",
  "Status": "active",
  "TokenId": "t1",
  "TokenName": "loyalty",
  "TokenType": "fungible",
  "UserId": "idcqa"
}
```

`GetAllAccounts`
This method returns details of all user accounts. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetAllAccounts() (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.GetAllAccounts", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Account.GetAllAccounts()
      }
```

Parameters:

- none

Returns:

- On success, a JSON array of all accounts.

Return Value

```
[
    {
        "AssetType": "ouaccount",
        "AccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "UserId": "idcqa",
        "OrgId": "appdev",
        "TotalAccounts": 3,
        "TotalFtAccounts": 2,
        "AssociatedFtAccounts": [
            {
                "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "TokenId": "loy1"
            },
            {
                "AccountId":
"oaccount~58c5a6b09a41befca2a9ea2550439838c4dcf4d8a2a4f7c98e9319cf8479bfc4",
                "TokenId": ""
            }
        ],
        "AssociatedNftAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371"
    },
    {
        "AssetType": "ouaccount",
        "AccountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
        "UserId": "user1_minter",
        "OrgId": "appdev",
        "TotalAccounts": 2,
        "TotalFtAccounts": 1,
        "AssociatedFtAccounts": [
            {
                "AccountId":
"oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c",
                "TokenId": "loy1"
            }
        ],
        "AssociatedNftAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446"
    },
]
```

GetAccountDetailsByUser

This method returns an account summary for a specified user and details of fungible and non-fungible tokens that are associated with the user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
func (t *Controller) GetAccountDetailsByUser(orgId string, userId string)
(interface{}, error) {
            accountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
            if err != nil {
                  return nil, err
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.GetAccountDetailsByUser",
 "TOKEN", map[string]string{"accountId": accountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return t.Ctx.ERC1155Account.GetAccountDetailsByUser(orgId, userId)
      }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes and account summary for the specified user and details of fungible and non-fungible tokens that are associated with the user. For fractional non-fungible tokens, the `TokenShare` property in the `AssociatedNFTs` section shows the share that the user owns.

Return Value

```
{
    "AssociatedFTAccounts": [
        {
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "Balance": 90,
          "TokenId": "FT"
        },
    ],
    "AssociatedNFTAccount": {
        "AccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "AssociatedNFTs": [
            {
              "NFTTokenId": "FNFT",
                "TokenShare": 230
            },
            {
              "NFTTokenId": "NFT"
            },
            {
              "NFTTokenId": "NFT2"
            }
        ]
    },
    "UserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
}
```

`GetUserByAccountId`
This method returns the user details of a specified account ID. This method can be called by any user.

```
func (t *Controller) GetUserByAccountId(accountId string) (interface{},
error) {
            return t.Ctx.ERC1155Account.GetUserByAccountById(accountId)
      }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `accountId string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`orgId` and `userId`).

Return Value

```
{
    "OrgId": "appdev",
    "UserId": "user2"
}
```

Methods for Role Management

`AddRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. The specified user must have a token account that is associated with the fungible token, or a non-fungible token account for NFT roles. The specified role must exist in the specification file for the token.

```
func (t *Controller) AddRole(orgId string, userId string, role string,
tokenDetail erc1155Role.TokenDetail) (interface{}, error) {
            userAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(orgId, userId, constants.UserAccount)
            if err != nil {
                  return nil, err
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.AddRoleMember", "TOKEN",
map[string]string{"accountId": userAccountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
            if err != nil {
                  return nil, err
            }
            return t.Ctx.ERC1155Token.AddRoleMember(role, userAccountId,
token)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `role: string` – The name of the role to add to the specified user.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"TokenName":"artCollection"}
  ```

Returns:

- On success, a message with account details.

Return Value

```
{
    "msg": "Successfully added role minter to
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(orgId : appdev, userId : idcqa)"
}
```

IsInRole

This method returns a Boolean value to indicate if a user has a specified role. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by the `Token Admin` of the chaincode or the `Account Owner` of the account. The specified user must have a token account that is associated with the fungible token, or a non-fungible token account for NFT roles. The specified role must exist in the specification file for the token.

```
func (t *Controller) IsInRole(orgId string, userId string, role string,
tokenDetail erc1155Role.TokenDetail) (interface{}, error) {
            userAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(orgId, userId, constants.UserAccount)
            if err != nil {
                  return nil, err
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.IsInRole", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
            if err != nil {
                  return nil, err
            }
            var result bool
            result, err = t.Ctx.ERC1155Token.IsInRole(role, userAccountId,
token)
            if err != nil {
                  return nil, fmt.Errorf("error in IsInRole %s", err.Error())
            }
```


Copyright © 2025, Oracle and/or its affiliates.

```
            response := make(map[string]interface{})
            response["result"] = result
            return response, nil
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `role: string` – The name of the role to search for.
- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"TokenName":"artCollection"}
  ```

Return Value

```
{
    "result": true
}
```

RemoveRole

This method removes a role from a specified user and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by the `Token Admin` of the chaincode. The specified user must have a token account that is associated with the fungible token, or a non-fungible token account for NFT roles. The specified role must exist in the specification file for the token.

```
func (t *Controller) RemoveRole(orgId string, userId string, role string,
tokenDetail erc1155Role.TokenDetail) (interface{}, error) {
            userAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(orgId, userId, constants.UserAccount)
            if err != nil {
                  return nil, err
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.RemoveRoleMember", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
            if err != nil {
                  return nil, err
            }
```


Copyright © 2025, Oracle and/or its affiliates.

```
            return t.Ctx.ERC1155Token.RemoveRoleMember(role, userAccountId,
token)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `role: string` – The name of the role to remove from the specified user.
- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"TokenName":"artCollection"}
  ```

Return Value

```
{
    "msg": "Successfully removed role 'minter' from Account Id:
oaccount~ec7e4de2f81e3ea071710e07b6ff7d9346e84ef665ca4650885dbe8c3e2bd4c0
(Org-Id: appdev, User-Id: idcqa)"
}
```

`GetAccountsByRole`
This method returns a list of all account IDs for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) GetAccountsByRole(role string, tokenDetail
erc1155Role.TokenDetail) (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Role.GetAccountsByRole", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
            if err != nil {
                  return nil, err
            }
            return t.Ctx.ERC1155Role.GetAccountsByRole(role, token)
      }
```

Parameters:

- `role: string` – The name of the role to search for.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```

  For non-fungible tokens, use the following format:

  ```
  {"TokenName":"artCollection"}
  ```

Return Value

```
{
    "accounts": [

"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",

"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b"
    ]
}
```

GetUsersByRole
This method returns a list of all users for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name. This method can be called only by the `Token Admin` of the chaincode.

```
func (t *Controller) GetUsersByRole(role string, tokenDetail
erc1155Role.TokenDetail) (interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Role.GetUsersByRole", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            token, err := t.Ctx.ERC1155Token.GetTokenByIdOrName(tokenDetail)
            if err != nil {
                  return nil, err
            }
            return t.Ctx.ERC1155Role.GetUsersByRole(role, token)
      }
```

Parameters:

- `role: string` – The name of the role to search for.
- `tokenDetails erc1155Role.TokenDetail` – The details that specify the token. For fungible tokens, use the following format:

  ```
  {"TokenId":"token1"}
  ```


Copyright © 2025, Oracle and/or its affiliates.

For non-fungible tokens, use the following format:

```
{"TokenName":"artCollection"}
```

Return Value

```
{
    "Users": [
        {
            "AccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "OrgId": "appdev",
            "UserId": "idcqa"
        },
        {
            "AccountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
            "OrgId": "appdev",
            "UserId": "user1"
        }
    ]
}
```

Methods for Transaction History Management

`GetAccountTransactionHistory`
This method returns account transaction history. This method can be called only by a `Token Admin` of the chaincode or by the account owner. For non-fungible tokens, this method can only be called when connected to the remote Oracle Blockchain Platform network.

```
func (t *Controller) GetAccountTransactionHistory(orgId string, userId
string, tokenId ...string) (interface{}, error) {
            userAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(orgId, userId, constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in
GetAccountTransactionHistory. Error: %s", err)
            }
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.GetAccountTransactionHist
ory", "TOKEN", map[string]string{"accountId": userAccountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId,
tokenId...)
            if err != nil {
                  return nil, fmt.Errorf("error in
GetAccountTransactionHistory. Error: %s", err)
            }
            tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
```


Copyright © 2025, Oracle and/or its affiliates.

```
            if err != nil {
                  return nil, fmt.Errorf("error in
GetAccountTransactionHistory. Error: %s", err)
            }
            return
t.Ctx.ERC1155Account.GetAccountTransactionHistory(tokenAccountId)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Return Value

```
[
    {
        "Balance": 90,
        "Timestamp": "2023-06-06T11:11:09Z",
        "TokenId": "FNFT",
        "TransactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "TransactedAmount": 10,
        "TransactionId":
"otransaction~0f4d96fbf8fed88ea8a313342897772109l467c701848d595ebc3fffa88b3657
~7c88c736df38d5622512f1e8dcdd50710eb47c953f1ecb24ac44790a9e2f475b",
        "TransactionType": "DEBIT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    },
    {
        "Timestamp": "2023-06-06T11:11:09Z",
        "TokenId": "NFT",
        "TransactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "TransactionId":
"otransaction~0f4d96fbf8fed88ea8a313342897772109l467c701848d595ebc3fffa88b3657
~178e3730bc5bee50d02f1464a4eebf733a051905f651e5789039adb4a3edc114",
        "TransactionType": "DEBIT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    },
    {
        "Timestamp": "2023-06-06T11:06:54Z",
        "TokenId": "NFT",
        "TransactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "TransactionId":
"otransaction~6a13667ea3f6edc4c854e85b127526eccb58783f653c348b42a3869f0f29a4fb
~a7cefb22ff39ee7e36967be71de27da6798548c872061a62dabc56d88d50b930",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "TransactionType": "MINT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    },
    {
        "Balance": 100,
        "Timestamp": "2023-06-05T16:34:33Z",
        "TokenId": "FNFT",
        "TransactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "TransactedAmount": 100,
        "TransactionId":
"otransaction~2bc15de1766d582d821bd8d61756bca02837dc683c0aa61f69657ccd1d95e335
~e4eb15d9354f694230df8835ade012100d82aa43672896a2c7125a86e3048f9f",
        "TransactionType": "MINT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    }
]
```

GetTransactionById
This method returns the transaction details for a specified transaction ID. Anyone can call this method.

```
func (t *Controller) GetTransactionById(transactionId string) (interface{},
error) {
            return t.Ctx.ERC1155Transaction.GetTransactionById(transactionId)
      }
```

Parameters:

- `transactionId: string` – The ID of the transaction.

Return Value

```
{
    "history": [
        {
            "IsDelete": "false",
            "Timestamp": "2022-12-08T10:45:56Z",
            "TxId":
"2da02a53aa1032602df6c68c5628a4ab8b22ba107c0201520ce495948901aa98",
            "Value": {
                "Amount": 5,
                "AssetType": "otransaction",
                "FromAccountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
                "Timestamp": "2022-12-08T10:45:56Z",
                "ToAccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
                "TokenId": "tokenOne",
                "TransactionId":
"otransaction~2da02a53aa1032602df6c68c5628a4ab8b22ba107c0201520ce495948901aa98
```


Copyright © 2025, Oracle and/or its affiliates.

```
~9c3ce5f21abd98ca018c196086d73a812f2f49dba323f1de4f6867eecfeec8ff",
                "TransactionType": "TRANSFER",
                "TriggeredByUserAccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc"
            }
        }
    ],
    "transactionId":
"otransaction~2da02a53aa1032602df6c68c5628a4ab8b22ba107c0201520ce495948901aa98
~9c3ce5f21abd98ca018c196086d73a812f2f49dba323f1de4f6867eecfeec8ff"
}
```

`DeleteHistoricalTransactions`
This method deletes transactions before a specified time stamp from the state database. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) DeleteHistoricalTransactions(timestamp string)
(interface{}, error) {
            auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Transaction.DeleteHistoricalTrans
actions", "TOKEN")
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            return
t.Ctx.ERC1155Transaction.DeleteHistoricalTransactions(timestamp)
      }
```

Parameters:

- `timestamp: string` – All transactions before this time stamp will be deleted.

Return Value

```
{
    "Transactions": [

"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
"
    ],
    "msg": "Successfuly deleted transaction older than
date:2022-04-06T08:17:53Z"
}
```

Methods for Token Behavior Management - Mintable Behavior

`MintBatch`
This method creates (mints) multiple tokens in a batch operation. This method creates only fungible tokens or fractional non-fungible tokens.
For fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. You cannot mint more than the `max_mint_quantity` property of the token, if that property was specified when the token was created or updated.

Copyright © 2025, Oracle and/or its affiliates.

For non-fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. Additionally, the caller must also be the creator of the token. There is no upper limit to the quantity of fractional non-fungible tokens that can be minted.
You cannot use this method to mint a whole non-fungible token.

```
func (t *Controller) MintBatch(orgId string, userId string, tokenIds
[]string, quantity []float64) (interface{}, error) {
            accountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in generating the accountId.
Error: %s", err)
            }
            var tokens []interface{}
            for _, tokenId := range tokenIds {
                  tokenAssetValue, err := t.getTokenObject(tokenId)
                  if err != nil {
                        return nil, err
                  }
                  tokens = append(tokens, tokenAssetValue.Interface())
            }
            return t.Ctx.ERC1155Token.MintBatch(accountId, tokens, quantity)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenIds []string` – The list of token IDs to mint tokens for.
- `quantity []float64` – The list of quantities of tokens to mint, corresponding to the token ID array.

Returns:

- On success, a JSON object that includes details on the minted tokens.

Return Value

```
{
    "details": [
        {
            "msg": "Successfully minted 100 tokens of fractional tokenId:
plot55 to Org-Id: appdev, User-Id: idcqa"
        },
        {
            "msg": "Successfully minted 100 tokens of tokenId: 'loyalty' to
Token-Account-Id
'oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e'"
        }
    ],
    "msg": "Successfully minted batch of tokens for User-Account-Id
```


Copyright © 2025, Oracle and/or its affiliates.

```
'ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38'
(Org-Id: 'appdev', User-Id: 'idcqa')"
}
```

Methods for Token Behavior Management - Transferable Behavior

`BatchTransferFrom`
This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.
For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.
For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.
This method does not validate that the caller of the method is the specified sender. This method can be called by any user.

```
func (t *Controller) BatchTransferFrom(fromOrgId string, fromUserId string,
toOrgId string, toUserId string, tokenIds []string, quantity []float64)
(interface{}, error) {
            fromAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(fromOrgId, fromUserId,
constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            toAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(toOrgId, toUserId,
constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            var tokens []interface{}
            for _, tokenId := range tokenIds {
                  tokenAssetValue, err := t.getTokenObject(tokenId)
                  if err != nil {
                        return nil, err
                  }
                  tokens = append(tokens, tokenAssetValue.Interface())
            }
            return t.Ctx.ERC1155Token.BatchTransferFrom(fromAccountId,
toAccountId, tokens, quantity)
      }
```

Parameters:

- `fromOrgId string` – The membership service provider (MSP) ID of the sender and token owner in the current organization.
- `fromUserId string` – The user name or email ID of the sender and token owner.
- `toOrgId string` – The membership service provider (MSP) ID of the receiver in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `toUserId string` – The user name or email ID of the receiver.
- `tokenIds string[]` – A list of token IDs for the tokens to transfer.
- `quantity float64[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

- On success, a message with details for each token transfer.

Return Value

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`SafeBatchTransferFrom`
This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.
For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.
For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.
The caller of the method must be the specified sender. This method can be called by any user.

```
func (t *Controller) SafeBatchTransferFrom(fromOrgId string, fromUserId
string, toOrgId string, toUserId string, tokenIds []string, quantity
[]float64) (interface{}, error) {
            fromAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(fromOrgId, fromUserId,
constants.UserAccount)
            if err != nil {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            toAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(toOrgId, toUserId,
constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            var tokens []interface{}
            for _, tokenId := range tokenIds {
                  tokenAssetValue, err := t.getTokenObject(tokenId)
                  if err != nil {
                        return nil, err
                  }
                  tokens = append(tokens, tokenAssetValue.Interface())
            }
            return t.Ctx.ERC1155Token.SafeBatchTransferFrom(fromAccountId,
toAccountId, tokens, quantity)
      }
```

Parameters:

- `fromOrgId string` – The membership service provider (MSP) ID of the sender and token owner in the current organization.
- `fromUserId string` – The user name or email ID of the sender and token owner.
- `toOrgId string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId string` – The user name or email ID of the receiver.
- `tokenIds string[]` – A list of token IDs for the tokens to transfer.
- `quantity float64[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

- On success, a message with details for each token transfer.

Return Value

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
```


Copyright © 2025, Oracle and/or its affiliates.

```
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`BalanceOfBatch`

This method completes a batch operation that gets the balance of token accounts. The account details are specified in three separate lists of organization IDs, user IDs, and token IDs. This method can be called only by a `Token Admin` of the chaincode or by account owners. Account owners can see balance details only for accounts that they own.

```
func (t *Controller) BalanceOfBatch(orgIds []string, userIds []string,
tokenIds []string) (interface{}, error) {
            callerAccountCheck := false
            _, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.BalanceOfBatch", "TOKEN")
            if err != nil {
                  callerAccountCheck = true
            }
            accountIds, err :=
t.Ctx.ERC1155Account.GenerateAccountIds(orgIds, userIds, callerAccountCheck)
            if err != nil {
                  return nil, fmt.Errorf("error in BalanceOfBatch. Error:
%s", err)
            }
            var tokens []interface{}
            for _, tokenId := range tokenIds {
                  tokenAssetValue, err := t.getTokenObject(tokenId)
                  if err != nil {
                        tokens = append(tokens, map[string]interface{}
{"TokenId": tokenId})
                  } else {
                        tokens = append(tokens, tokenAssetValue.Interface())
                  }
            }
            return t.Ctx.ERC1155Account.BalanceOfBatch(accountIds, tokens)
      }
```

Parameters:

- `orgIds []string` – A list of the membership service provider (MSP) IDs in the current organization.
- `userIds []string` – A list of the user name or email IDs.
- `tokenIds []string` – A list of the token IDs.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:
In the following example, the token ID `FNFT` represents a fractional non-fungible token and the token ID `FT` represents a fungible token.

```
[
    {
        "OrgId": "appdev",
        "UserId": "idcqa",
        "UserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "TokenAccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "TokenId": "FNFT",
        "Balance": 100
    },
    {
        "OrgId": "appdev",
        "UserId": "idcqa",
        "UserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "TokenAccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
        "TokenId": "FT",
        "Balance": 50
    },
    {
        "OrgId": "appdev",
        "UserId": "user1_minter",
        "UserAccountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
        "TokenAccountId":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "TokenId": "FNFT",
        "Balance": 10
    }
]
```

`ExchangeToken`
This method exchanges tokens between specified accounts. This method only supports exchanging between an NFT and a fungible token or a fungible token and an NFT. This method can be called only by the account owner.

```
func (t *Controller) ExchangeToken(fromTokenId string, fromOrgId string,
fromUserId string, fromTokenQuantity float64, toTokenId string, toOrgId
string, toUserId string, toTokenQuantity float64) (interface{}, error) {
            fromUserAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(fromOrgId, fromUserId,
constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            auth, err :=
```


Copyright © 2025, Oracle and/or its affiliates.

```
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Account.ExchangeToken", "TOKEN",
map[string]string{"accountId": fromUserAccountId})
            if err != nil && !auth {
                  return nil, fmt.Errorf("error in authorizing the caller
%s", err.Error())
            }
            toUserAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(toOrgId, toUserId,
constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            return t.Ctx.ERC1155Token.ExchangeToken(fromTokenId,
fromUserAccountId, fromTokenQuantity, toTokenId, toUserAccountId,
toTokenQuantity)
      }
```

Parameters:

- `fromTokenId string` – The ID of the token that the sender owns.
- `fromOrgId string` – The membership service provider (MSP) ID of the sender in the current organization.
- `fromUserId string` – The user name or email ID of the sender.
- `fromTokenQuantity float64` – The quantity of tokens from the sender to exchange with the receiver.
- `toTokenId string` – The ID of the token that the receiver owns.
- `toOrgId string` – The membership service provider (MSP) ID of the receiver in the current organization.
- `toUserId string` – The user name or email ID of the receiver.
- `toTokenQuantity float64` – The quantity of tokens from the receiver to exchange with the sender.

Returns:

- On success, a message with token exchange details.

Return Value

```
{
    "msg": "Succesfully exchanged 10 tokens of type 'nonfungible' with
tokenId: [r1] from Account
'oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371'
(OrgId: appdev, UserId: idcqa) to 10 tokens of type 'fungible' with tokenId:
[loy1] from Account
'oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c'
(OrgId: 'appdev', UserId: 'user1_minter')"
}
```

Methods for Token Behavior Management - Burnable Behavior

Copyright © 2025, Oracle and/or its affiliates.

BurnBatch
This method deactivates, or burns, the specified fungible and non-fungible tokens. Any user can call this method.

```
func (t *Controller) BurnBatch(orgId string, userId string, tokenIds
[]string, quantity []float64) (interface{}, error) {
            accountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
            if err != nil {
                  return nil, fmt.Errorf("error in BatachTransferFrom. Error:
%s", err)
            }
            var tokens []interface{}
            for _, tokenId := range tokenIds {
                  tokenAssetValue, err := t.getTokenObject(tokenId)
                  if err != nil {
                        return nil, err
                  }
                  tokens = append(tokens, tokenAssetValue.Interface())
            }
            return t.Ctx.ERC1155Token.Burn(accountId, tokens, quantity)
      }
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID in the current organization.
- `userId string` – The user name or email ID.
- `tokenIds []string` – The list of the token IDs to burn
- `quantity []float64` – The list of quantities of tokens to burn, corresponding to the token ID array..

Returns:

- On success, a message with details about the burn operations.

Return Value

```
[
    {
        "msg": "Successfully burned NFT token: 'art' from Account-Id:
oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6
(Org-Id: appdev, User-Id: idcqa)"
    },
    {
        "msg": "Successfully burned 2 tokens of tokenId: FT from Account-ID
oaccount~9a940587fd322ccc8400233244cd3b13f3aa2a52e418e4c71fb819a2217bc49e
(Org-Id: AutoF1377358917, User-Id: idcqa)"
    },
    {
        "msg": "Successfully burned 2 token share of tokenId: FNFT from
Account-ID
oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a
(Org-Id: AutoF1377358917, User-Id: idcqa)"
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
]
```

BurnNFT

This method deactivates, or burns, the specified non-fungible token, and returns a token object and token history. Any user with the burner role can call this method.

```
func (t *Controller) BurnNFT(orgId string, userId string, tokenId string)
(interface{}, error) {
            tokenAsset, err := t.Ctx.ERC1155Token.Get(tokenId)
            if err != nil {
                  return nil, err
            }
            token := tokenAsset.(map[string]interface{})
            if token[constants.TokenType] != constants.NonFungible {
                  return nil, fmt.Errorf("only nonfungible tokens are
allowed")
            }
            if token[constants.IsBurned] == true {
                  return nil, fmt.Errorf("token with tokenId %s is already
burned", tokenId)
            }
            tokenQuantity := float64(1)
            tokenUnit := token[constants.TokenUnit]
            tokenHistory, err := t.Ctx.ERC1155Token.History(tokenId)
            if err != nil {
                  return nil, err
            }
            if tokenUnit == constants.Fractional {
                  owners, err := t.Ctx.ERC1155Token.OwnerOf(tokenId)
                  if err != nil {
                        return nil, err
                  }
                  ownersList := owners.([]map[string]interface{})
                  if len(ownersList) != 1 {
                        return nil, fmt.Errorf("NFT has multiple owners, to
completely burn this NFT it should have only one owner")
                  }
                  tokenQuantity = token[constants.Quantity].(float64)
            }
            token[constants.TokenId], err =
strconv.Atoi(token[constants.TokenId].(string))
            if err != nil {
                  return nil, fmt.Errorf("tokenId is expected to be integer
but found %s", tokenId)
            }
            tokenHistoryBytes, err := json.Marshal(tokenHistory)
            if err != nil {
                  return nil, err
            }
            var tokenHistoryAsRawJson json.RawMessage
            err = json.Unmarshal(tokenHistoryBytes, &tokenHistoryAsRawJson)
            if err != nil {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                return nil, err
            }
            token[constants.TokenHistory] = string(tokenHistoryAsRawJson)
            tokenIds := []string{tokenId}
            tokenQuantities := []float64{tokenQuantity}
            token[constants.IsBurned] = true
            _, err = t.BurnBatch(orgId, userId, tokenIds, tokenQuantities)
            if err != nil {
                return nil, err
            }
            return token, nil
      }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID in the current organization.
- `userId: string` – The user name or email ID.
- `tokenId: string` – The ID of the non-fungible token to burn

Returns:

- On success, a token object in JSON format that includes token history information.

Return Value


```
{
    "AssetType": "otoken",
    "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2023-08-22T13:14:46+05:30",
    "IsBurned": true,
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "On_sale_flag": false,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Price": 120,
    "Quantity": 1,
    "Roles": {
        "minter_role_name": "minter"
    },
    "TokenDesc": "",
    "TokenHistory":
"[{\"IsDelete\":\"false\",\"Timestamp\":\"2023-08-22T13:14:46+05:30\",\"TxId\"
:\"c0ea212f19197c5b86323bfca11c6ca545aa0af5d40cd04f9e955b5371fd40ae\",\"Value\
```


Copyright © 2025, Oracle and/or its affiliates.

```
":{\"AssetType\":\"otoken\",\"Behaviors\":
[\"indivisible\",\"singleton\",\"mintable\",\"transferable\",\"burnable\",\"ro
les\"],\"CreatedBy\":\"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd
1aae201d347ad1288d\",\"CreationDate\":\"2023-08-22T13:14:46+05:30\",\"IsBurned
\":false,\"Mintable\":
{\"Max_mint_quantity\":20000},\"On_sale_flag\":false,\"Owner\":\"oaccount~42e8
9f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d\",\"Price\":120,\
"Quantity\":1,\"Roles\":
{\"minter_role_name\":\"minter\"},\"TokenDesc\":\"\",\"TokenId\":\"1\",\"Token
Metadata\":
{\"Description\":\"\",\"Image\":\"\",\"Painter_name\":\"\",\"Painting_name\":\
"\"},\"TokenName\":\"artcollection\",\"TokenStandard\":\"erc1155+
\",\"TokenType\":\"nonfungible\",\"TokenUnit\":\"whole\",\"TokenUri\":\"exampl
e.com\"}}]",
    "TokenId": 1,
    "TokenMetadata": {
        "Description": "",
        "Image": "",
        "Painter_name": "",
        "Painting_name": ""
    },
    "TokenName": "artcollection",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "example.com"
}
```


SDK Methods

- Access Control Management
- Token Configuration Management
- Account Management
- Role Management
- Transaction History Management
- Token Behavior Management
  - Mintable Behavior
  - Transferable Behavior
  - Burnable Behavior

Methods for Access Control Management

`IsUserTokenAdmin`
This method returns the Boolean value `true` if the specified user is a `Token Admin`, and `false` otherwise. The method can be called only by a `Token Admin` of the token chaincode.

```
Ctx.ERC1155Auth.IsUserTokenAdmin(orgId string, userId string) (interface{},
error)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Return Value

```
{
  "result": true
}
```

`AddAdmin`
This method adds a user as a `Token Admin` of the token chaincode. The method can be called only by a `Token Admin` of the token chaincode.

```
Ctx.ERC1155Admin.AddAdmin(orgId string, userId string) (interface{}, error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user added as a `Token Admin` of the token chaincode.

Return Value

```
{
  "msg": "Successfully added Admin (orgId: appdev, userId: user1)"
}
```

`RemoveAdmin`
This method removes a user as a `Token Admin` of the token chaincode. The method can be called only by a `Token Admin` of the token chaincode. You cannot remove yourself as a `Token Admin`.

```
Ctx.ERC1155Admin.RemoveAdmin(orgId string, userId string) (interface{}, error)
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Returns:

- On success, a message that lists details for the user removed as a `Token Admin` of the token chaincode.

Copyright © 2025, Oracle and/or its affiliates.

Return Value

```
{
    "msg": "Successfully removed Admin (orgId appdev userId user1)"
}
```

GetAllAdminUsers
This method returns a list of all `Token Admin` users.

```
Ctx.ERC1155Admin.GetAllAdminUsers() (interface{}, error)
```

Parameters:

- none

Returns:

- On success, a list of all `Token Admin` users, identifed by organization ID and user ID.

Return Value

```
{
  "admins": [
    {
      "OrgId": "appdev",
      "UserId": "idcqa"
    },
    {
      "OrgId": "appdev",
      "UserId": "user1"
    }
  ]
}
```

CheckAuthorization
Use this method to add an access control check to an operation. This is an asynchronous function. Most automatically generated methods include access control. Certain token methods can be run only by the `ERC721Admin` or `Account Owner` of the token or by a `MultipleAccountOwner` for users with multiple accounts. The `CheckAuthorization` method is part of the `erc721Auth` class, which you access via the `Ctx` object. The access control mapping is described in the `oChainUtil.go` file, as shown in the following text. You can modify access control by editing the `oChainUtil.go` file.

```
  var t TokenAccess
      var r RoleAccess
      var a AccountAccess
      var as AccountStatusAccess
      var h HoldAccess
      var ad AdminAccess
      var trx TransactionAccess
      var tc TokenConversionAccess
      var auth AuthAccess
      var erc721ad ERC721AdminAccess
```


Copyright © 2025, Oracle and/or its affiliates.

```
        var erc721t ERC721TokenAccess
        var erc721r ERC721RoleAccess
        var erc721a ERC721AccountAccess
        var erc721as ERC721AccountStatusAccess
        var erc721trx ERC721TransactionAccess

        var erc1155ad ERC1155AdminAccess
        var erc1155t ERC1155TokenAccess
        var erc1155a ERC1155AccountAccess
        var erc1155as ERC1155AccountStatusAccess
        var erc1155trx ERC1155TransactionAccess
        var erc1155role ERC1155RoleAccess
        trx.DeleteHistoricalTransactions = []string{"Admin"}
        ad.AddAdmin = []string{"Admin"}
        ad.RemoveAdmin = []string{"Admin"}
        ad.GetAllAdmins = []string{"Admin", "OrgAdmin"}
        ad.AddOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
        ad.RemoveOrgAdmin = []string{"Admin", "OrgAdminOrgIdCheck"}
        ad.GetOrgAdmins = []string{"Admin", "OrgAdmin"}
        ad.IsTokenAdmin = []string{"Admin", "MultipleAccountOwner", "OrgAdmin"}
        t.Save = []string{"Admin"}
        t.GetAllTokens = []string{"Admin", "OrgAdmin"}
        t.Update = []string{"Admin"}
        t.GetTokenDecimals = []string{"Admin", "OrgAdmin"}
        t.GetTokensByName = []string{"Admin", "OrgAdmin"}
        t.AddRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
        t.RemoveRoleMember = []string{"Admin", "OrgAdminRoleCheck"}
        t.IsInRole = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
        t.GetTotalMintedTokens = []string{"Admin", "OrgAdmin"}
        t.GetNetTokens = []string{"Admin", "OrgAdmin"}
        t.Get = []string{"Admin", "OrgAdmin"}
        t.GetTokenHistory = []string{"Admin", "OrgAdmin"}
        a.CreateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        a.AssociateToken = []string{"Admin", "OrgAdminAccountIdCheck"}
        a.GetAllAccounts = []string{"Admin"}
        a.GetAllOrgAccounts = []string{"Admin", "OrgAdminOrgIdCheck"}
        a.GetAccount = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.History = []string{"Admin", "AccountOwner"}
        a.GetAccountTransactionHistory = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
        a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"OrgAdminAccountIdCheck", "AccountOwner"}
        a.GetSubTransactionsById = []string{"Admin", "TransactionInvoker"}
        a.GetSubTransactionsByIdWithFilters = []string{"Admin",
"TransactionInvoker"}
        a.GetAccountBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.GetAccountOnHoldBalance = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.GetOnHoldIds = []string{"Admin", "OrgAdminAccountIdCheck",
"AccountOwner"}
        a.GetAccountsByUser = []string{"Admin", "OrgAdminOrgIdCheck",
"MultipleAccountOwner"}
```


Copyright © 2025, Oracle and/or its affiliates.

```
        as.Get = []string{"Admin", "OrgAdminAccountIdCheck", "AccountOwner"}
        as.ActivateAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        as.SuspendAccount = []string{"Admin", "OrgAdminOrgIdCheck"}
        as.DeleteAccount = []string{"Admin", "OrgAdminOrgIdCheck"}

        r.GetAccountsByRole = []string{"Admin"}
        r.GetUsersByRole = []string{"Admin"}
        r.GetOrgAccountsByRole = []string{"Admin", "OrgAdminOrgIdCheck"}
        r.GetOrgUsersByRole = []string{"Admin", "OrgAdminOrgIdCheck"}
        tc.InitializeExchangePoolUser = []string{"Admin"}
        tc.AddConversionRate = []string{"Admin"}
        tc.UpdateConversionRate = []string{"Admin"}
        tc.GetConversionRate = []string{"Admin", "OrgAdmin", "AnyAccountOwner"}
        tc.GetConversionRateHistory = []string{"Admin", "OrgAdmin",
"AnyAccountOwner"}
        tc.TokenConversion = []string{"Admin", "AnyAccountOwner"}
        tc.GetExchangePoolUser = []string{"Admin"}

        erc721ad.AddAdmin = []string{"Admin"}
        erc721ad.GetAllAdmins = []string{"Admin"}
        erc721ad.IsTokenAdmin = []string{"Admin"}
        erc721ad.RemoveAdmin = []string{"Admin"}
        erc721trx.DeleteHistoricalTransactions = []string{"Admin"}
        erc721t.Save = []string{"Admin"}
        erc721t.GetAllTokens = []string{"Admin"}
        erc721t.Update = []string{"Admin"}
        erc721t.GetTokensByName = []string{"Admin"}
        erc721t.AddRoleMember = []string{"Admin"}
        erc721t.RemoveRoleMember = []string{"Admin"}
        erc721t.IsInRole = []string{"Admin", "AccountOwner"}
        erc721t.Get = []string{"Admin", "TokenOwner"}
        erc721t.GetAllTokensByUser = []string{"Admin", "AccountOwner"}
        erc721t.TotalSupply = []string{"Admin"}
        erc721t.TotalNetSupply = []string{"Admin"}
        erc721t.History = []string{"Admin"}
        erc721a.CreateAccount = []string{"Admin"}
        erc721a.CreateUserAccount = []string{"Admin"}
        erc721a.CreateTokenAccount = []string{"Admin"}
        erc721a.AssociateFungibleTokenToAccount = []string{"Admin",
"AccountOwner"}
        erc721a.GetAllAccounts = []string{"Admin"}
        erc721a.History = []string{"Admin", "AccountOwner"}
        erc721a.GetAccountTransactionHistory = []string{"Admin", "AccountOwner"}
        erc721a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"AccountOwner"}
        erc721a.GetAccountByUser = []string{"Admin", "MultipleAccountOwner"}
        erc721a.BalanceOf = []string{"Admin", "MultipleAccountOwner"}

        erc721as.Get = []string{"Admin", "AccountOwner"}
        erc721as.ActivateAccount = []string{"Admin"}
        erc721as.SuspendAccount = []string{"Admin"}
        erc721as.DeleteAccount = []string{"Admin"}
```


Copyright © 2025, Oracle and/or its affiliates.

```
        erc721r.GetAccountsByRole = []string{"Admin"}
        erc721r.GetUsersByRole = []string{"Admin"}

        erc1155ad.AddAdmin = []string{"Admin"}
        erc1155ad.GetAllAdmins = []string{"Admin"}
        erc1155ad.IsUserTokenAdmin = []string{"Admin"}
        erc1155ad.RemoveAdmin = []string{"Admin"}
        erc1155t.AddRoleMember = []string{"Admin"}
        erc1155t.IsInRole = []string{"Admin"}
        erc1155t.GetAllTokens = []string{"Admin"}
        erc1155t.GetAllTokensByUser = []string{"Admin", "AccountOwner"}
        erc1155t.Get = []string{"Admin", "TokenOwner"}
        erc1155t.RemoveRoleMember = []string{"Admin"}
        erc1155t.TotalNetSupply = []string{"Admin"}
        erc1155t.TotalSupply = []string{"Admin"}
        erc1155t.GetTokenDecimal = []string{"Admin"}
        erc1155t.GetTokensByName = []string{"Admin"}
        erc1155t.GetTotalMintedTokens = []string{"Admin"}
        erc1155t.GetNetTokens = []string{"Admin"}
        erc1155t.Save = []string{"Admin"}
        erc1155t.Update = []string{"Admin"}

        erc1155trx.DeleteHistoricalTransactions = []string{"Admin"}

        erc1155role.GetAccountsByRole = []string{"Admin"}
        erc1155role.GetUsersByRole = []string{"Admin"}
        erc1155a.AssociateFungibleTokenToAccount = []string{"Admin",
"AccountOwner"}
        erc1155a.BalanceOfBatch = []string{"Admin"}
        erc1155a.CreateAccount = []string{"Admin"}
        erc1155a.CreateTokenAccount = []string{"Admin"}
        erc1155a.CreateUserAccount = []string{"Admin"}
        erc1155a.GetAccountTransactionHistory = []string{"Admin",
"AccountOwner"}
        erc1155a.GetAccountTransactionHistoryWithFilters = []string{"Admin",
"AccountOwner"}
        erc1155a.GetAccountsByUser = []string{"Admin", "AccountOwner"}
        erc1155a.GetAccount = []string{"Admin", "AccountOwner"}
        erc1155a.History = []string{"Admin", "AccountOwner"}
        erc1155a.GetAllAccounts = []string{"Admin"}
        erc1155a.ExchangeToken = []string{"AccountOwner"}
        erc1155a.GetAccountDetailsByUser = []string{"Admin", "AccountOwner"}

        erc1155as.Get = []string{"Admin", "AccountOwner"}
        erc1155as.ActivateAccount = []string{"Admin"}
        erc1155as.SuspendAccount = []string{"Admin"}
        erc1155as.DeleteAccount = []string{"Admin"}
        var accessMap TokenAccessControl
        accessMap.Token = t
        accessMap.Account = a
        accessMap.AccountStatus = as
        accessMap.Hold = h
        accessMap.Role = r
        accessMap.Admin = ad
```


Copyright © 2025, Oracle and/or its affiliates.

```
accessMap.Auth = auth
accessMap.TokenConversion = tc
accessMap.ERC721ADMIN = erc721ad
accessMap.ERC721TOKEN = erc721t
accessMap.ERC721ACCOUNT = erc721a
accessMap.ERC721AccountStatus = erc721as
accessMap.ERC721ROLE = erc721r
accessMap.ERC721TRANSACTION = erc721trx
accessMap.ERC1155Account = erc1155a
accessMap.ERC1155AccountStatus = erc1155as
accessMap.ERC1155Admin = erc1155ad
accessMap.ERC1155Token = erc1155t
accessMap.ERC1155Transaction = erc1155trx
accessMap.ERC1155Role = erc1155role
```

```
Ctx.ERC1155Auth.CheckAuthorization(funcName string, args []string) (bool,
error)
```

Parameters:

- `funcName string` – The map value between the receivers and methods as described in the `oChainUtil.go` file.
- `args` – A variable argument where `args[0]` takes the constant `'TOKEN'` and `args[1]` takes the `accountId` parameter to add an access control check for an `AccountOwner`. To add an access control check for a `MultipleAccountOwner`, `args[1]` takes the `orgId` parameter and `args[2]` takes the `userId` parameter. To add an access control check for a `TokenOwner`, `args[1]` takes the token ID parameter.

Returns:

- A `bool` response of error as required.

Methods for Token Configuration Management

Save

This method creates tokens. Every token that is defined has its own create method. For non-fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method to create an NFT. If not, any user can use this method to create (mint) NFTs. The user who calls this method becomes the owner of the NFT (whole or fractional).

```
Ctx.ERC1155Token.Save(token interface{}, quantity ...float64) (interface{},
error)
```

Parameters:

- `tokenAsset: interface{}` – The token asset. The properties of the asset are defined in the model file.
- `quantity: number` – For non-fungible tokens only, the number of tokens to mint. The only supported value for this parameter is `1`.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, the token asset in JSON format, which includes the following information.
- `Behavior` or `Behaviors` – A list of token behaviors. This property cannot be edited.
- `CreatedBy` – The account ID of the caller, who is the user minting the token. This property cannot be edited.
- `CreationDate` – The time stamp of the minting transaction. This property cannot be edited.
- `IsBurned` – This property indicates whether the token is burned. This property cannot be edited.
- `Mintable` – The properties related to minting. The `max_mint_quantity` value defines the maximum number of tokens that can be created for the token class.
- `Owner` – The account ID of the current owner, who is the caller of the method.
- `Symbol` – The symbol of the token. This property cannot be edited.
- `TokenDesc` – The description of the token.
- `TokenMetadata` – JSON information that describes the token.
- `TokenName` – The name of the token. This property cannot be edited.
- `TokenStandard` – The standard of the token. This property cannot be edited.
- `TokenType` – The type of the token (fungible or non-fungible). This property cannot be edited.
- `TokenUnit` – The unit of the token (whole or fractional). This property cannot be edited.
- `TokenUri` – The URI of the token.
- `Quantity` – The quantity of the token.

Return Value Example (Whole NFT):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2022-12-29T09:57:03+05:30",
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 500
    },
    "OnSaleFlag": false,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Price": 100,
    "Quantity": 1,
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "token description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "artcollection",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
}
```


Return Value Example (Fungible Token):


```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "Currency_name": "Dollar",
    "Divisible": {
        "Decimal": 2
    },
    "Mintable": {
        "Max_mint_quantity": 10000
    },
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "Loyalty",
    "TokenName": "loyalty",
    "TokenStandard": "erc1155+",
    "TokenType": "fungible",
    "TokenUnit": "fractional",
    "Token_to_currency_ratio": 0
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fractional NFT):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2023-06-14T09:53:53+05:30",
    "Divisible": {
        "Decimal": 2
    },
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 20000
    },
    "On_sale_flag": false,
    "Price": 1000,
    "Quantity": 100,
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "realEstate",
    "TokenMetadata": {
        "Description": "Painting Description",
        "Image": "",
        "Painter_name": "",
        "Painting_name": "Paint"
    },
    "TokenName": "realestate",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "fractional",
    "TokenUri": "www.realestate.example.com"
}
```

`Update`
This method updates tokens. You cannot update token metadata or the token URI of non-fungible tokens.

```
Ctx.ERC1155Token.Update(tokenAsset interface{}) (interface{}, error)
```

Parameters:

- `tokenAsset: interface{}` – The token asset. The properties of the asset are defined in the model file.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, the updated token asset in JSON format.

Return Value Example (Whole NFT):


```
{
    "AssetType": "otoken",
    "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "CreationDate": "2022-12-29T09:57:03+05:30",
    "IsBurned": false,
    "Mintable": {
        "Max_mint_quantity": 500
    },
    "OnSaleFlag": false,
    "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "Price": 100,
    "Quantity": 1,
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "token description",
    "TokenId": "monalisa",
    "TokenMetadata": {
        "Description": "Mona Lisa Painting",
        "Image": "monalisa.jpeg",
        "PainterName": "Leonardo_da_Vinci",
        "PaintingName": "Mona_Lisa"
    },
    "TokenName": "artcollection",
    "TokenStandard": "erc1155+",
    "TokenType": "nonfungible",
    "TokenUnit": "whole",
    "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
}
```


History (Token)
This method returns the history for a specified token ID.

```
Ctx.ERC1155Token.History(tokenId string) (interface{}, error)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `tokenId string` – The ID of the token.

Returns:

- On success, a JSON array that contains the token history.

Return Value Example (Fungible Token):


```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-08T09:54:11Z",
    "TxId":
"823sa7c7a00941c62285c86f922bc4d3f5326a20f4bf2f82daa5bc661e4682e8",
    "Value": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "Rupees",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "Updated Token Description",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-08T09:54:11Z",
    "TxId":
"711bb7c7a00941c62285c86f922bc3b3f5326a20f4bf2f82daa5bc661e4682e8",
    "Value": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "roles"
      ],
      "Currency_name": "Dollar",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  }
]
```

Return Value Example (Fractional NFT):

```
[
    {
        "Timestamp": "2023-06-20T01:06:33Z",
        "TrxId":
"16e53db4f8107f9634b7c3a0a2a81a00f69b634f2a52902b809e544d07f272b1",
        "Value": {
            "AssetType": "otoken",
            "Behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:02:27Z",
            "Divisible": {
                "Decimal": 2
            },
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "On_sale_flag": true,
            "Owners": [
                {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "AccountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                    "TokenShare": 10
                },
                {
                    "AccountId":
"oaccount~3cddfdaa855900579d963aa6f755a4aed1f3a474a2462c1b45bd7f36df673224",
                    "TokenShare": 10
                }
            ],
            "Price": 2000,
            "Quantity": 20,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "FNFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "realestate",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "fractional",
            "TokenUri": "www.FNFT.example.com"
        }
    },
    {
        "Timestamp": "2023-06-20T01:02:27Z",
        "TrxId":
"cec80910d087682554048f911d2cf98b66382bbcf1615483fa1c96c7ea08077c",
        "Value": {
            "AssetType": "otoken",
            "Behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:02:27Z",
            "Divisible": {
                "Decimal": 2
            },
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "On_sale_flag": true,
            "Owners": [
                {
                    "AccountId":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                    "TokenShare": 20
                }
            ],
            "Price": 2000,
            "Quantity": 20,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "FNFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "realestate",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "fractional",
            "TokenUri": "www.FNFT.example.com"
        }
    }
]
```

Return Value Example (Whole NFT):

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-20T01:15:56Z",
        "TxId":
"89a3df3ebbe6dca2bcfbd51fc7dca9aab818a2af746b79a92dc8155b729ab22d",
        "Value": {
            "AssetType": "otoken",
            "Behaviors": [
                "indivisible",
                "singleton",
                "mintable",
                "transferable",
                "roles"
            ],
            "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:15:56Z",
            "IsBurned": false,
            "Mintable": {
```


Copyright © 2025, Oracle and/or its affiliates.

```
                    "Max_mint_quantity": 20000
                },
                "On_sale_flag": true,
                "Owner":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                "Price": 2000,
                "Quantity": 1,
                "Roles": {
                    "burner_role_name": "burner",
                    "minter_role_name": "minter"
                },
                "TokenDesc": "Updated Token Description",
                "TokenId": "NFT",
                "TokenMetadata": {
                    "Description": "",
                    "Image": "",
                    "Painter_name": "",
                    "Painting_name": ""
                },
                "TokenName": "artcollection",
                "TokenStandard": "erc1155+",
                "TokenType": "nonfungible",
                "TokenUnit": "whole",
                "TokenUri": "www.NFT.example.com"
            }
        },
        {
            "IsDelete": "false",
            "Timestamp": "2023-06-20T01:15:56Z",
            "TxId":
"90d6af3ebbe6dca2bcfbd51fc7dca9aab818a2af746b79a92dc8155b729ab22d",
            "Value": {
                "AssetType": "otoken",
                "Behaviors": [
                    "indivisible",
                    "singleton",
                    "mintable",
                    "transferable",
                    "roles"
                ],
                "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                "CreationDate": "2023-06-20T01:15:56Z",
                "IsBurned": false,
                "Mintable": {
                    "Max_mint_quantity": 20000
                },
                "On_sale_flag": true,
                "Owner":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
                "Price": 2000,
                "Quantity": 1,
                "Roles": {
                    "burner_role_name": "burner",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "NFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "artcollection",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "whole",
            "TokenUri": "www.NFT.example.com"
        }
    }
]
```

GetAllTokens

This method returns all of the token assets that are saved in the state database. This method can be called only by a `Token Admin` of the chaincode. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Token.GetAllTokens()() (interface{}, error)
```

Parameters:

- none

Return Value

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenTwo",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "art",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
      ],
      "BurnedBy":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "BurnedDate": "2022-12-08T10:49:37Z",
      "CreatedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "CreationDate": "2022-12-08T10:45:10Z",
      "IsBurned": true,
      "Mintable": {
        "Max_mint_quantity": 20000
      },
      "OnSaleFlag": false,
      "Owner": "",
      "Price": 0,
      "Roles": {
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "art",
      "TokenMetadata": {
        "Description": "",
        "Image": "",
        "PainterName": "",
        "PaintingName": ""
      },
      "TokenName": "artcollection",
      "TokenStandard": "erc1155+",
      "TokenType": "nonfungible",
      "TokenUnit": "whole",
      "TokenUri": "art.example.com",
      "TransferredBy":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
      "TransferredDate": "2022-12-08T10:47:04Z"
    }
  },
  {
    "key": "FNFT",
    "valueJson": {
        "AssetType": "otoken",
        "Behaviors": [
            "divisible",
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "CreatedBy":
"oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
        "CreationDate": "2023-06-20T01:02:27Z",
        "Divisible": {
            "Decimal": 2
        },
        "IsBurned": false,
        "Mintable": {
            "Max_mint_quantity": 20000
        },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "On_sale_flag": true,
            "Price": 2000,
            "Quantity": 20,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "FNFT",
            "TokenMetadata": {
                "Description": "",
                "Image": "",
                "Painter_name": "",
                "Painting_name": ""
            },
            "TokenName": "realestate",
            "TokenStandard": "erc1155+",
            "TokenType": "nonfungible",
            "TokenUnit": "fractional",
            "TokenUri": "www.FNFT.example.com"
        }
      },
      {
        "key": "FNFT",
        "valueJson": {
            "AssetType": "otoken",
            "Behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "CreatedBy":
    "oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a",
            "CreationDate": "2023-06-20T01:02:27Z",
            "Divisible": {
                "Decimal": 2
            },
            "IsBurned": false,
            "Mintable": {
                "Max_mint_quantity": 20000
            },
            "On_sale_flag": true,
            "Price": 2000,
            "Quantity": 20,
            "Roles": {
                "burner_role_name": "burner",
                "minter_role_name": "minter"
            },
            "TokenDesc": "",
            "TokenId": "FNFT",
            "TokenMetadata": {
                "Description": "",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "Image": "",
            "Painter_name": "",
            "Painting_name": ""
        },
        "TokenName": "realestate",
        "TokenStandard": "erc1155+",
        "TokenType": "nonfungible",
        "TokenUnit": "fractional",
        "TokenUri": "www.FNFT.example.com"
    }
  }
]
```

Get (Token)
This method returns a token object if the token is present in the state database. This method can be called only by a `Token Admin` of the chaincode or the token owner. For fractional NFTs, the list of owners is also returned.

```
Ctx.ERC1155Token.Get(id string, result ...interface{}) (interface{}, error)
```

Parameters:

- `id string` – The ID of the token to get.

Return Value Example (Whole NFT):

```
{
  "AssetType": "otoken",
  "Behaviors": [
    "indivisible",
    "singleton",
    "mintable",
    "transferable",
    "burnable",
    "roles"
  ],
  "CreatedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "CreationDate": "2022-12-08T10:55:29Z",
  "IsBurned": false,
  "Mintable": {
    "Max_mint_quantity": 20000
  },
  "OnSaleFlag": false,
  "Owner":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "Price": 0,
  "Quantity": 1,
  "Roles": {
    "minter_role_name": "minter"
  },
  "TokenDesc": "",
  "TokenId": "nftToken",
```


Copyright © 2025, Oracle and/or its affiliates.

```
  "TokenMetadata": {
    "Description": "",
    "Image": "",
    "PainterName": "",
    "PaintingName": ""
  },
  "TokenName": "artcollection",
  "TokenStandard": "erc1155+",
  "TokenType": "nonfungible",
  "TokenUnit": "whole",
  "TokenUri": "example.com"
}
```

Return Value Example (Fungible Token):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "Currency_name": "Dollar",
    "Divisible": {
        "Decimal": 2
    },
    "Mintable": {
        "Max_mint_quantity": 10000
    },
    "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
    },
    "TokenDesc": "Token Description",
    "TokenId": "Loyalty",
    "TokenName": "loyalty",
    "TokenStandard": "erc1155+",
    "TokenType": "fungible",
    "TokenUnit": "fractional",
    "Token_to_currency_ratio": 0
}
```

Return Value Example (Fractional NFT):

```
{
    "AssetType": "otoken",
    "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "roles"
        ],
        "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "CreationDate": "2023-06-14T09:53:53+05:30",
        "Divisible": {
            "Decimal": 2
        },
        "IsBurned": false,
        "Mintable": {
            "Max_mint_quantity": 20000
        },
        "On_sale_flag": false,
        "Owners": [
            {
                "AccountId":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
                "TokenShare": 100
            }
        ],
        "Price": 1000,
        "Quantity": 100,
        "Roles": {
            "burner_role_name": "burner",
            "minter_role_name": "minter"
        },
        "TokenDesc": "Token Description",
        "TokenId": "realEstate",
        "TokenMetadata": {
            "Description": "Painting Description",
            "Image": "",
            "Painter_name": "",
            "Painting_name": "Paint"
        },
        "TokenName": "realestate",
        "TokenStandard": "erc1155+",
        "TokenType": "nonfungible",
        "TokenUnit": "fractional",
        "TokenUri": "www.realestate.example.com"
}
```

`GetAllTokensByUser`

This method returns all of the token assets that are owned by a specified user. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Token.GetAllTokensByUser(accountId string) (interface{}, error)
```

Parameters:

- `accountId string` – The account ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

Return Value


```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "nftToken",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
      ],
      "CreatedBy":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "CreationDate": "2022-12-08T10:55:29Z",
      "IsBurned": false,
      "Mintable": {
        "Max_mint_quantity": 20000
      },
      "OnSaleFlag": false,
      "Owner":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
      "Price": 0,
      "Quantity": 1,
      "Roles": {
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "nftToken",
      "TokenMetadata": {
        "Description": "",
        "Image": "",
        "PainterName": "",
        "PaintingName": ""
      },
      "TokenName": "artcollection",
      "TokenStandard": "erc1155+",
      "TokenType": "nonfungible",
      "TokenUnit": "whole",
      "TokenUri": "example.com"
    }
  }
]
```

`OwnerOf`
This method returns the account ID, organization ID, and user ID of the owner of the specified token ID.

```
Ctx.ERC1155Token.OwnerOf(tokenId string) (interface{}, error)
```

Parameters:

- `tokenId string` – The ID of the token.

Return Value

```
{
  "AccountId":
"oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6",
  "OrgId": "appdev",
  "UserId": "idcqa"
}
```

`TokenURI`
This method returns the URI of a specified token. Anyone can call this method.

```
Ctx.ERC1155Token.TokenURI(tokenId string) (interface{}, error)
```

Parameters:

- `tokenId string` – The ID of the token.

Copyright © 2025, Oracle and/or its affiliates.

Return Value

```
{
    "TokenUri": "example.com"
}
```

Name
This method returns the name of the token class. Anyone can call this method.

```
Ctx.ERC1155Token.Name(tokenId string) (interface{}, error)
```

Parameters:

- `tokenId string` – The ID of the token.

Return Value

```
{"TokenName": "artcollection"}
```

TotalSupply
This method returns the total number of minted tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.TotalSupply(tokenAsset interface{}) (map[string]interface{},
error)
```

Parameters:

- `tokenAsset interface{}` – The token asset.

Return Value

```
{"TotalSupply": 100}
```

TotalNetSupply
This method returns the total number of minted tokens minus the number of burned tokens. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.TotalNetSupply(token interface{}) (interface{}, error)
```

Parameters:

- `token interface{}` – The token asset.

Return Value

```
{"TotalNetSupply": 100}
```


Copyright © 2025, Oracle and/or its affiliates.

`GetTokensByName`
This method returns all of the token assets for a specified token name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Token.GetTokensByName(tokenName string) (interface{}, error)
```

Parameters:

- `tokenName: string` – The name of the token.

Return Value

```
[
  {
    "key": "tokenOne",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenOne",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  },
  {
    "key": "tokenTwo",
    "valueJson": {
      "AssetType": "otoken",
      "Behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "roles"
      ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "Currency_name": "",
      "Divisible": {
        "Decimal": 2
      },
      "Mintable": {
        "Max_mint_quantity": 1000
      },
      "Roles": {
        "burner_role_name": "burner",
        "minter_role_name": "minter"
      },
      "TokenDesc": "",
      "TokenId": "tokenTwo",
      "TokenName": "moneytok",
      "TokenStandard": "erc1155+",
      "TokenType": "fungible",
      "TokenUnit": "fractional",
      "Token_to_currency_ratio": 0
    }
  }
]
```

`GetDecimals`
This method returns the number of decimal places for a specified token. If the divisible behavior is not specified for the token, then the default value of zero decimal places is returned.

```
Ctx.ERC1155Token.GetDecimals(tokenId string) (int, error)
```

Parameters:

- `tokenId: string` – The ID of the token.

Return Value

```
2
```

Methods for Account Management

`CreateAccount`
This method creates an account for a specified user and associated token accounts for fungible or non-fungible tokens. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a user has. Users must have accounts in the network to complete token-related operations. This method can be called only by a `Token Admin` of the chaincode.
A user account has a unique ID, which is formed by an SHA-256 hash of the `orgId` parameter and the `userId` parameter.
A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).
A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter,

Copyright © 2025, Oracle and/or its affiliates.

and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this single non-fungible token account.

```
Ctx.ERC1155Account.CreateAccount(orgId string, userId string, ftAccount bool,
nftAccount bool) (ERC1155UserAccount, error)
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `ftAccount bool` – If true, a fungible token account is created and associated with the user account.
- `nftAccount bool` – If true, a non-fungible token account is created and associated with the user account.

Returns:

- On success, a JSON object of the account that was created.

Return Value

```
{
  "AssetType": "ouaccount",
  "AccountId":
"ouaccount~cf20877546f52687f387e7c91d88b9722c97e1a456cc0484f40c747f7804feae",
  "UserId": "user1",
  "OrgId": "appdev",
  "TotalAccounts": 2,
  "TotalFtAccounts": 1,
  "AssociatedFtAccounts": [
    {
      "AccountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
      "TokenId": ""
    }
  ],
  "AssociatedNftAccount":
"oaccount~73c3e835dac6d0a56ca9d8def08269f83cefd59b9d297fe2cdc5a9083828fa58"
}
```

`CreateUserAccount`
This method creates an account for a specified user. An account must be created for any user who will have tokens at any point. The user account tracks the NFT account and the fungible token accounts that a user has. Users must have accounts in the network to complete token-related operations.
An account ID is an SHA-256 hash of the `orgId` parameter and the `userId` parameter. This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.ERC1155Account.CreateUserAccount(orgId string, userId string)
(interface{}, error)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- On success, a JSON object of the user account that was created.

Return Value

```
{
  "AssetType": "ouaccount",
  "AccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
  "UserId": "idcqa",
  "OrgId": "appdev",
  "TotalAccounts": 0,
  "TotalFtAccounts": 0,
  "AssociatedFtAccounts": [],
  "AssociatedNftAccount": ""
}
```

`CreateTokenAccount`

This method creates a fungible or non-fungible token account to associate with a user account.

A user can have multiple fungible token accounts with unique account IDs. Fungible token account IDs are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, the constant string `ft` separated by the tilde symbol (~), and a counter number that signifies the index of the fungible account that is being created separated by the tilde symbol (~).

A user can have only one non-fungible token account. Non-fungible token account IDs are unique and are formed by an SHA-256 hash of the `orgId` parameter, the `userId` parameter, and the constant string `nft` separated by the tilde symbol (~). All non-fungible tokens that a user owns, whether whole or fractional, are linked to this single non-fungible token account.

This method can be called only by a `Token Admin` of the chaincode.

```
Ctx.ERC1155Account.CreateTokenAccount(orgId string, userId string, tokenType
string) (ERC1155UserAccount, error)
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.
- `tokenType erc1155Token.TokenType` – The type of token account to create. The only supported token types are `nonfungible` and `fungible`.

Returns:

- On success, a JSON object of the token account that was created.

Copyright © 2025, Oracle and/or its affiliates.

Return Value

```
{
    "AssetType": "ouaccount",
    "AccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
    "UserId": "idcqa",
    "OrgId": "appdev",
    "TotalAccounts": 1,
    "TotalFtAccounts": 1,
    "AssociatedFtAccounts": [
        {
            "AccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "TokenId": ""
        }
    ],
    "AssociatedNftAccount": ""
}
```

`AssociateTokenToToken`
This method associates a user's fungible token account to a particular fungible token.

```
Ctx.ERC1155Account.AssociateTokenToToken(accountId string, tokenId string)
(interface{}, error)
```

Parameters:

- `accountId string` – The user account ID.
- `tokenId string` – The ID of the token.

Returns:

- On success, a JSON object of the user account, which shows that the fungible token was associated to the token account. For example, in the following example, the first object in the `AssociatedFtAccounts` array shows that the fungible token account ID and the token ID are associated.

Return Value

```
{
    "AssetType": "ouaccount",
    "AccountId":
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc",
    "UserId": "idcqa",
    "OrgId": "appdev",
    "TotalAccounts": 1,
    "TotalFtAccounts": 1,
    "AssociatedFtAccounts": [
        {
            "AccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "TokenId": "tokenOne"
```


Copyright © 2025, Oracle and/or its affiliates.

```
        }
    ],
    "AssociatedNftAccount": ""
}
```

GetAccountHistory
This method returns history for a specified token account.

```
Ctx.ERC1155Account.GetAccountHistory(accountId string)
```

Parameters:

- `accountId string` – The token account ID.

Returns:

- On success, an array of JSON objects that describes the account history.

Return Value

```
[
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-06T11:03:48Z",
        "TxId":
"c5180f3be3d9130f25a4b4e866f38a4283117dcbfbffb4f55e2c5b03dba0dd29",
        "Value": {
            "AccountCategory": "",
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "AssetType": "oaccount",
            "Balance": 100,
            "BapAccountVersion": 1
            "OrgId": "appdev",
            "TokenId": "loy1",
            "TokenName": "loyalty",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-06T11:02:39Z",
        "TxId":
"6f81b0c94b451d375a3892446aefbdf78d9fd1ac43699daa89f0dff10db5fd22",
        "Value": {
            "AccountCategory": "",
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "AssetType": "oaccount",
            "Balance": 0,
            "BapAccountVersion": 0
            "OrgId": "appdev",
            "TokenId": "loy1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "TokenName": "loyalty",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    },
    {
        "IsDelete": "false",
        "Timestamp": "2023-06-05T16:28:46Z",
        "TxId":
"8185af648546e909488e72149be497b210f74f95ada252c42da9c35cb9d98691",
        "Value": {
            "AccountCategory": "",
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "AssetType": "oaccount",
            "Balance": 0,
            "BapAccountVersion": 0
            "OrgId": "appdev",
            "TokenId": "",
            "TokenName": "",
            "TokenType": "fungible",
            "UserId": "idcqa"
        }
    }
]
```

`GetAccountWithStatus`
This method returns token account details for a specified user, including account status. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC1155Account.GetAccountWithStatus(userAccountId string,
tokenId ...string) (interface{}, error)
```

Parameters:

- `userAccountId string` – The account ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON object that includes token account details, including account status.

Return Value Example (Non-Fungible Token Account):

```
{
    "AccountId":
"oaccount~cc301bee057f14236a97d434909ec1084970921b008f6baab09c2a0f5f419a9a",
    "AssetType": "oaccount",
    "BapAccountVersion": 1,
    "NoOfNfts": 1,
    "OrgId": "appdev",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "Status": "active",
    "TokenType": "nonfungible",
    "UserId": "idcqa"
}
```

Return Value Example (Fungible Token Account):

```
{
    "AccountCategory": "",
    "AccountId":
"oaccount~2de8db6b91964f8c9009136831126d3cfa94e1d00c4285c1ea3e6d1f36479ed4",
    "AssetType": "oaccount",
    "Balance": 0,
    "BapAccountVersion": 0,
    "OrgId": "appdev",
    "Status": "active",
    "TokenId": "t1",
    "TokenName": "loyalty",
    "TokenType": "fungible",
    "UserId": "idcqa"
}
```

`GetAccount`
This method returns token account details for a specified user. This method can be called only by a `Token Admin` of the chaincode or the `Account Owner` of the account.

```
Ctx.ERC1155Account.Get(accountId string, tokenId ...string) (interface{},
error)
```

Parameters:

- `userAccountId string` – The account ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON object that includes token account details.

Return Value Example (Non-Fungible Token Account):

```
{
    "AccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
    "BapAccountVersion": 0,
    "AssetType": "oaccount",
    "NoOfNfts": 4,
    "OrgId": "appdev",
    "TokenType": "nonfungible",
    "UserId": "idcqa"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Return Value Example (Fungible Token Account):

```
{
    "AssetType": "oaccount",
    "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
    "UserId": "idcqa",
    "OrgId": "appdev",
    "BapAccountVersion": 0,
    "TokenType": "fungible",
    "TokenId": "loy1",
    "TokenName": "loyalty",
    "Balance": 90,
    "AccountCategory": ""
}
```

`GetAllAccounts`
This method returns details of all user accounts.

```
Ctx.ERC1155Account.GetAllAccounts() (interface{}, error)
```

Parameters:

- none

Return Value

```
[
    {
        "AssetType": "ouaccount",
        "AccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "UserId": "idcqa",
        "OrgId": "appdev",
        "TotalAccounts": 3,
        "TotalFtAccounts": 2,
        "AssociatedFtAccounts": [
            {
                "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
                "TokenId": "loy1"
            },
            {
                "AccountId":
"oaccount~58c5a6b09a41befca2a9ea2550439838c4dcf4d8a2a4f7c98e9319cf8479bfc4",
                "TokenId": ""
            }
        ],
        "AssociatedNftAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371"
    },
    {
        "AssetType": "ouaccount",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "AccountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
        "UserId": "user1_minter",
        "OrgId": "appdev",
        "TotalAccounts": 2,
        "TotalFtAccounts": 1,
        "AssociatedFtAccounts": [
            {
                "AccountId":
"oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c",
                "TokenId": "loy1"
            }
        ],
        "AssociatedNftAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446"
    },
]
```

`GetAccountDetailsByUser`
This method returns an account summary for a specified user and details of fungible and non-fungible tokens that are associated with the user.

```
Ctx.ERC1155Account.GetAccountDetailsByUser(orgId string, userId string)
(interface{}, error)
```

Parameters:

- `orgId string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId string` – The user name or email ID of the user.

Returns:

- On success, a JSON account object that includes and account summary for the specified user and details of fungible and non-fungible tokens that are associated with the user. For fractional non-fungible tokens, the `tokenShare` property in the `associatedNFTs` section shows the share that the user owns.

Return Value

```
{
    "AssociatedFTAccounts": [
        {
            "AccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
            "Balance": 90,
            "TokenId": "FT"
        },
    ],
    "AssociatedNFTAccount": {
        "AccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "AssociatedNFTs": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
            {
                "NFTTokenId": "FNFT",
                "TokenShare": 230
            },
            {
                "NFTTokenId": "NFT"
            },
            {
                "NFTTokenId": "NFT2"
            }
        ]
    },
    "UserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
}
```

`GetUserByAccountId`
This method returns the user details of a specified account ID.

```
Ctx.ERC1155Account.GetUserByAccountById(accountId string)
(map[string]interface{}, error)
```

Parameters:

- `accountId string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`orgId` and `userId`).

Return Value

```
{
    "OrgId": "appdev",
    "UserId": "idcqa"
}
```

Methods for Role Management

`AddRoleMember`
This method adds a role to a specified user and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.AddRoleMember(userRole string, userAccountId string, asset
interface{}) (interface{}, error)
```

Parameters:

- `userRole: string` – The name of the role to add to the specified user.
- `userAccountId: string` – The account ID of the user.
- `asset: interface{}` – The token asset.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message with account details.

Return Value

```
{
    "msg": "Successfully added role minter to
oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a
(orgId : appdev, userId : idcqa)"
}
```

`IsInRole`
This method returns a Boolean value to indicate if a user has a specified role. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.IsInRole(userRole string, userAccountId string, asset
interface{}) (bool, error)
```

Parameters:

- `userRole: string` – The name of the role to search for.
- `userAccountId: string` – The account ID of the user.
- `asset: interface{}` – The token asset.

Return Value

```
{
    "result": true
}
```

`RemoveRoleMember`
This method removes a role from a specified user and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Token.RemoveRoleMember(userRole string, userAccountId string,
asset interface{}) (interface{}, error)
```

Parameters:

- `userRole: string` – The name of the role to remove.
- `userAccountId: string` – The account ID of the user.
- `asset: interface{}` – The token asset.

Return Value

```
{
    "msg": "Successfully removed role 'minter' from Account Id:
oaccount~ec7e4de2f81e3ea071710e07b6ff7d9346e84ef665ca4650885dbe8c3e2bd4c0
(Org-Id: appdev, User-Id: idcqa)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAccountsByRole
This method returns a list of all account IDs for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Role.GetAccountsByRole(roleName string, token interface{})
(interface{}, error)
```

Parameters:

- `roleName: string` – The name of the role to search for.
- `token: interface{}` – The token asset.

Return Value

```
{
    "accounts": [

"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",

"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b"
    ]
}
```

GetUsersByRole
This method returns a list of all users for a specified role and token. Fungible tokens are specified by the token ID. Non-fungible tokens are specified by the token name.

```
Ctx.ERC1155Role.GetUsersByRole(roleName string, token interface{})
(interface{}, error)
```

Parameters:

- `roleName: string` – The name of the role to search for.
- `token: interface{}` – The token asset.

Return Value

```
{
    "Users": [
        {
            "AccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
            "OrgId": "appdev",
            "UserId": "idcqa"
        },
        {
            "AccountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
            "OrgId": "appdev",
            "UserId": "user1"
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
    ]
}
```

Methods for Transaction History Management

`GetAccountTransactionHistory`
This method returns account transaction history. This method can be called only by a `Token Admin` of the chaincode or by the account owner. For non-fungible tokens, this method can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.ERC1155Account.GetAccountTransactionHistory(tokenAccountId string)
(interface{}, error)
```

Parameters:

- `accountId string` – The user account ID.

Return Value

```
[
    {
        "Balance": 90,
        "Timestamp": "2023-06-06T11:11:09Z",
        "TokenId": "FNFT",
        "TransactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "TransactedAmount": 10,
        "TransactionId":
"otransaction~0f4d96fbf8fed88ea8a313342897772109l467c701848d595ebc3fffa88b3657
~7c88c736df38d5622512f1e8dcdd50710eb47c953f1ecb24ac44790a9e2f475b",
        "TransactionType": "DEBIT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    },
    {
        "Timestamp": "2023-06-06T11:11:09Z",
        "TokenId": "NFT",
        "TransactedAccount":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "TransactionId":
"otransaction~0f4d96fbf8fed88ea8a313342897772109l467c701848d595ebc3fffa88b3657
~178e3730bc5bee50d02f1464a4eebf733a051905f651e5789039adb4a3edc114",
        "TransactionType": "DEBIT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    },
    {
        "Timestamp": "2023-06-06T11:06:54Z",
        "TokenId": "NFT",
        "TransactedAccount":
"oaccount~e88276a3be547e31b567346bddde52d37734da4d5fae83ab2e5c98a10097371",
        "TransactionId":
"otransaction~6a13667ea3f6edc4c854e85b127526eccb58783f653c348b42a3869f0f29a4fb
```


Copyright © 2025, Oracle and/or its affiliates.

```
~a7cefb22ff39ee7e36967be71de27da6798548c872061a62dabc56d88d50b930",
        "TransactionType": "MINT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    },
    {
        "Balance": 100,
        "Timestamp": "2023-06-05T16:34:33Z",
        "TokenId": "FNFT",
        "TransactedAccount":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
        "TransactedAmount": 100,
        "TransactionId":
"otransaction~2bc15de1766d582d821bd8d61756bca02837dc683c0aa61f69657ccd1d95e335
~e4eb15d9354f694230df8835ade012100d82aa43672896a2c7125a86e3048f9f",
        "TransactionType": "MINT",
        "TriggeredByUserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38"
    }
]
```

GetTransactionById
This method returns the transaction details for a specified transaction ID.

```
Ctx.ERC1155Transaction.GetTransactionById(trxId string) (interface{}, error)
```

Parameters:

- `trxId string` – The ID of the transaction.

Return Value

```
{
    "history": [
        {
            "IsDelete": "false",
            "Timestamp": "2022-12-08T10:45:56Z",
            "TxId":
"2da02a53aa1032602df6c68c5628a4ab8b22ba107c0201520ce495948901aa98",
            "Value": {
                "Amount": 5,
                "AssetType": "otransaction",
                "FromAccountId":
"oaccount~60bb20c14a83f6e426e1437c479c5891e1c6477dfd7ad18b73acac5d80bc504b",
                "Timestamp": "2022-12-08T10:45:56Z",
                "ToAccountId":
"oaccount~1422a74d262a3a55a37cd9023ef8836f765d0be7b49d397696b9961d7434d22a",
                "TokenId": "tokenOne",
                "TransactionId":
"otransaction~2da02a53aa1032602df6c68c5628a4ab8b22ba107c0201520ce495948901aa98
~9c3ce5f21abd98ca018c196086d73a812f2f49dba323f1de4f6867eecfeec8ff",
                "TransactionType": "TRANSFER",
                "TriggeredByUserAccountId":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"ouaccount~24ffd4d32a028a85b4b960f5d55536c837b5429bc7f346150adfa904ec2937cc"
            }
        }
    ],
    "transactionId":
"otransaction~2da02a53aa1032602df6c68c5628a4ab8b22ba107c0201520ce495948901aa98
~9c3ce5f21abd98ca018c196086d73a812f2f49dba323f1de4f6867eecfeec8ff"
}
```

`DeleteHistoricalTransactions`
This method deletes transactions before a specified time stamp from the state database.

```
Ctx.ERC1155Transaction.DeleteHistoricalTransactions(referenceTime string)
(interface{}, error)
```

Parameters:

- `referenceTime string` – All transactions before this time stamp will be deleted.

Return Value

```
{
    "Transactions": [

"otransaction~750f68538451847f57948f7d5261dcb81570cd9e429f928a4cb7bfa76392ecf7
"
    ],
    "msg": "Successfuly deleted transaction older than
date:2022-04-06T08:17:53Z"
}
```

Methods for Token Behavior Management - Mintable Behavior

`MintBatch`
This method creates (mints) multiple tokens in a batch operation. This method creates only fungible tokens or fractional non-fungible tokens.
For fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. You cannot mint more than the `max_mint_quantity` property of the token, if that property was specified when the token was created or updated.
For non-fungible tokens, if the minter role is defined in the specification file, then any user with the minter role can call this method. If not, any user can use this method to mint tokens. Additionally, the caller must also be the creator of the token. There is no upper limit to the quantity of fractional non-fungible tokens that can be minted.
You cannot use this method to mint a whole non-fungible token.

```
Ctx.ERC1155Token.MintBatch(accountId string, tokens []interface{}, quantities
[]float64) (interface{}, error)
```

Parameters:

- `accountId string` – The account ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenIds []string` – The list of token IDs to mint tokens for.
- `quantity []float64` – The list of quantities of tokens to mint, corresponding to the token ID array.

Returns:

- On success, a JSON object that includes details on the minted tokens.

Return Value

```
{
    "details": [
        {
            "msg": "Successfully minted 100 tokens of fractional tokenId:
plot55 to Org-Id: appdev, User-Id: idcqa"
        },
        {
            "msg": "Successfully minted 100 tokens of tokenId: 'loyalty' to
Token-Account-Id
'oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e'"
        }
    ],
    "msg": "Successfully minted batch of tokens for User-Account-Id
'ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38'
(Org-Id: 'appdev', User-Id: 'idcqa')"
}
```

Methods for Token Behavior Management - Transferable Behavior

`BatchTransferFrom`
This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.
For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.
For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.
This method does not validate that the caller of the method is the specified sender.

```
Ctx.ERC1155Token.BatchTransferFrom(fromAccountId string, toAccountId string,
tokens []interface{}, quantities []float64) (interface{}, error)
```

Parameters:

- `fromUserAccountId string` – The account ID of the sender and token owner in the current organization.
- `toUserAccountId string` – The account ID of the receiver.
- `tokenIds string[]` – A list of token IDs for the tokens to transfer.
- `quantity float64[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message with details for each token transfer.

Return Value

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`SafeBatchTransferFrom`
This method completes a batch operation that transfers tokens specified in a list of token IDs from one user to another user.
For NFTs, because the method transfers ownership of the NFT, the sender of the NFT must own the token.
For fractional NFTs, if a user (including the creator of the token) transfers all of the shares that they own, then they lose ownership of the token. If any share of a token is transferred to a user, that user automatically becomes one of the owners of the fractional NFT.
The caller of the method must be the specified sender.

```
Ctx.ERC1155Token.SafeBatchTransferFrom(fromAccountId string, toAccountId
string, tokens []interface{}, quantities []float64) (interface{}, error)
```

Parameters:

- `fromUserAccountId string` – The account ID of the sender and token owner in the current organization.
- `toUserAccountId string` – The account ID of the receiver.
- `tokenIds string[]` – A list of token IDs for the tokens to transfer.
- `quantity float64[]` – The list of quantities of tokens to transfer, corresponding to the token ID array.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message with details for each token transfer.

Return Value

```
[
    {
        "msg": "Successfully transferred NFT token: 'FNFT' of '10' quantity
from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred 10 FT token: 'FT' from Account-Id:
oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c
(Org-Id: appdev, User-Id: user1_minter)"
    },
    {
        "msg": "Successfully transferred NFT token: 'NFT' from Account-Id:
oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371
(Org-Id: appdev, User-Id: idcqa) to Account-Id:
oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446
(Org-Id: appdev, User-Id: user1_minter)"
    }
]
```

`BalanceOfBatch`
This method completes a batch operation that gets the balance of token accounts. The account details are specified in three separate lists of organization IDs, user IDs, and token IDs. This method can be called only by a `Token Admin` of the chaincode or by account owners. Account owners can see balance details only for accounts that they own.

```
Ctx.ERC1155Account.BalanceOfBatch(accountIds []string, tokens []interface{})
(interface{}, error)
```

Parameters:

- `accountIds []string` – A list of the user account IDs.
- `tokenIds []string` – A list of the token IDs.

Return Value

```
[
    {
        "OrgId": "appdev",
        "UserId": "idcqa",
        "UserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "TokenAccountId":
"oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "TokenId": "FNFT",
        "Balance": 100
    },
    {
        "OrgId": "appdev",
        "UserId": "idcqa",
        "UserAccountId":
"ouaccount~412de5e3998dcd100973e1bad6e8729fddc1c7ff610beab8376d733a35c51f38",
        "TokenAccountId":
"oaccount~21206f309941a2a23c4f158a0fe1b8f12bb8e2b0c9a2e1358f5efebc0c7d410e",
        "TokenId": "FT",
        "Balance": 50
    },
    {
        "OrgId": "appdev",
        "UserId": "user1_minter",
        "UserAccountId":
"ouaccount~9501bb774c156eb8354dfe489250ea91f757523d70f08ee494bda98bb352003b",
        "TokenAccountId":
"oaccount~dcee860665db8740cb77b846e823752185a1e9a185814d0acb305890f5903446",
        "TokenId": "FNFT",
        "Balance": 10
    }
]
```

`ExchangeToken`
This method exchanges tokens between specified accounts. This method only supports exchanging between an NFT (whole or fractional) and a fungible token or a fungible token and an NFT (whole or fractional). This method can be called only by the account owner.

```
Ctx.ERC1155Token.ExchangeToken(fromTokenId string, fromUserAccountId string,
fromTokenQuantity float64, toTokenId string, toUserAccountId string,
toTokenQuantity float64) (interface{}, error)
```

Parameters:

- `fromTokenId string` – The ID of the token that the sender owns.
- `fromUserAccountId string` – The account ID of the sender.
- `fromTokenQuantity float64` – The quantity of tokens from the sender to exchange with the receiver.
- `toTokenId string` – The ID of the token that the receiver owns.
- `toUserAccountId string` – The account ID of the receiver.
- `toTokenQuantity float64` – The quantity of tokens from the receiver to exchange with the sender.

Returns:

- On success, a message with token exchange details.

Copyright © 2025, Oracle and/or its affiliates.

Return Value

```
{
    "msg": "Succesfully exchanged 10 tokens of type 'nonfungible' with
tokenId: [r1] from Account
'oaccount~e88276a3be547e31b567346bdddde52d37734da4d5fae83ab2e5c98a10097371'
(OrgId: appdev, UserId: idcqa) to 10 tokens of type 'fungible' with tokenId:
[loy1] from Account
'oaccount~1089ee5122f367ee0ca38c6660298f4b81f199627e4f67f3691c0f628237974c'
(OrgId: 'appdev', UserId: 'user1_minter')"
}
```

Methods for Token Behavior Management - Burnable Behavior

`Burn`
This method deactivates, or burns, the specified fungible and non-fungible tokens.

```
Ctx.ERC1155Token.Burn(accountId string, tokens []interface{}, quantities
[]float64) (interface{}, error)
```

Parameters:

- `accountId string` – The account ID of the user.
- `tokenIds []string` – The list of token IDs to burn.
- `quantity []float64` – The list of quantities of tokens to burn, corresponding to the token ID array.

Returns:

- On success, a message with details about the burn operations.

Return Value

```
[
    {
        "msg": "Successfully burned NFT token: 'art' from Account-Id:
oaccount~76cb672eeb1bd535899562a840d0c15a356db89e24bc8b43ac1dba845a4282c6
(Org-Id: appdev, User-Id: idcqa)"
    },
    {
        "msg": "Successfully burned 2 tokens of tokenId: FT from Account-ID
oaccount~9a940587fd322ccc8400233244cd3b13f3aa2a52e418e4c71fb819a2217bc49e
(Org-Id: AutoF1377358917, User-Id: idcqa)"
    },
    {
        "msg": "Successfully burned 2 token share of tokenId: FNFT from
Account-ID
oaccount~87bcb699d507368ee3966cd03ee6d7736ffc55dde8c0f0e16b14866334ac504a
(Org-Id: AutoF1377358917, User-Id: idcqa)"
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

Go Methods for ERC-1155 NFT Locking

Blockchain App Builder automatically generates methods that you can use to lock non-fungible tokens that use the extended ERC-1155 standard.

A locked token cannot be burned or transferred to other users. All other properties, such as the token's state, owner, and history are preserved. You can use the NFT locking functionality when transferring a token to another blockchain network, such as Ethereum or Polygon.

Before you can lock NFTs, you must assign the vault manager role to a user. The vault manager is a special type of role, a `TokenSys` role. `TokenSys` roles are different from asset-based roles such as burner, minter, and notary, and from administrative roles such as `Token Admin` and `Org Admin`. Currently Blockchain App Builder supports the `vault` `TokenSys` role. The single user who has the `vault` role for a chaincode is the vault manager of the chaincode, and can manage locked NFTs.

The typical flow for using the NFT locking functionality follows these steps.

- Create a non-fungible token that has the lockable behavior.
- Use the `AddTokenSysRole` method to give the `vault` role to a user, the vault manager.
- Call the `LockNFT` method to lock a non-fungible token, specified by the token ID.

TokenSys Role Management Methods

`AddTokenSysRole`
This method adds a `TokenSys` role to a specified user. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) AddTokenSysRole(orgId string, userId string, role
string) (interface{}, error) {
    userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.AddTokenSysRole", "TOKEN",
map[string]string{"accountId": userAccountId})
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC1155Token.AddTokenSysRoleMember(role, userAccountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the `TokenSys` role to give to the user.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully added role 'vault' to Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

`IsInTokenSysRole`
This method returns a Boolean value to indicate if a user has a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) IsInTokenSysRole(orgId string, userId string, role
string) (interface{}, error) {
    userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.IsInTokenSysRole", "TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC1155Token.IsInTokenSysRoleMember(role, userAccountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `role: string` – The name of the `TokenSys` role to check.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "result": true,
    "msg": "Account Id
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1) has vault role"
}
```


Copyright © 2025, Oracle and/or its affiliates.

RemoveTokenSysRole
This method removes a TokenSys role from a specified user. This method can be called only by a Token Admin of the chaincode.

```
func (t *Controller) RemoveTokenSysRole(orgId string, userId string, role
string) (interface{}, error) {
    userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
    if err != nil {
        return nil, err
    }
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.RemoveTokenSysRole",
"TOKEN", map[string]string{"accountId": userAccountId})
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC1155Token.RemoveTokenSysRoleMember(role, userAccountId)
}
```

Parameters:

- orgId: string – The membership service provider (MSP) ID of the user in the current organization.
- userId: string – The user name or email ID of the user.
- role: string – The name of the TokenSys role to remove.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully removed role 'vault' from Account Id:
oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba
(Org-Id: Org1MSP, User-Id: user1)"
}
```

TransferTokenSysRole
This method transfers a TokenSys role from a user to another user. This method can be called only by a Token Admin of the chaincode.

```
func (t *Controller) TransferTokenSysRole(role string, fromOrgId string,
fromUserId string, toOrgId string, toUserId string) (interface{}, error) {
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.TransferTokenSysRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
```


Copyright © 2025, Oracle and/or its affiliates.

```
    }
    fromUserAccountId, err :=
t.Ctx.ERC1155Account.GenerateAccountId(fromOrgId, fromUserId,
constants.UserAccount)
    if err != nil {
        return nil, fmt.Errorf("error in TransferTokenSysRole. Error: %s",
err)
    }
    toUserAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(toOrgId,
toUserId, constants.UserAccount)
    if err != nil {
        return nil, fmt.Errorf("error in TransferTokenSysRole. Error: %s",
err)
    }
    return t.Ctx.ERC1155Token.TransferTokenSysRole(role, fromUserAccountId,
toUserAccountId)
}
```

Parameters:

- `fromOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role from.
- `fromUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role from.
- `toOrgId: string` – The membership service provider (MSP) ID of the user to transfer the `TokenSys` role to.
- `toUserId: string` – The user name or email ID of the user to transfer the `TokenSys` role to.
- `role: string` – The name of the `TokenSys` role to transfer.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "msg": "Successfully transfered role 'vault' from Account Id:
ouaccount~f4e311528f03fffa7810753d643f66289ff6c9080fcf839902f28a1d3aff1789
(Org-Id: Org1MSP, User-Id: user1) to Account Id:
ouaccount~ae5be2ae8f98d6d32f5d02b43877d987114e7937c7bacbc30390dcce09996a19
(Org-Id: Org1MSP, User-Id: user2)"
}
```

`GetAccountsByTokenSysRole`
This method returns a list of all account IDs for a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetAccountsByTokenSysRole(role string) (interface{},
error) {
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Role.GetAccountsByTokenSysRole",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC1155Token.GetAccountsByTokenSysRole(role)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
    "accountIds": [

"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2caf03eba"
    ]
}
```

`GetUsersByTokenSysRole`
This method returns user information for all users with a specified `TokenSys` role. This method can be called only by a `Token Admin` of the chaincode.

```
func (t *Controller) GetUsersByTokenSysRole(role string) (interface{}, error)
{
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Role.GetUsersByTokenSysRole",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    return t.Ctx.ERC1155Token.GetUsersByTokenSysRole(role)
}
```

Parameters:

- `role: string` – The name of the `TokenSys` role.

Returns:

- On success, a message that contains relevant details of the operation.

Return Value Example:

```
{
   "Users":[
      {
```


Copyright © 2025, Oracle and/or its affiliates.

```
"accountId":"oaccount~bf07f584a94be44781e49d9101bfaf58c6fbbe77a4dfebdb83c874c2
caf03eba",
         "orgId":"Org1MSP",
         "userId":"user1"
      }
   ]
}
```

NFT Locking Methods

`LockNFT`
This method locks a specified non-fungible token. To lock a token, there must be a user with the `TokenSys vault` role, who acts as the vault manager. This method can be called only by the owner of the token.

```
func (t *Controller) LockNFT(orgId string, userId string, tokenId string)
(interface{}, error) {
    return t.Ctx.ERC1155Token.LockNFT(orgId, userId, tokenId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user (optional).
- `tokenID: string` – The ID of the token to lock.

Returns:

- On success, a JSON representation of the token object.

Return Value Example:

```
{
   "AssetType":"otoken",
   "Behaviors":[
      "indivisible",
      "mintable",
      "transferable",
      "burnable",
      "lockable",
      "roles"
   ],

"CreatedBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
   "CreationDate":"2023-10-20T12:22:47Z",
   "IsBurned":false,
   "IsLocked":true,
   "Mintable":{
      "Max_mint_quantity":20000
   },
```


Copyright © 2025, Oracle and/or its affiliates.

```
   "On_sale_flag":false,

"Owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
   "Price":120,
   "Quantity":1,
   "Roles":{
      "minter_role_name":"minter"
   },
   "TokenDesc":"",
   "TokenId":"token1",
   "TokenMetadata":{
      "Description":"",
      "Image":"",
      "Painter_name":"",
      "Painting_name":""
   },
   "TokenName":"artcollection",
   "TokenStandard":"erc1155+",
   "TokenType":"nonfungible",
   "TokenUnit":"whole",
   "TokenUri":"token1.example.com"
}
```

`IsNFTLocked`
This method returns a Boolean value to indicate if a specified token is locked. This method can be called only by the token owner, the vault manager (the user with the `TokenSys vault` role), or a `Token Admin` of the chaincode.

```
func (t *Controller) IsNFTLocked(tokenId string) (interface{}, error) {
        auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.IsNFTLocked", "TOKEN",
map[string]string{"tokenId": tokenId})
        if err != nil && !auth {
            isCallerTokenSysRoleHolder, error :=
t.Ctx.ERC1155Token.IsCallerTokenSysRoleHolder(constants.Vault)
            if error != nil {
                return nil, error
            }
            if !isCallerTokenSysRoleHolder {
                return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
            }
        }
        return t.Ctx.ERC1155Token.IsNFTLocked(tokenId)
    }
```

Parameters:

- `tokenID: string` – The ID of the token.

Returns:

- On success, a message that contains relevant details of the operation.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
   "isNFTLocked":true
}
```

`GetAllLockedNFTs`
This method returns a list of all locked non-fungible tokens. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
func (t *Controller) GetAllLockedNFTs() (interface{}, error) {
        auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.GetAllLockedNFTs", "TOKEN")
        if err != nil && !auth {
            isCallerTokenSysRoleHolder, error :=
t.Ctx.ERC1155Token.IsCallerTokenSysRoleHolder(constants.Vault)
            if error != nil {
                return nil, error
            }
            if !isCallerTokenSysRoleHolder {
                return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
            }
        }
        return t.Ctx.ERC1155Token.GetAllLockedNFTs()
    }
```

Parameters:

- None

Returns:

- On success, an array of the locked non-fungible token objects.

Return Value Example:

```
[
   {
      "key":"token1",
      "valueJson":{
         "AssetType":"otoken",
         "Behaviors":[
            "indivisible",
            "mintable",
            "transferable",
            "burnable",
            "lockable",
            "roles"
         ],

"CreatedBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "CreationDate":"2023-10-20T12:22:47Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
         "IsBurned":false,
         "IsLocked":true,
         "Mintable":{
            "Max_mint_quantity":20000
         },
         "On_sale_flag":false,

"Owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "Price":120,
         "Quantity":1,
         "Roles":{
            "minter_role_name":"minter"
         },
         "TokenDesc":"",
         "TokenId":"token1",
         "TokenMetadata":{
            "Description":"",
            "Image":"",
            "Painter_name":"",
            "Painting_name":""
         },
         "TokenName":"artcollection",
         "TokenStandard":"erc1155+",
         "TokenType":"nonfungible",
         "TokenUnit":"whole",
         "TokenUri":"token1.example.com"
      }
   }
]
```

`GetAllLockedNFTsByOrg`

This method returns a list of all locked non-fungible tokens for a specified organization and optionally a specified user. This method can be called only by the vault manager (the user with the `TokenSys vault` role) or a `Token Admin` of the chaincode.

```
func (t *Controller) GetLockedNFTsByOrg(orgId string, userId ...string)
(interface{}, error) {
        auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155Token.GetLockedNFTsByOrg",
"TOKEN")
        if err != nil && !auth {
            isCallerTokenSysRoleHolder, error :=
t.Ctx.ERC1155Token.IsCallerTokenSysRoleHolder(constants.Vault)
            if error != nil {
                return nil, error
            }
            if !isCallerTokenSysRoleHolder {
                return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
            }
        }
```


Copyright © 2025, Oracle and/or its affiliates.

```
        return t.Ctx.ERC1155Token.GetLockedNFTsByOrg(orgId, userId...)
    }
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user (optional).

Returns:

- On success, an array of the locked non-fungible token objects.

Return Value Example:

```
[
   {
      "key":"token1",
      "valueJson":{
         "AssetType":"otoken",
         "Behaviors":[
            "indivisible",
            "mintable",
            "transferable",
            "burnable",
            "lockable",
            "roles"
         ],

"CreatedBy":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6
a7733463",
         "CreationDate":"2023-10-20T12:22:47Z",
         "IsBurned":false,
         "IsLocked":true,
         "Mintable":{
            "Max_mint_quantity":20000
         },
         "On_sale_flag":false,

"Owner":"oaccount~208e3345ac84b4849f0d2648b2f2f018019886a1230f99304ebff1b6a773
3463",
         "Price":120,
         "Quantity":1,
         "Roles":{
            "minter_role_name":"minter"
         },
         "TokenDesc":"",
         "TokenId":"token1",
         "TokenMetadata":{
            "Description":"",
            "Image":"",
            "Painter_name":"",
            "Painting_name":""
         },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "TokenName":"artcollection",
            "TokenStandard":"erc1155+",
            "TokenType":"nonfungible",
            "TokenUnit":"whole",
            "TokenUri":"token1.example.com"
         }
      }
]
```

Go Methods for ERC-1155 Token Account Status

Blockchain App Builder automatically generates methods that you can use to manage account status for tokens that use the extended ERC-1155 standard.

You can use the following methods to put token user accounts in the active, suspended, or deleted states.

When an account is suspended, the account user cannot complete any write operations, which include minting, burning, and transferring tokens. Additionally, other users cannot transfer tokens to a suspended account. A suspended account can still complete read operations.

An account with a non-zero token balance cannot be deleted. You must transfer or burn all tokens in an account before you can delete the account. After an account is in the deleted state, the account state cannot be changed back to active or suspended.

- Automatically Generated Account Status Methods
- Account Status SDK Methods

Automatically Generated Account Status Methods

`GetAccountStatus`
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
func (t *Controller) GetAccountStatus(orgId string, userId string,
tokenId ...string) (interface{}, error) {
      userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating accountId
of (Org-Id: %s, User-Id: %s)", orgId, userId)
      }
      auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155AccountStatus.Get", "TOKEN",
map[string]string{"accountId": userAccountId})
      if err != nil && !auth {
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId, tokenId...)
      if err != nil {
            return nil, fmt.Errorf("error in GetAccountStatus. Error: %s",
err)
      }
```


Copyright © 2025, Oracle and/or its affiliates.

```
      tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
      accountStatus, err :=
t.Ctx.ERC1155AccountStatus.GetAccountStatus(tokenAccountId)
      if err != nil {
            return
t.Ctx.ERC1155AccountStatus.GetDefaultAccountStatus(tokenAccountId)
      }
      return accountStatus, nil
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`GetAccountStatusHistory`
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode or by the token account owner.

```
func (t *Controller) GetAccountStatusHistory(orgId string, userId string,
tokenId ...string) (interface{}, error) {
      userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
      if err != nil {
            return nil, fmt.Errorf("error in getting the generating accountId
of (Org-Id: %s, User-Id: %s)", orgId, userId)
      }
      auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155AccountStatus.Get", "TOKEN",
map[string]string{"accountId": userAccountId})
      if err != nil && !auth {
```


Copyright © 2025, Oracle and/or its affiliates.

```
            return nil, fmt.Errorf("error in authorizing the caller %s",
err.Error())
      }
      tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId, tokenId...)
      if err != nil {
            return nil, fmt.Errorf("error in GetAccountStatusHistory. Error:
%s", err)
      }
      tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
      statusId, err :=
t.Ctx.ERC1155AccountStatus.GenerateAccountStatusId(tokenAccountId)
      if err != nil {
            return nil, err
      }
      accountStatusHistory, err :=
t.Ctx.ERC1155AccountStatus.History(statusId)
      if err != nil {
            return []map[string]interface{}{}, nil
      }
      return accountStatusHistory, nil
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, the account status history in JSON format.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

ActivateAccount
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode. Deleted accounts cannot be activated. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
func (t *Controller) ActivateAccount(orgId string, userId string,
tokenId ...string) (interface{}, error) {
    userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
    if err != nil {
        return nil, fmt.Errorf("error in getting the generating accountId of
(Org-Id: %s, User-Id: %s)", orgId, userId)
    }
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155AccountStatus.ActivateAccount",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId, tokenId...)
    if err != nil {
        return nil, fmt.Errorf("error in GetAccountTransactionHistory. Error:
%s", err)
    }
    tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
    return t.Ctx.ERC1155Account.ActivateAccount(tokenAccountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

`SuspendAccount`
This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
func (t *Controller) SuspendAccount(orgId string, userId string,
tokenId ...string) (interface{}, error) {
    userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
    if err != nil {
        return nil, fmt.Errorf("error in getting the generating accountId of
(Org-Id: %s, User-Id: %s)", orgId, userId)
    }
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155AccountStatus.SuspendAccount",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId, tokenId...)
    if err != nil {
        return nil, fmt.Errorf("error in GetAccountTransactionHistory. Error:
%s", err)
    }
    tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
    return t.Ctx.ERC1155Account.SuspendAccount(tokenAccountId)
}
```

Parameters:

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.

Copyright © 2025, Oracle and/or its affiliates.

- `userId: string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

`DeleteAccount`
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance must be zero.

```
func (t *Controller) DeleteAccount(orgId string, userId string,
tokenId ...string) (interface{}, error) {
    userAccountId, err := t.Ctx.ERC1155Account.GenerateAccountId(orgId,
userId, constants.UserAccount)
    if err != nil {
        return nil, fmt.Errorf("error in getting the generating accountId of
(Org-Id: %s, User-Id: %s)", orgId, userId)
    }
    auth, err :=
t.Ctx.ERC1155Auth.CheckAuthorization("ERC1155AccountStatus.DeleteAccount",
"TOKEN")
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    tokenAccount, err := t.Ctx.ERC1155Account.Get(userAccountId, tokenId...)
    if err != nil {
        return nil, fmt.Errorf("error in GetAccountTransactionHistory. Error:
%s", err)
    }
    tokenAccountId, err := util.GetAccountProperty(tokenAccount,
constants.AccountId)
    return t.Ctx.ERC1155Account.DeleteAccount(tokenAccountId)
}
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `orgId: string` – The membership service provider (MSP) ID of the user in the current organization.
- `userId: string` – The user name or email ID of the user.
- `tokenId ...string` – For a non-fungible token account, an empty string. For a fungible token account, the token ID.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```

Account Status SDK Methods

`GetDefaultAccountStatus`
This method gets the current status of a token account, with the status as `active` for any account that does not have account status stored in the ledger (because the account was created prior to the account status functionality).

```
Ctx.GetDefaultAccountStatus(accountId string) (ERC1155AccountStatus, error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the token account status.

Return Value Example:

```
{
      "AssetType": "oaccountStatus",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "Status": "active"
}
```


Copyright © 2025, Oracle and/or its affiliates.

GetAccountStatus
This method gets the current status of the token account.

```
Ctx.ERC1155AccountStatus.GetAccountStatus(accountId string)
(NFTAccountStatus, error)
```

Parameters:

- `accountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the token account status. If no status is found in the ledger for the account because the account was created before the account status functionality was available, the status is listed as `active` in the response.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```

GetAccountStatusHistory
This method gets the history of the account status.

```
Ctx.ERC1155AccountStatus.History(statusId string) (interface{}, error)
```

Parameters:

- `statusId: string` – The ID of the account status object.

Returns:

- On success, a JSON representation of the account status history.

Return Value Example:

```
[
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:20:34+05:30",
    "TxId":
"af1601c7a14b4becf4bb3b285d85553b39bf234caaf1cd488a284a31a2d9df78",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "suspended",
```


Copyright © 2025, Oracle and/or its affiliates.

```
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  },
  {
    "IsDelete": "false",
    "Timestamp": "2022-12-02T16:19:15+05:30",
    "TxId":
"4b307b989063e43add5357ab110e19174d586b9746cc8a30c9aa3a2b0b48a34e",
    "Value": {
      "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "AssetType": "oaccountStatus",
      "Status": "active",
      "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7"
    }
  }
]
```

`ActivateAccount`
This method activates a token account. For any accounts created prior to the account status functionality being available, you must call this method to set the account status as `active`.

```
Ctx.ERC1155Account.ActivateAccount(tokenAccountId string) (interface{}, error)
```

Parameters:

- `tokenAccountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "active"
}
```


Copyright © 2025, Oracle and/or its affiliates.

SuspendAccount
This method suspends a token account.

```
Ctx.ERC1155Account.SuspendAccount(tokenAccountId string) (interface{}, error)
```

Parameters:

- `tokenAccountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "suspended"
}
```

DeleteAccount
This method deletes a token account.

```
Ctx.ERC1155Account.DeleteAccount(tokenAccountId string) (interface{}, error)
```

Parameters:

- `tokenAccountId: string` – The ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "AssetType": "oaccountStatus",
    "StatusId":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "AccountId":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "Status": "deleted"
}
```


Copyright © 2025, Oracle and/or its affiliates.

Deploying and Testing Token Chaincode

You deploy token chaincode projects by following the same steps as other chaincode projects, but there are some special issues to consider.

You can deploy chaincode to only one organization or instance when you use the standard deployment steps with Blockchain App Builder. When a token chaincode is deployed, the list of `Token Admin` users is specified. The `Token Admin` user can add or remove other users by calling the `addAdmin` and `removeAdmin` methods.

To deploy to multiple organizations or instances, package the chaincode and then manually deploy it to all the instances. For more information, see the applicable following topic:

- CLI: Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network
- Visual Studio Code: Deploy Your Chaincode to a Remote Oracle Blockchain Platform Network

To test token projects locally with multiple users, see the applicable following topic:

- CLI: Testing Multiple Token Users Locally
- Visual Studio Code: Testing Multiple Token Users Locally

To test token projects on a remote Oracle Blockchain Platform network, see the applicable following topic:

- CLI: Testing Token Projects on a Remote Oracle Blockchain Platform Network
- Visual Studio Code: Testing Token Projects on a Remote Oracle Blockchain Platform Network

Adding Enrollments for Token Use Cases

Oracle Blockchain Platform supports enrollments to the REST proxy. You use enrollments with token chaincodes to ensure the identities of the users completing token transactions. To do this, when you add enrollments for token use cases, specify a user ID for each enrollment, and specify one and only one user for each enrollment.

Working With the Sample Token Specification Files

You can use the sample token specification files that come with Blockchain App Builder to investigate the complete life cycle of a token.

To get the Blockchain App Builder samples, in the service console open the **Developer Tools** tab and then select the **Blockchain App Builder** pane. Click **Download Specification Samples and Related Code** and then extract the downloaded archive file (`obp-app-builder-samples.zip`). The archive file includes token specification files for the Fiat Money Token, Loyalty Token, NFT Art Collection Marketplace, and Fractional NFT in Real Estate samples.

For the Fiat Money Token and Loyalty Token samples, the archive file also includes scaffolded token chaincode in Go and a corresponding Postman collection file (specific to Go), which you can use to test token life cycle operations on an Oracle Blockchain Platform network.

The NFT Art Collection Marketplace and Fractional NFT in Real Estate samples include specification files that are designed for development in TypeScript. The NFT Art Collection Marketplace sample includes scaffolded token chaincode in TypeScript and a corresponding Postman collection file (specific to TypeScript), which you can use to test token life cycle

Copyright © 2025, Oracle and/or its affiliates.

operations for a non-fungible token on an Oracle Blockchain Platform network. There are two NFT Art Collection Marketplace samples: one for the ERC-721 standard and one for the ERC-1155 standard.

The token asset name is specified in the sample specification file. A token is defined by behaviors, which are also in the specification file. The sample token assets also include custom properties.

Oracle Blockchain Platform Digital Assets Edition includes additional samples that support wholesale central bank digital currency (CBDC), bond marketplaces, and generic token frameworks. For more information about those samples, see Oracle Blockchain Platform Digital Assets Edition.

For more information about using the token samples, see the `readme.md` file in the sample archive file.

For more information about creating specification files in Blockchain App Builder, see the following topics:

- Input Specification File
- Input Specification File for Token Taxonomy Framework
- Input Specification File for ERC-721
- Input Specification File for ERC-1155

For more information about deploying and testing chaincode projects, see Deploying and Testing Token Chaincode.

Disaster Recovery Support for Tokenization

You can configure Oracle Blockchain Platform for disaster recovery support in tokenization scenarios.

Before you can set up disaster recovery support in a tokenization scenario, the blockchain network must be configured for disaster recovery, using the following guidelines. For more information, see Add OBCS Participant Organizations to the Network.

- In addition to the founder instance of Oracle Blockchain Platform, create at least two more participant instances and join them to the blockchain network.
- Add at least two orderers from each participant instance to the blockchain network. Typically you add at least three orderers from each participant instance.
- Join the participant organizations to the same channels and install chaincodes on the participant instances.
- Ensure that the orderers from the participant instances are joined to the channels on the founder instance.

You can then prepare for disaster recovery in a tokenization scenario by designating a second instance of Oracle Blockchain Platform as a disaster recovery organization. When a primary Oracle Blockchain Platform instance is down and unable to send transactions to the chaincode, a secondary Oracle Blockchain Platform instance set up as a disaster recovery organization can be used to send the transaction to the chaincode on behalf of the primary instance.

In the following example, the founder organization on the primary instance is Org1MSP and the participant organization on the disaster recovery instance is Org2MSP.

Copyright © 2025, Oracle and/or its affiliates.

To set up a second instance for disaster recovery in tokenization scenarios, complete the following steps.

1. Create custom enrollments on the disaster recovery organization and add a custom attribute for `primaryOrgMSPId` that is the ID of the founder organization (Org1MSP). You can use a tool such as Postman to create the custom enrollment. This enrollment indicates that Org2MSP is a disaster recovery organization for the primary organization, Org1MSP.
   The following example shows an example request body for the REST endpoint `{{bc-url}}/console/admin/api/v2/nodes/restproxies/{{bc-restproxy-id}}/enrollments`.
   You can get the `bc-restproxy-id` value in the response from a GET request for the REST endpoint `{{bc-url}}/console/admin/api/v2/nodes`.

   ```
   {
      "enrollmentId": "<enrollmentId>",
       "attributes":{
           "primaryOrgMSPId": "<primaryOrgId>"
       }
    }
   ```

   The following example is the expected response.

   ```
   {
   "respMesg": "SUCCESS"
   }
   ```

2. Add users to the custom enrollment. You can use Postman or the Oracle Blockchain Platform console to add users. The enrollment IDs and user names in the disaster recovery organization (Org2MSP) must match the enrollment IDs and user names in the primary organization (Org1MSP). The only difference is the custom attribute, `primaryOrgMSPId`, which points to the primary organization (Org1MSP).
   The following example shows an example request body for the REST endpoint `{{bc-url}}/console/admin/api/v2/nodes/restproxies/{{bc-restproxy-id}}/enrollments/{{bc-enrolment-id}}/users` .

   ```
   {
       "userName": "<userId>"
    }
   ```

   The following example is the expected response.

   ```
   {
   "respMesg": "SUCCESS"
   }
   ```

Digital Assets Edition Enhancements

Oracle Blockchain Platform Digital Assets Edition includes an enhanced version of Blockchain App Builder.

The following functions are supported by the enhanced version of Blockchain App Builder.

- Automatic generation of complete chaincode applications for wholesale CBDC and bond marketplace scenarios.

Copyright © 2025, Oracle and/or its affiliates.

- Automatic generation of wrapper API packages for API Gateway, with dedicated endpoints for each chaincode method. You can configure the names of the endpoints to match your organization's naming conventions.
- Support for chaincode events, which can send real-time notifications and trigger workflows.
- Support for endorsement parameters in generated Postman collections.
- Support for chaincodes that use the extended Token Taxonomy Framework standard to restrict confidential or sensitive data so that it is accessible only to the organizations involved in the transaction.
- New account, role, and transaction functions in the extended Token Taxonomy Framework standard.
- Enhancements to the CLI and the Visual Studio Code extension UI.

CLI Enhancements

The `ochain invoke` command supports the following additional optional arguments.

Transient Map
Option: `-t, --transient`

Specifies key/value pairs for the transient map as a JSON string.

Example:

```
ochain invoke <method> 's01' 's10' \
  -r <remote OBP URL> -u <username> -s <password> \
  -t '{"key": "value"}'
```

Headers
Option: `-H, --headers`

Specifies additional headers as key/value pairs in JSON format. The `Content-Type: application/json` header is set by default and cannot be overridden. Do not include it in custom headers.

Example:

```
ochain invoke <method> 's01' 's10' \
  -r <remote OBP URL> -u <username> -s <password> \
  -H '{"confidential-transaction": "true"}'
```

Endorsers
Option: `-e, --endorsers`

Specifies endorser peer URLs to use as a JSON array of strings. You cannot use this option in combination with the `-S, --sameorgendorser` option to use endorsers from the same organization.

Example:

```
ochain invoke <method> 's01' 's10' \
  -r <remote OBP URL> -u <username> -s <password> \
```


Copyright © 2025, Oracle and/or its affiliates.

```
  -e '["org1-xyz-abc.blockchain.ocp.oraclecloud.com:20009", "org1-xyz-
abc.blockchain.ocp.oraclecloud.com:20010"]'
```

Same Organization Endorsers
Option: `-S, --sameorgendorser`

A Boolean flag that indicates whether to use endorsers from the same organization. You cannot use this option in combination with the `-e, --endorsers` option to specify endorsers.

Example:

```
ochain invoke <method> 's01' 's10' \
  -r <remote OBP URL> -u <username> -s <password> \
  -S true
```

The following example shows all of the optional parameters:

```
ochain invoke <method> 's01' 's10' \
  -r <remote OBP URL> -u <username> -s <password> \
  -H '{"confidential-transaction": "true"}' \
  -S false \
  -t '{"key": "value"}' \
  -e '["org1-xyz-abc.blockchain.ocp.oraclecloud.com:20009", "org1-xyz-
abc.blockchain.ocp.oraclecloud.com:20010"]'
```

Visual Studio Code Extension Enhancements

The options for transient maps, headers, endorsers, and same organization endorsers are available in the Execute pane for remote environments. When you select a remote environment, the **Show Advanced Options** control is available. Click **Show Advanced Options** to set these options for the currently selected method (function). If you select a different method, the advanced options are reset.

For each advanced option, you can add and remove values and select or clear the options to include when the method is called. The **Same Org Endorsers** checkbox is available on the Endorsing Peers dialog box. If you select **Same Org Endorsers**, any endorsing peers that you specified manually are ignored. Instead the transaction will be endorsed by peers from the same organization as the user.

For confidential chaincodes, the `Confidential-Transaction: true` header is set by default for all methods in the Visual Studio Code extension. You can edit this header.

Wrapper APIs

The enhanced version of Blockchain App Builder can generate wrapper APIs for API Gateway, which support dedicated endpoints for each chaincode method.

A wrapper API is an abstraction layer over an Oracle Blockchain Platform endpoint. Wrapper APIs support configuring common parameters and API-specific endpoint names, which can simplify usage and clarity.

In the base version of Oracle Blockchain Platform, an API endpoint requires parameters including the chaincode name, timeout and sync values, arguments (including the method name), channel name, and instance URL. These parameters must be passed with every invocation. Because most of these parameters are common to all methods, you can use the wrapper API functionality to create an endpoint for each method, which you then call using only

Copyright © 2025, Oracle and/or its affiliates.

the method parameters. Instead of a common endpoint, either a transaction or a query, you can have a custom endpoint name for each method. Whereas API endpoints in Oracle Blockchain Platform use POST requests, wrapper APIs support both POST and GET requests. You can also add an extra layer of authentication with wrapper APIs by using the API Gateway authentication mechanism.

The following example shows the createAccount method API in Oracle Blockchain Platform.

Endpoint: `https://<blockchain_instance>:7443/restproxy/api/v2/channels/<channelName>/transactions`

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "createAccount",
        "{{bc-org-id}}",
        "{{bc-user-id}}",
        "fungible",
        "{\"max_daily_amount\":10000,\"max_daily_transactions\":100}"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}}
}
```

If you configure wrapper APIs, you can make the same method call as shown in the following example.

Endpoint: `https://<gateway-hostname>/<deployment-path-prefix>/<route-path>`

```
{
    "orgId": "{{bc-org-id}}",
    "userId": "user1",
    "tokenType": "fungible",
    "dailyLimits":
"{\"max_daily_amount\":10000,\"max_daily_transactions\":100}",
}
```

Generate Wrapper APIs Using the CLI

The enhanced version of Blockchain App Builder includes the `generateFnAPI` command, which generates wrapper APIs for all controller APIs in a chaincode project.

```
Usage: ochain generateFnAPI [options]
Generation of WrapperAPI package for controller functions
Options:
-h, --help                  Output command usage information
-D, --debug                 Enable debug logging
-m, --mapping <mapping>     Info about functions endpoints mapping is
mandatory.
-c, --config <config>       Info about configuration variables is mandatory.
-a, --all <all>             Generate the wrapperAPI package for entire
controller functions
-p, --project <path>        Path to the Ochain chaincode project to generate
the WrapperAPI package. If not specified, it defaults to current directory.
```


Copyright © 2025, Oracle and/or its affiliates.

The following example shows how to generate wrapper APIs for a local chaincode project.

```
ochain generateFnAPI --mapping <info object about function mapping> --config
<info about the config variables> --project <Path to the Ochain chaincode
project> --out <Path to the generated postman collection>
```

When you run the `generateFnAPI` command, you are prompted whether to overwrite any previous wrapper API packages that exist in the selection location.

API Gateway limits the number of routes that can be created in a single deployment to 50. If you try to use the `generateFnAPI` command to generate more than 50 wrapper APIs, you are prompted whether to create multiple gateways. Creating multiple gateways with the same deployment path prefix maintains a consistent base path across all wrapper API endpoints. If you choose not to create multiple gateways, wrapper API generation is canceled. You can use the `gateway_ocids` parameter as described in the following information to reuse existing gateways.

Command Options

-m, --mapping <mapping>
The mapping option defines the configuration of controller APIs, specifying the endpoint names and which controller APIs require wrapper API generation. The map is a JSON object of key/value pairs, where the key is the name of the controller API and the value is the custom endpoint to associate with that controller API.
The following example shows a mapping for chaincode that uses the extended Token Taxonomy Framework standard.

```
ochain generateFnAPI --mapping '{"addConversionRate":"/
addConversionRateTest", "addTokenAdmin": "/addTokenAdminTest", "approveBurn":
"/approveBurnTest", "createAccount": "/createAccount",
"associateTokenToAccount": "/associateTokenToAccountTest",
"getAllOrgAccounts": "/getAllOrgAccounts"}' --config <info about the config
variables> --project <Path to the Ochain chaincode project> --out <Path to
the generated postman collection>
```

-c, --config <config>
The config option is used to pass the configuration variables that are required for the wrapper API package. It is a JSON object of key/value pairs, where the key is the configuration variable name and the value is the configuration variable value. All of the configuration variables except `gateway_ocids` are mandatory for generating a wrapper API package. If any of the following variables except `gateway_ocids` are not passed to the command, wrapper API generation fails. You can edit these values after you create the wrapper API by extracting the wrapper API `.zip` file.

Variable name	Description
`compartment_ocid`	The OCID of the compartment in Oracle Cloud Infrastructure (OCI).
`compartment_name`	The name of the OCI compartment.
`identity_domain`	The identity domain to use.
`blockchain_channel`	The name of the Oracle Blockchain Platform channel where the chaincode is deployed.

Copyright © 2025, Oracle and/or its affiliates.

Variable name	Description
`blockchain_url`	The Oracle Blockchain Platform URL associated with the chaincode deployment.
`blockchain_chaincode`	The name of the chaincode to generate wrapper APIs for.
`blockchain_sync`	The sync value to include in the payload for API calls.
`blockchain_timeout`	The timeout value to include in the payload for API calls.
`vcn_display_name`	The display name of the OCI virtual cloud network.
`application_display_name`	The display name of the OCI application.
`gateway_display_name`	The display name of API Gateway.
`deployment_display_name`	The display name of the deployment in API Gateway.
`deployment_path_prefix`	The deployment path prefix in API Gateway, which specifies the path where routes are deployed. The `deployment_path_prefix` variable must begin with a slash (/).
`ocir_repo_name`	The OCI Registry repository name. The `ocir_repo_name` variable must be all lowercase letters.
`policy_name`	The name of the policy that enables controlled management and access to APIs through defined permissions for groups and compartments within the organization
`gateway_ocids`	A JSON object of Oracle Cloud Identifiers (OCIDs) for existing API Gateways to reuse. You must specify one gateway for every 50 wrapper APIs. If you do not specify a `gateway_ocids` parameter, the command creates API Gateways.

The following example shows a command for chaincode that uses the extended Token Taxonomy Framework standard.

```
ochain generateFnAPI -m '{"addConversionRate":"/addConversionRateTest",
"addTokenAdmin": "/addTokenAdminTest", "approveBurn": "/approveBurnTest",
"createAccount": "/createAccount", "associateTokenToAccount": "/
associateTokenToAccountTest", "getAllOrgAccounts": "/getAllOrgAccounts"}' -c
'{"compartment_ocid": "compartment_ocid value", "compartment_name":
"compartment_name value", "identity_domain" : "OracleIdentityCloudService",
"blockchain_channel": "default", "blockchain_url": "blockchain_url value",
"blockchain_chaincode": "blockchain_chaincode value", "blockchain_sync":
true, "blockchain_timeout": 6000, "vcn_display_name": "vcn_display_name
value", "application_display_name": "application_display_name value",
"gateway_display_name": "gateway_display_name value",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"deployment_display_name": "deployment_display_name value",
"deployment_path_prefix": "deployment_path_prefix value", "ocir_repo_name":
"ocir_repo_name value", "policy_name": "policy_name value"}'
```

The following example shows a command that uses the `gateway_ocids` parameter.

```
ochain generateFnAPI -m '{"addConversionRate":"/addConversionRateTest",
"addTokenAdmin": "/addTokenAdminTest"}' -c
'{"compartment_ocid":"ocid1.compartment.oc1..xxxx","compartment_name":"MyCompa
rtment","identity_domain":"OracleIdentityCloudService","blockchain_channel":"d
efault","blockchain_url":"https://blockchain-
url","blockchain_chaincode":"my_chaincode","blockchain_sync":true,"blockchain_
timeout":6000,"vcn_display_name":"myVCN","application_display_name":"myApp","g
ateway_display_name":"myGateway","deployment_display_name":"myDeployment","dep
loyment_path_prefix":"myprefix","ocir_repo_name":"myRepo","policy_name":"myPol
icy","gateway_ocids":"{\"1\":\"ocid1.apigateway.oc1.xxx\",\"2\":\"ocid1.apigat
eway.oc1.yyy\"}"}'
```

-a, --all <all>
The all option generates wrapper API for all controller APIs that are defined in the chaincode. If `true`, this option overrides the mapping option. Every wrapper API is assigned a default endpoint, which corresponds to the API name. If you specify the all option, you must still provide an empty JSON object ({}) for the mapping option.
The following example shows a mapping for chaincode that uses the extended Token Taxonomy Framework standard.

```
ochain generateFnAPI -m '{}' -c '{"compartment_ocid": "compartment_ocid
value", "compartment_name": "compartment_name value", "identity_domain" :
"OracleIdentityCloudService", "blockchain_channel": "default",
"blockchain_url": "blockchain_url value", "blockchain_chaincode":
"blockchain_chaincode value", "blockchain_sync": true, "blockchain_timeout":
6000, "vcn_display_name": "vcn_display_name value",
"application_display_name": "application_display_name value",
"gateway_display_name": "gateway_display_name value",
"deployment_display_name": "deployment_display_name value",
"deployment_path_prefix": "deployment_path_prefix value", "ocir_repo_name":
"ocir_repo_name value", "policy_name": "policy_name value"}' -a true
```

Generate Wrapper APIs Using Visual Studio Code

The enhanced version of Blockchain App Builder enables you to generate wrapper APIs in Visual Studio Code.

When you generate wrapper APIs in Visual Studio Code, you are prompted whether to overwrite any previous wrapper API packages that exist in the selection location.

API Gateway limits the number of routes that can be created in a single deployment to 50. If you try to generate more than 50 wrapper APIs, you are prompted whether to create multiple gateways. Creating multiple gateways with the same deployment path prefix maintains a consistent base path across all wrapper API endpoints. If you choose not to create multiple gateways, wrapper API generation is canceled. You can also reuse existing gateways, as described in the following steps.

1. Select the chaincode project in the Chaincodes pane.

Copyright © 2025, Oracle and/or its affiliates.

2. Right-click the chaincode name and then select **Generate Wrapper API Package**. The Oracle Blockchain Platform - Wrapper API Package Generation pane opens.
3. Enter the configuration variables as described in the following table.

Field	Description	Default value
Compartment OCID	The OCID of the compartment in Oracle Cloud Infrastructure (OCI).	none
Compartment Name	The name of the OCI compartment.	none
Identity Domain	The identity domain to use.	none
Blockchain Channel	The name of the Oracle Blockchain Platform channel where the chaincode is deployed.	none
Blockchain URL	The Oracle Blockchain Platform URL associated with the chaincode deployment.	none
Location	The system directory to generate the wrapper API package in.	none

New API Gateway	To create API Gateways, select **Yes** and then specify a **Gateway Display Name**. To reuse existing API Gateways, select **No** and then specify an array of Oracle Cloud Identifiers (OCIDs) for existing API Gateways to reuse in the following format. You must specify one gateway for every 50 wrapper APIs.
	
	`["ocid1.apigateway.xxx"`
	`,`
	`"ocid1.apigateway.yyy"]`
		`Yes`

Deployment Path Prefix	The deployment path prefix in API Gateway, which specifies the path where routes are deployed. The deployment path prefix must begin with a slash (/).	`/<ChaincodeName>`
Blockchain Sync	The sync value to include in the payload for API calls.	`true`
Blockchain Timeout	The timeout value to include in the payload for API calls.	`6000`
Policy Name	The name of the policy that enables controlled management and access to APIs through defined permissions for groups and compartments in the organization	`ChaincodeName`
VCN Display Name	The display name of the OCI virtual cloud network.	`ChaincodeName`

Copyright © 2025, Oracle and/or its affiliates.

Field	Description	Default value
Application Display Name	The display name of the OCI application.	`ChaincodeName`
Gateway Display Name	The display name of API Gateway.	`ChaincodeName`
Deployment Display Name	The display name of API Gateway.	`ChaincodeName`
OCI Repo Name	The OCI Registry repository name, which must be all lowercase letters.	`ChaincodeName` in lowercase characters

4. Click **Save**.
5. Click the **Mapping** tab. The Mapping pane shows a table of all of the controller APIs that are available in the chaincode. Each method has a corresponding **API Endpoint**, which you can edit as needed. Select the corresponding **Generate Wrapper APIs** checkbox to include the controller API in the wrapper API package. Click **Select All** to include all controller APIs.
6. Click **Save**.
7. Click **Generate**. The wrapper API package and associated Postman collection are generated.

Wrapper API Package Components

Wrapper API packages contain an archive file of the wrapper APIs, a Terraform script for deployment, and a corresponding Postman collection.

- The wrapper API archive file is named `<ChaincodeName>WrapperAPI.zip`. It also contains a Terraform script that must be deployed to the stack resource.
- The Postman collection file is named `<ChaincodeName>_WrapperAPI.postman_collection.json`. You can use this collection to call all of the wrapper APIs.

Wrapper API Package

The wrapper API package contains a Terraform script that provisions all of the Oracle Cloud Infrastructure (OCI) resources that are necessary for creating the wrapper APIs. There are additional Terraform support files and an Oracle Functions folder for each API. The wrapper API packages creates the following OCI resources.

- Virtual Cloud Network (VCN): Establishes the network infrastructure for communication.
- Applications (Oracle Functions): Deploys serverless functions to handle API logic.
- API Gateway: Creates the gateway to manage and route API requests.
- API Deployment: Configures and deploys the APIs on the API Gateway.
- API Deployment Policy: Sets up the necessary IAM policies to enable secure access.
- OCI Registry: Provides a container registry for managing Docker images.

After you generate wrapper APIs, if you want to change any configuration variables, you can update them in Visual Studio Code, or you can extract the wrapper API package and update the `terraform.tfvars` file with updated endpoints and resource names and the `routes.go` file with updated endpoints. Edit the `function_paths` variable in the `terraform.tfvars` file to update an endpoint. The function_paths variable defines

Copyright © 2025, Oracle and/or its affiliates.

endpoints with the following syntax: `{endpoint, methodType}`. The following text shows an example of a `function_paths` variable.

```
function_paths="[{\"endpoint\":\"/activateCBDCAccount\",\"methodType\":
[\"POST\"]},{\"endpoint\":\"/addCBAdmin\",\"methodType\":[\"POST\"]},
{\"endpoint\":\"/approveBurn\",\"methodType\":[\"POST\"]}]"
```

The `routes.go` file is in the <*ChaincodeName*>`OCIFunction` folder. The `routeData` variable contains metadata for all routes created in the gateway, including the route name, arguments, and whether there are optional arguments. When you update an endpoint in the `function_paths` variable in the `terraform.tfvars` file, you must also updat the corresponding entry in the `routeData` variable in the `routes.go` file, as shown in the following example.

```
var routeData = map[string]Route{
    "/activateCBDCAccount": {
        Args:           []string{"activateAccount", "orgId", "userId",
"tokenId"},
        OptionalParams: true,
    },
}
```

Postman Collection

Postman collections now support confidential chaincode and OAuth 2.0 authentication. For more information, see Enhancements to Postman Collections.

The Postman collection includes updated endpoints and payloads for all APIs. The following code shows an example payload.

```
{
    "orgId": "{{bc-org-id}}",
    "userId": "user1",
    "tokenType": "fungible",
    "applicationGroups": "[\"application_groups value\"]",
    "dailyLimits":
"{\"max_daily_amount\":10000,\"max_daily_transactions\":100}",
    "endorsers": {{endorsers}}
}
```

The following table shows the Postman collection variables.

Variable	Description	Default value
`bc-admin-user`	The administrative user, who has the `admin` role and can access all POST requests. By default, this user is the caller of all POST requests in the chaincode.	`bc-admin-user value`
`bc-admin-user-password`	Administrative user password.	`bc-admin-user-password value`
`bc-org-id`	The default organization ID in all POST requests where `orgId` is the parameter name	`bc-org-id value`

Copyright © 2025, Oracle and/or its affiliates.

Variable	Description	Default value
bc-user-id	The default user ID in all POST requests where userId is the parameter name	bc-user-id value
bc-token-id	The default token ID in all POST requests where tokenId is the parameter name	bc-token-id value
endorsers	The endorsers array lists the specific peers (for example: peer1, peer2) to endorse this transaction.	["org1-xyz-abc.blockchain.ocp.oraclecloud.com:20009", "org2-xyz-abc.blockchain.ocp.oraclecloud.com:20009"]
api-gateway-endpoint	The endpoint of each request, which serves as a base path for wrapper API endpoints. If there are fewer than 50 APIs, a single endpoint is used. If there are more than 50 APIs, the endpoints are dynamically generated as api-gateway-endpoint1, api-gateway-endpoint2, and so on, based on the number of APIs.	https://xyz.apigateway.region.oci.customer-oci.com/CBDC
bc-url	The REST proxy URL of the Oracle Blockchain Platform instance.	https://test-xyz-abc.blockchain.ocp.oraclecloud.com:7443/restproxy
access-token-url	The Oracle Identity Cloud Service (IDCS) access token URL. For example: <idcs_endpoint>/oauth/v1/token	access-token-url value
client-id	The client ID of the Oracle Blockchain Platform instance.	client-id
client-secret	The client secret of the Oracle Blockchain Platform instance.	client-secret
peer	This variable exists only for the confidential chaincode wrapper API Postman collection, which requires the peer header for all setter and getter APIs.	org-xyz-abc.blockchain.ocp.oraclecloud.com:20009

After you deploy the wrapper API package, the output from the stack resource deployment is a JSON object that contains the gateway endpoint values. If you generate more than 50 APIs, multiple gateway endpoints are generated, one for every 50 APIs. You must update the Postman collection variables related to these endpoints. The endpoint-related variables in the Postman collection must be updated with the appropriate values from the output of the wrapper API package deployment in the Stack Resource Manager.

All setter APIs in the wrapper API Postman collection include either the endorsers or sameOrgEndorser parameter in the request payload. The information that specifies which APIs require the sameOrgEndorser parameter is defined in the sameOrgEndorserOptionInWrapperAPI parameter in the .ochain.json file in the chaincode. APIs listed in this parameter will have sameOrgEndorser set to true in their payloads. All other

Copyright © 2025, Oracle and/or its affiliates.

setter APIs will include the `endorsers` parameter instead. The following example shows the parameter for the wholesale CBDC chaincode.

```
"sameOrgEndorserOptionInWrapperAPI":
["addConversionRate","addTokenAdmin","addTokenAuditor","approveBurn","approveM
int","burnTokens","createExchangePoolAccounts","deleteHistoricalTransactions",
"initializeCBDCToken","initializeExchangePoolUser","mintWithFundingExchangePoo
l","rejectBurn","rejectMint","removeTokenAdmin","removeTokenAuditor","requestB
urn","requestMint","updateCBDCToken","updateConversionRate"]
```

You can customize the `sameOrgEndorserOptionInWrapperAPI` parameter in the `.ochain.json` file as needed. When you generate wrapper APIs, the specified APIs will then include the `sameOrgEndorser` parameter as `true` in their payloads.

For more details about using Postman collections, see the following topics.

- Generate a Postman Collection Using the CLI
- Generate a Postman Collection Using Visual Studio Code
- Enhancements to Postman Collections

Deploy Wrapper APIs

Oracle Blockchain Platform Digital Assets Edition can generate wrapper APIs for all controller APIs in chaincode generated by Blockchain App Builder. Complete the following steps to deploy the wrapper API package in the stack resource manager and to test the wrapper APIs by using the associated Postman collection.

When you generate a wrapper API package using Blockchain App Builder, it creates two files with default naming conventions in the chosen directory:

- `<ChaincodeName>WrapperAPI.zip`: An archive file that contains the Terraform script required to deploy the wrapper APIs in Oracle Cloud Infrastructure (OCI) Resource Manager.
- `<ChaincodeName>_WrapperAPI.postman_collection.json`: A Postman collection used to test the wrapper APIs after deployment.

To deploy the wrapper APIs, you use the `WrapperAPI.zip` file to create a stack in OCI Resource Manager.

The following prerequisites must be met to deploy the wrapper API package.

- The wrapper API package must be generated using Blockchain App Builder.
- The package must include all required configuration variable values and endpoint values for the wrapper APIs.

1. Log in to OCI.

   Open the OCI console and sign in: https://cloud.oracle.com/. Ensure that you're in the compartment where the stack will be deployed.

2. Expand the **Navigation** menu. Select **Developer Services**. Under **Resource Manager**, select **Stacks**.

   The Resource Manager interface for stacks opens. You can view existing stacks, create stacks, and manage your infrastructure as code using Terraform configurations.

3. Click **Create Stack**.

4. Configure the stack information.

Copyright © 2025, Oracle and/or its affiliates.

a. In the **Terraform Configuration** section, select **My Configuration**.

b. In **Stack Configuration**, **Terraform configuration source**, select **.Zip file**.

c. Click **Browse** and navigate to your wrapper API `.zip` file.

d. After the `.zip` file is uploaded, ensure that the stack name is correct.

e. In the **Terraform version** drop-down list, select v1.2 or higher. By default, Resource Manager uses the latest version. If you select a version earlier than v1.2, the deployment will fail.

f. All remaining settings can be kept as default. Click **Next**.

5. Configure the stack variables.

   a. Verify that all configuration variable values are accurate and align with the configuration variables specified in the wrapper API package to meet the requirements for your deployment.

   b. For **ocir_user_name**, enter the email address associated with your OCI account.

   c. For **ocir_user_password** enter your password, which is your OCI auth token. You can generate this token in the profile or user settings section of the OCI console. The auth token functions as a password for logging into the Oracle Cloud Infrastructure Registry.

      For more information about generating an auth token, see Generating an Auth Token.

   d. All remaining settings can be kept as default. Click **Next**.

6. Review the stack information. If everything is correct, click **Create**.

7. Run the Terraform plan.

   a. On the stack's **Details** page, click **Plan** to start creating an execution plan.

   b. Optionally, configure the job plan. You can edit the default name or select advanced options to change the provider version or adjust settings.

   c. Click **Plan** to create and run the plan job.

      Your Terraform configuration is parsed and an execution plan is generated and run, which completes the following tasks:

      - Validates the Terraform script to ensure there are no syntax or configuration errors.
      - Simulates the resource creation process without making any changes to the actual infrastructure.
      - Provides an output summary that lists the resources the script will create, modify, or destroy.

   d. Monitor the status in the **Jobs** section of your stack. After the plan completes successfully, review the output to confirm the number of resources that will be created and to verify that there are no issues.

8. Apply the Terraform plan to the stack.

   a. On the stack's **Details** page, click **Apply**.

   b. In the Terraform Apply panel, click **Apply**.

      Applying the Terraform plan completes the following tasks:

      - Provisions all of the following resources as defined in the Terraform script:
        – Virtual cloud network (VCN)
        – Applications (OCI Functions)

Copyright © 2025, Oracle and/or its affiliates.

- API Gateway
- API deployment
- IAM policy
- OCI Registry repository

- Creates all of the required infrastructure for the wrapper APIs.

c. Monitor the status in the **Jobs** section of your stack to ensure the job completes successfully with no errors.

9. Retrieve the API Gateway endpoints.

a. After the apply job completes, go to the **Outputs** section of the Apply job page to view the generated output values.

b. The output returns a JSON object where the keys represent endpoint names and the values are their corresponding endpoint values:

```
api_gateway_endpoints{"api_gateway_endpoint":"https://
grqkdiwsugp3fp2m5z3zgpo4.apigateway.ap-mumbai-1.oci.customer-oci.com/
WholesaleCBDC"}
```

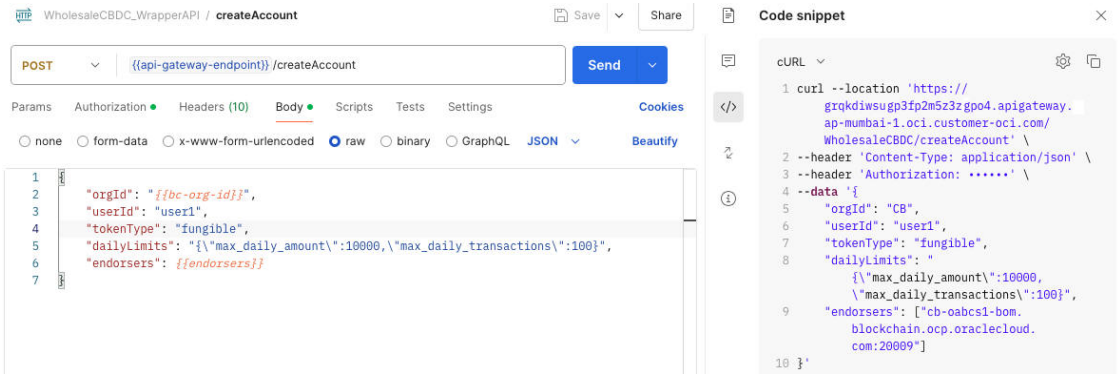
c. Copy the endpoint. You need the endpoint to update the API Postman collection.

Testing Wrapper APIs with Postman

You can test the wrapper APIs by using Postman. Configure all of the variables in the wrapper API Postman collection by assigning the appropriate values. Get the `api_gateway_endpoint` value from the output of the stack deployment in OCI, where the dynamically generated gateway endpoints are provided.

After the variables in the Postman collection are updated, you can test the wrapper APIs. Each API request in the collection includes the necessary payload specific to the corresponding wrapper API.

Figure 4-1 Testing an API Wrapper Collection with Postman



The following information can be used to troubleshoot problems when deploying and using wrapper APIs.

Unauthorized: Invalid ocir_user_name or ocir_user_password

This error occurs when either the user name or password credentials are incorrect. To resolve this issue, follow these steps:

Copyright © 2025, Oracle and/or its affiliates.

1. Navigate to the **Variables** section in the stack Details page.
2. Click **Edit Variables** to modify the user name and password values.
3. Update the credentials with the correct information.
4. Click **Apply** to redeploy the stack.

Deployment Failure - "denied: Anonymous users are only allowed read access on public repos"
Occasionally, the deployment might fail with the following error:

```
denied: Anonymous users are only allowed read access on public repos
```

This is an intermittent issue that is related to Docker operations in the stack's back end. Specifically, this error occurs when attempting to push Docker images to the Oracle Cloud Infrastructure Registry without proper authentication.
While the wrapper API package includes a script that runs the `docker login` command to ensure proper authentication before pushing the images, this error can still occur sporadically. It typically happens when the Docker authentication step is not properly recognized by the system at the time of image push.

If this error occurs, follow these steps to resolve the issue:

1. Click **Destroy** in the OCI Stack Resource Manager to destroy all of the resources that were created during the deployment process.
2. After the resources are destroyed, click **Apply** again to redeploy the stack. This triggers the creation of the necessary resources including proper Docker authentication.

Generic Deployment Failure: Intermittent Issues
Stack deployment can fail sometimes due to random Docker-related issues. When this happens, follow these steps to resolve the issue:

1. Click **Destroy** to remove all of the resources that were created during the failed deployment.
2. After the resources are destroyed, navigate back to the stack and click **Apply** to redeploy the stack.

This can resolve intermittent issues and allow for a successful deployment.

Chaincode Events

The enhanced version of Blockchain App Builder can generate chaincode events for token operations.

Chaincode events are specific notifications that are emitted when transactions run. Events include transaction information that can be used to notify external systems about specific conditions or changes in the blockchain ledger state. You can use chaincode events to enable real-time integration and interaction with applications that are not on the blockchain, and to facilitate event-driven workflows and monitoring across the blockchain environment. Chaincode events have two components, the event name and the payload.

Chaincode events are supported for all Blockchain App Builder specification files. If you enable chaincode events, all controller functions in your scaffolded project will emit events, except for getter methods. For example, in a token scenario, chaincode events will be emitted when tokens are minted, transferred, burned, or locked

Copyright © 2025, Oracle and/or its affiliates.

You use the Boolean `events` parameter in the specification file to enable chaincode events, as shown in the following example.

```
assets:
    - name: FiatMoneyTOK # Asset name
      type: token  # Asset type
      events: true  # Generate events for create, update and delete APIs
```

If you enable events, the controller functions in your scaffolded chaincode project will include event creation methods, as shown in the following examples.

TypeScript:

```
@Validator(yup.string(), yup.string(), yup.string())
public async createAccount(org_id: string, user_id: string, token_type:
string) {
  await this.Ctx.Auth.checkAuthorization("ACCOUNT.createAccount", "TOKEN",
{ org_id });
  await this.Ctx.Model.createEvent(EVENT_NAME.CREATE_ACCOUNT, { org_id,
user_id, token_type });
  return await this.Ctx.Account.createAccount(org_id, user_id, token_type);
}
```

Go:

```
func (t *Controller) CreateAccount(org_id string, user_id string, token_type
string, daily_limits ...account.AccountDailyLimits) (interface{}, error) {
    auth, err := t.Ctx.Auth.CheckAuthorization("Account.CreateAccount",
"TOKEN", map[string]string{"org_id": org_id})
    if err != nil && !auth {
        return nil, fmt.Errorf("error in authorizing the caller  %s",
err.Error())
    }
    err = t.Ctx.Model.CreateEvent(constants.CreateAccountEventName,
map[string]interface{}{"org_id": org_id, "user_id": user_id, "token_type":
token_type})
    if err != nil {
        return nil, err
    }
    return t.Ctx.Account.CreateAccount(org_id, user_id, token_type,
daily_limits...)
}
```

Chaincode events use the following default values for the name and payload components. You can modify the default values as needed.

EventName
The name of the controller function.

Payload
A JSON object that contains all of the input parameters of the controller function.

The extended Token Taxonomy Framework and ERC-1155 standards include an `events` parameter in the token details. If the `events` parameter in the specification file is set to true, the `events` parameter in the generated token is set to true. If the `events` parameter in the

Copyright © 2025, Oracle and/or its affiliates.

specification file is set to false or not defined, the `events` parameter in the generated token is set to false. The following examples show a token with the new `events` parameter for both TypeScript and Go.

TypeScript:

```
{
    "metadata": {
        "paintingName": "monalisa",
        "description": "monalisa painting",
        "image": "image link",
        "painterName": "Leonardo da Vinci"
    },
    "assetType": "otoken",
    "events": true,
    "quantity": 1,
    "tokenId": "artnft",
    "tokenName": "artcollection",
    "tokenDesc": "artcollection nft",
    "tokenStandard": "erc1155+",
    "tokenType": "nonfungible",
    "tokenUnit": "whole",
    "behaviors": [
        "indivisible",
        "singleton",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter",
        "burner_role_name": "burner"
    },
    "mintable": {
        "max_mint_quantity": 500
    },
    "owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "createdBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
    "creationDate": "2022-12-29T04:08:35.000Z",
    "isBurned": false,
    "tokenUri": "tu",
    "price": 10000,
    "onSaleFlag": false
}
```

Go:

```
{
    "AssetType": "otoken",
    "Behavior": [
        "indivisible",
        "singleton",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "mintable",
            "transferable",
            "burnable",
            "roles"
        ],
        "CreatedBy":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "CreationDate": "2022-12-29T09:57:03+05:30",
        "Events": true,
        "IsBurned": false,
        "Mintable": {
            "Max_mint_quantity": 500
        },
        "OnSaleFlag": false,
        "Owner":
"oaccount~42e89f4c72dfde9502814876423c6da630d466e87436dd1aae201d347ad1288d",
        "Price": 100,
        "Quantity": 1,
        "Roles": {
            "burner_role_name": "burner",
            "minter_role_name": "minter"
        },
        "TokenDesc": "token description",
        "TokenId": "monalisa",
        "TokenMetadata": {
            "Description": "Mona Lisa Painting",
            "Image": "monalisa.jpeg",
            "PainterName": "Leonardo_da_Vinci",
            "PaintingName": "Mona_Lisa"
        },
        "TokenName": "artcollection",
        "TokenStandard": "erc1155+",
        "TokenType": "nonfungible",
        "TokenUnit": "whole",
        "TokenUri": "https://
bafybeid6pmpp62bongoip5iy2skosvyxh3gr7r2e35x3ctvawjco6ddmsq\\\\ .ipfs.infura-
ipfs.io/?filename=MonaLisa.jpeg"
    }
```

Generating Events

The `stub.setEvent` method enables the chaincode to create and emit an event when a transaction runs. The following code shows the TypeScript version of the method.

```
async setEvent(eventName: string, payload: Buffer): Promise<void>
```

In this example, `eventName` is the name to assign to the event, and `payload` is data associated with the event. The payload can contain any information that you want to send with the event, typically serialized in JSON format.

Chaincode Events for Batch Methods

The enhanced ERC-1155 standard supports batch methods. Batch methods operate on multiple tokens that are passed as parameters. For batch methods, chaincode events are emitted for only the tokens where the `events` parameter is set to `true` in the specification file.

Copyright © 2025, Oracle and/or its affiliates.

For each transaction completed in the batch method, a corresponding chaincode event is generated. The payload of each chaincode event contains the transaction details. For example, if you use the `BatchTransfer` method to transfer quantities of five different tokens, five corresponding chaincode events are emitted. The payload of each event contains the token details and the quantity transferred, along with common parameters that are applicable to all batch transfers.

The following example shows events code using the enhanced ERC-1155 standard.

```
@Validator(yup.string(), yup.string(), yup.string(), yup.string(),
yup.array().of(yup.string()), yup.array().of(yup.number()))
  public async batchTransferFrom(
    fromOrgId: string,
    fromUserId: string,
    toOrgId: string,
    toUserId: string,
    tokenIds: string[],
    quantity: number[]
  ) {
    const fromAccountId =
this.Ctx.ERC1155Account.generateAccountId(fromOrgId, fromUserId,
ACCOUNT_TYPE.USER_ACCOUNT);
    const toAccountId = this.Ctx.ERC1155Account.generateAccountId(toOrgId,
toUserId, ACCOUNT_TYPE.USER_ACCOUNT);
    let tokenAssets = [];
    for (let i = 0; i < tokenIds.length; i++) {
      const tokenAsset = await this.Ctx.ERC1155Token.get(tokenIds[i]);
      tokenAssets.push(tokenAsset);
    }
    await this.Ctx.Model.createEventForBatch(EVENT_NAME.BATCH_TRANSFER_FROM,
{ fromOrgId, fromUserId, toOrgId, toUserId }, quantity, tokenAssets);
    return await this.Ctx.ERC1155Token.batchTransferFrom(fromAccountId,
toAccountId, tokenIds, quantity);
  }
```

Chaincode Events for Multiple Assets

The enhanced Token Taxonomy Framework and ERC-1155 standards support defining more than one token asset in a specification file. The chaincode event behavior is different depending on whether a method is token-specific (such as creating or updating a token) or common (such as minting or burning).

For token-specific methods, chaincode events are generated for only the tokens where the `events` parameter is set to `true` in the specification file.

For common methods, chaincode events are generated in the scaffolded project if any token has the `events` parameter set to `true` in the specification file. The actual chaincode event behavior is based on the number of token ID parameters passed to the method.

- If a single token ID is passed as a parameter, chaincode events are generated only if the `events` parameter in the corresponding token details is set to true.
- If multiple token IDs are passed as parameters, chaincode events are generated only if the `events` parameter in any one of the token details is set to true.
- If no token ID is passed as a parameter, chaincode events are always generated.

Copyright © 2025, Oracle and/or its affiliates.

The following list shows common methods that must be passed two tokens. This list applies to the extended Token Taxonomy Framework standard.

- `addConversionRate(from_token_id: string, to_token_id: string, token_conversion_rate: number)`
- `updateConversionRate(from_token_id: string, to_token_id: string, token_conversion_rate: number)`
- `tokenConversion(from_token_id: string, to_token_id: string, to_org_id: string, to_user_id: string,token_quantity: number)`
- `exchangeToken(fromTokenId: string, fromOrgId: string, fromUserId: string, fromTokenQuantity: number, toTokenId: string, toOrgId: string,toUserId: string,toTokenQuantity: number)`

The following list shows common methods that do not take tokens as arguments The following list applies to the extended Token Taxonomy Framework standard.

- `addTokenAdmin(org_id: string, user_id: string)`
- `removeTokenAdmin(org_id: string, user_id: string)`
- `addOrgAdmin(org_id: string, user_id: string)`
- `removeOrgAdmin(org_id: string, user_id: string)`
- `createAccount(org_id: string, user_id: string, token_type: string)`
- `deleteHistoricalTransactions(time_to_expiration: Date)`
- `initializeExchangePoolUser(org_id: string, user_id: string)`

This list shows common methods that do not take tokens as arguments for the extended ERC-1155 standard.

- `addTokenAdmin(orgId: string, userId: string)`
- `removeTokenAdmin(orgId: string, userId: string)`
- `createAccount(orgId: string, userId: string, ftAccount: boolean, nftAccount: boolean)`
- `createUserAccount(orgId: string, userId: string)`
- `createTokenAccount(orgId: string, userId: string, tokenType: TokenType)`
- `addTokenSysRole(orgId: string, userId: string, role: string)`
- `removeTokenSysRole(orgId: string, userId: string, role: string)`
- `transferTokenSysRole(fromOrgId: string, fromUserId: string, toOrgId: string, toUserId: string, role: string)`
- `deleteHistoricalTransactions(time_to_expiration: Date)`

TypeScript SDK Methods for Chaincode Events

`createEvent`
This method generates events based on a specified event name and payload.

```
public async createEvent(eventName: any, eventPayload: any, assets?: any)
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `eventName: string` – The event name to use when generating events.
- `eventPayload: map[string]interface{}` – The event payload to use when generating events.
- `assets` – Optionally, the token asset can be passed as a parameter to the method.

`createEventForBatch`
This method generates events for batch operations such as the `mintBatch` or `burnBatch` methods.

```
public async createEventForBatch(eventName: any, eventPayload: any,
quantities: number[], assets: any)
```

Parameters:

- `eventName: string` – The event name to use when generating events.
- `eventPayload: map[string]interface{}` – The event payload to use when generating events.
- `quantities: number[]` – A list of amounts, corresponding to each token ID, that represent the number of tokens to use in the batch method transactions.
- `assets` – Optionally, the token asset can be passed as a parameter to the method.

Go SDK Methods for Chaincode Events

`CreateEvent`
This method generates events based on a specified event name and payload.

```
func (m *Model) CreateEvent(eventName string, eventPayload
map[string]interface{}, assets ...interface{})
```

Parameters:

- `eventName: string` – The event name to use when generating events.
- `eventPayload: map[string]interface{}` – The event payload to use when generating events.
- `assets` – Optionally, the token asset can be passed as a parameter to the method.

`CreateEventForBatch`
This method generates events for batch operations such as the `mintBatch` or `burnBatch` methods.

```
func (m *Model) CreateEventForBatch(eventName string, eventPayload
map[string]interface{}, quantities []float64, assets []interface{})
```

Parameters:

- `eventName: string` – The event name to use when generating events.
- `eventPayload: map[string]interface{}` – The event payload to use when generating events.

Copyright © 2025, Oracle and/or its affiliates.

- `quantities: []float64` – A list of amounts, corresponding to each token ID, that represent the number of tokens to use in the batch method transactions.
- `assets` – Optionally, the token asset can be passed as a parameter to the method.

Token Taxonomy Framework Enhancements

The enhanced version of Blockchain App Builder includes new functionality related to the extended Token Taxonomy Framework standard.

For more information on all of the following enhancements, see the method descriptions in Scaffolded TypeScript Project for Token Taxonomy Framework and Scaffolded Go Project for Token Taxonomy Framework.

Daily Transaction Limits

You can restrict the number of transactions an account can complete daily, as well as the number of tokens that can be acted on. The `max_daily_amount` and `max_daily_transactions` input parameters to the `createAccount` method control this behavior. These parameters are optional.

You can achieve higher throughput if you do not set the daily transaction limits for an account.

Approval Requirements for Minting and Burning

You can set up approvals for minting and burning tokens, so that users with the minter or burner role must submit a request to an approver, instead of minting or burning tokens directly. Approvers can accept or reject requests to mint or burn tokens. To enable approvals for minting and burning, you use the `mint_approval_required` and `burn_approval_required` parameters in the specification file. You must then also specify values for `mint_approver_role_name` and `burn_approval_role_name`, as shown in the following example.

```
behavior: # Token behaviors
          - divisible:
                decimal: 2
          - mintable:
                max_mint_quantity: 1000
                mint_approval_required: true
          - transferable
          - burnable
                burn_approval_required: true
          - holdable
          - roles:
                minter_role_name: minter
                notary_role_name: notary
                mint_approver_role_name: minter_notary
                burn_approver_role_name: burner_notary
```

When a burn request is made, the specified amount is immediately deducted from the available balance and added to the `onhold_burn_balance` field in the user's account object. If the request is approved, the tokens are burned. If the request is rejected, the tokens are returned from the `onhold_burn_balance` field to the available balance. You can use the `getAccountOnHoldBurnBalance` method to retrieve the account balance that is on hold for a particular user because of a burn request.

Copyright © 2025, Oracle and/or its affiliates.

Fetching Transaction History from the Rich History Database

You can synchronize data to the rich history database and then fetch the data using chaincode API calls to retrieve transaction history from the rich history database. Before you can use these methods, you must run Oracle Autonomous Database with Oracle REST Data Services (ORDS) and OAuth enabled, as described in Oracle Database View Definitions for Wholesale CBDC.

Category and Description Attributes in Transaction Objects

- Category and description attributes must be included in the `transferTokens`, `holdTokens`, `issueTokens`, `requestMint`, `requestBurn`, `burnTokens` and `rejectBurn` methods in the controller file. The corresponding SDK methods must also include category and description attributes.
- The category and description attribute input is in the form of a JSON object named `info_details`, as shown in the following example.

```
{
     "category" : "category input",
     "description" : "description input"
}
```

- The `info_details` field is optional. You can pass only a category or only a description as needed.
- The GET methods related to any transactions for `transferTokens`, `holdTokens`, `executeHold`, `releaseHold`, `requestMint`, `approveMint`, `rejectMint`, `requestBurn`, `approveBurn` and `rejectBurn` must include category and description attributes in the payload response if they are present.
- The category field is limited 20 characters and the description field is limited to 250 characters.

Confidential Payments Model

The enhanced version of Blockchain App Builder includes a model attribute that generates additional methods for confidential mode.

If you include the `confidential: true` parameter in the specification file for tokens that use the extended Token Taxonomy Framework standard, Blockchain App Builder generates chaincode for use in confidential mode, including the following modified versions of the standard automatically generated and SDK methods (TypeScript only).

For more information about confidential payments and the confidential wholesale central bank digital currency (CBDC) scenario, see Wholesale Central Bank Digital Currency with Confidential Payment.

Modified TypeScript Methods for Confidential Chaincode

If you include the `confidential: true` parameter in the specification file for tokens that use the extended Token Taxonomy Framework standard, Blockchain App Builder generates modified versions of the standard automatically generated methods as well as additional methods.

Copyright © 2025, Oracle and/or its affiliates.

isTokenAdmin
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`, otherwise it returns `false`. A `Token Admin`, `Token Auditor`, any `Org Admin`, or any `Org Auditor` can call this function on any other user in the blockchain network. Other users can call this method only on their own accounts.

```
@GetMethod()
 @Validator(yup.string(), yup.string())
 public async isTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.isUserTokenAdmin", "TOKEN",
{ org_id });
    return await this.Ctx.Auth.isUserTokenAdmin(org_id, user_id);
 }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- The method returns `true` if the caller is a `Token Admin`, otherwise it returns `false`.

addTokenAdmin
This method adds a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
@Validator(yup.string(), yup.string())
  public async addTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.addTokenAdmin", "TOKEN");
    await this.Ctx.Model.createEvent(EVENT_NAME.ADD_TOKEN_ADMIN, { org_id,
user_id });
    return await this.Ctx.Admin.addTokenAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Admin` of the chaincode.

Return Value Example:

```
{
  "msg": "Successfully added Token Admin (Org_Id: CentralBank, User_Id: cb1)"
}
```


Copyright © 2025, Oracle and/or its affiliates.

`removeTokenAdmin`
This method removes a user as a `Token Admin` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
  public async removeTokenAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.removeTokenAdmin", "TOKEN");
    await this.Ctx.Model.createEvent(EVENT_NAME.REMOVE_TOKEN_ADMIN, { org_id,
user_id });
    return await this.Ctx.Admin.removeTokenAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as a `Token Admin` of the chaincode.

Return Value Example:

```
{"msg": "Successfully removed Admin (Org_Id: Org1MSP, User_Id: User1)"}
```

`addOrgAdmin`
This method adds a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
@Validator(yup.string(), yup.string())
  public async addOrgAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.addOrgAdmin", "TOKEN",
{ org_id });
    await this.Ctx.Model.createEvent(EVENT_NAME.ADD_ORG_ADMIN, { org_id,
user_id });
    return await this.Ctx.Admin.addOrgAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as an `Org Admin` of the organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`removeOrgAdmin`
This method removes a user as an `Org Admin` of the organization. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string())
  public async removeOrgAdmin(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.removeOrgAdmin", "TOKEN",
{ org_id });
    await this.Ctx.Model.createEvent(EVENT_NAME.REMOVE_ORG_ADMIN, { org_id,
user_id });
    return await this.Ctx.Admin.removeOrgAdmin(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was removed as an `Org Admin` of the organization.

Return Value Example:

```
{
  "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

`addTokenAuditor`
This method adds a user as a `Token Auditor` of the chaincode. This method can be called only by a `Token Admin` of the chaincode. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
@Validator(yup.string(), yup.string())
  public async addTokenAuditor(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.addTokenAuditor", "TOKEN");
    await this.Ctx.Model.createEvent(EVENT_NAME.ADD_TOKEN_AUDITOR, { org_id,
user_id });
    return await this.Ctx.Admin.addTokenAuditor(org_id, user_id);
  }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
  "msg": "Successfully added Token Auditor (Org_Id: CentralBank, User_Id:
cb_admin_demo)"
}
```

`removeTokenAuditor`
This method removes a user as a `Token Auditor` of the chaincode. This method can be called only by a `Token Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
  public async removeTokenAuditor(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.removeTokenAuditor",
"TOKEN");
    await this.Ctx.Model.createEvent(EVENT_NAME.REMOVE_TOKEN_AUDITOR,
{ org_id, user_id });
    return await this.Ctx.Admin.removeTokenAuditor(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
          "msg": "Successfully removed Token Auditor (Org_Id: CB, User_Id:
cb)"
       }
```

`addOrgAuditor`
This method adds a user as a `Org Auditor` of the chaincode. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
@Validator(yup.string(), yup.string())
  public async addOrgAuditor(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.addOrgAuditor", "TOKEN",
{ org_id });
    await this.Ctx.Model.createEvent(EVENT_NAME.ADD_ORG_AUDITOR, { org_id,
user_id });
    return await this.Ctx.Admin.addOrgAuditor(org_id, user_id);
  }
```

Parameters:

Copyright © 2025, Oracle and/or its affiliates.

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Org Auditor` of the chaincode.

Return Value Example:

```
{
           "msg": "Successfully added Org Auditor (Org_Id: CentralBank,
User_Id: cb_admin_demo)"
       }
```

`removeOrgAuditor`
This method removes a user as a `Org Auditor` of the chaincode. This method can be called only by a `Token Admin` or `Org Admin` of the chaincode.

```
@Validator(yup.string(), yup.string())
  public async removeOrgAuditor(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ADMIN.removeOrgAuditor", "TOKEN",
{ org_id });
    await this.Ctx.Model.createEvent(EVENT_NAME.REMOVE_ORG_AUDITOR, { org_id,
user_id });
    return await this.Ctx.Admin.removeOrgAuditor(org_id, user_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
           "msg": "Successfully removed Org Auditor (Org_Id: CB, User_Id: cb)"
       }
```

`createAccount`
This method creates an account for a specified user and token. An account must be created for any user who will have tokens at any point. Accounts track balances, on-hold balances, and transaction history. An account ID is formed by concatenating the asset type and token ID and then creating a SHA-256 hash over a concatenation of the organization ID and user ID. This method can be called only by a `Token Admin` or by an `Org Admin` of the specified organization.

```
@Validator(yup.string(),yup.string(), yup.string(), yup.object().nullable())
  public async createAccount(org_id: string, user_id: string, token_type:
string, dailyLimits?: DailyLimits) {
```


Copyright © 2025, Oracle and/or its affiliates.

```
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.createAccount", "TOKEN",
{ org_id });
    await this.Ctx.Model.createEvent(EVENT_NAME.CREATE_ACCOUNT, { org_id,
user_id, token_type, dailyLimits });
    return await this.Ctx.Account.createAccount(org_id, user_id, token_type,
dailyLimits);
  }
```

Parameters:

- `org_id` – The membership service provider (MSP) ID of the user to create the account for. The ID must begin with an alphanumeric character and can include letters, numbers, and special characters such as underscores (_), periods (.), at signs (@), and hyphens (-).
- `user_id` – The user name or email ID of the user. The ID must begin with an alphanumeric character and can include letters, numbers, and special characters such as underscores (_), periods (.), at signs (@), and hyphens (-).
- `token_type: string` – The type of token, which must be `fungible`.
- `daily_limits: DailyLimits` – A JSON object of the following type.

  ```
  {
       "max_daily_amount": 100000
       "max_daily_transactions": 10000
   }
  ```

  In the example, the `max_daily_amount` value is the maximum amount of tokens that can be transacted daily and `max_daily_transactions` value is the maximum number of transactions that can be completed daily.

Return Value Example:

```
{
    "bapAccountVersion": 0,
    "assetType": "oaccount",
    "account_id":
"oaccount~b53cb2c19c92d1d5c8cb9f6e988e7761c34e03e014e6c4b889565fc0abf46c8a",
    "org_id": "CentralBank",
    "token_type": "fungible",
    "token_id": "",
    "token_name": "",
    "balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_burn_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
}
```


Copyright © 2025, Oracle and/or its affiliates.

`associateTokenToAccount`
This method associates a fungible token with an account. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the relevant organization.

```
@Validator(yup.string(), yup.string(), yup.string().optional())
  public async associateTokenToAccount(account_id: string, token_id: string,
custom_account_id?: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.associateToken", "TOKEN",
{ account_id });
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.ASSOCIATE_TOKEN_TO_ACCOUNT,
{ account_id, token_asset }, token_asset);
    return await this.Ctx.Account.associateToken(account_id, token_id,
custom_account_id);
  }
```

Parameters:

- `account_id: string` – The ID of the account.
- `token_id: string` – The ID of the token.
- `custom_account_id: string` – For confidential mode, the bank account ID of the account, a unique random alphanumeric identifier. By default, 14 characters long.

Returns:

- On success, a JSON object of the updated account. The `bapAccountVersion` parameter is defined in the account object for internal use.

Return Value Example:

```
{
    "bapAccountVersion": 0,
    "assetType": "oaccount",
    "account_id":
"oaccount~b53cb2c19c92d1d5c8cb9f6e988e7761c34e03e014e6c4b889565fc0abf46c8a",
    "org_id": "CentralBank",
    "token_type": "fungible",
    "token_id": "USD",
    "token_name": "cbdc",
    "balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_burn_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "application_groups": [
        "CENTRAL_BANK_USERS"
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getAllAccounts`
This method returns a list of all accounts. This method can be called only by a `Token Admin` or `Token Auditor`. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
@GetMethod()
  @Validator()
  public async getAllAccounts() {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAllAccounts", "TOKEN");
    return await this.Ctx.Account.getAllAccounts();
  }
```

Parameters:

- none

Return Value Example:

```
[
    {
        "key":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
        "valueJson": {
            "bapAccountVersion": 3,
            "assetType": "oaccount",
            "account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance":
"028a27c752e9b518d156e61da6589e723b179b6d7351dc34ec92bbb27b783ed1dc",
            "onhold_balance":
"02d53a7eda9484bda7252212714f647b70d32663cfa416f93f488ef6b6bc8fb2eb",
            "onhold_burn_balance":
"02d53a7eda9484bda7252212714f647b70d32663cfa416f93f488ef6b6bc8fb2eb",
            "application_groups": [
                "application_groups value"
            ]
        }
    },
    {
        "key":
"oaccount~1da238c1491c919b151f777a615fb5779032c34b3ef3adeca6afe591bac10aaf",
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~1da238c1491c919b151f777a615fb5779032c34b3ef3adeca6afe591bac10aaf",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "balance":
"020da01cd57249c6470b6dd03c2ee4c49de58c71ccc64f2de1203e835967a0846d",
            "onhold_balance":
"020da01cd57249c6470b6dd03c2ee4c49de58c71ccc64f2de1203e835967a0846d",
            "onhold_burn_balance":
"020da01cd57249c6470b6dd03c2ee4c49de58c71ccc64f2de1203e835967a0846d",
            "application_groups": [
                "application_groups value"
            ]
        }
    }
]
```

`getAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string())
  public async getAllOrgAccounts(org_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAllOrgAccounts",
"TOKEN", { org_id });
    return await this.Ctx.Account.getAllOrgAccounts(org_id);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a list of all accounts for the specified organization.

Return Value Example:

```
[
    {
        "key":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
        "valueJson": {
            "bapAccountVersion": 2,
            "assetType": "oaccount",
            "account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance": "18",
            "onhold_balance": "0",
            "onhold_burn_balance": "0",
            "application_groups": [
                "application_groups value"
```


Copyright © 2025, Oracle and/or its affiliates.

```
            ],
            "user_id": "cb",
            "custom_account_id": "1234567jh"
        }
    },
    {
        "key":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
        "valueJson": {
            "bapAccountVersion": 1,
            "assetType": "oaccount",
            "account_id":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance": "22",
            "onhold_balance": "0",
            "onhold_burn_balance": "0",
            "application_groups": [
                "application_groups value"
            ],
            "user_id": "cb1",
            "custom_account_id": "1234567jh"
        }
    }
]
```

getAccountsByUser

This method returns a list of all account IDs for a specified organization ID and user ID. This method can be called only by the `Token Admin` or `Token Auditor`, by the `Org Admin` or `Org Auditor` of the specified organization, or by the `Account Owner` specified in the parameters.

```
@GetMethod()
  @Validator(yup.string(), yup.string())
  public async getAccountsByUser(org_id: string, user_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountsByUser",
"TOKEN", { org_id, user_id });
    return await this.Ctx.Account.getAccountsByUser(org_id, user_id);
  }
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id string` – The user name or email ID of the user.

Returns:

- On success, a JSON array of account IDs.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
    {
        "bapAccountVersion": 2,
        "assetType": "oaccount",
        "user_id": "cb",
        "custom_account_id": "1234567jh",
        "account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
        "org_id": "CentralBank",
        "token_type": "fungible",
        "token_id": "token",
        "token_name": "cbdc",
        "balance": "18",
        "onhold_balance": "0",
        "onhold_burn_balance": "0",
        "application_groups": [
            "application_groups value"
        ]
    }
]
```

`getUserByAccountId`
This method returns user details (`org_id` and `user_id`) for a specified account. This method can be called by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string())
  public async getUserByAccountId(account_id: string) {
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getUserByAccountId",
"TOKEN", { account_id });
    return await this.Ctx.Account.getUserByAccountId(account_id);
  }
```

Parameters:

- `account_id: string` – The ID of the account.

Returns:

- On success, a JSON object of the user details (`org_id`, `token_id`, and `user_id`).

Return Value Example:

```
{
    "token_id": "USD",
    "org_id": "CentralBank",
    "user_id": "cb_admin_demo",
    "custom_account_id": "10101234000123"
}
```


Copyright © 2025, Oracle and/or its affiliates.

getAccount
This method returns details for a specified account. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string())
  public async getAccount(account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccount", "TOKEN",
{ account_id });
    return await this.Ctx.Account.getAccountWithStatus(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
    "bapAccountVersion": 2,
    "assetType": "oaccount",
    "user_id": "cb",
    "custom_account_id": "1234567jh",
    "status": "active",
    "account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
    "org_id": "CentralBank",
    "token_type": "fungible",
    "token_id": "token",
    "token_name": "cbdc",
    "balance": "18",
    "onhold_balance": "0",
    "onhold_burn_balance": "0",
    "application_groups": [
        "application_groups value"
    ]
}
```

getAccountDetailsByCustomAccountId
This method returns details for a specified custom account ID. This method can be called only by a `Token Admin` or `Token Auditor`, or an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string(), yup.string())
  public async getAccountDetailsByCustomAccountId(custom_account_id: string,
org_id: string) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountDetailsByCustomAccountId",
 "TOKEN", { org_id });
    return await
```


Copyright © 2025, Oracle and/or its affiliates.

```
this.Ctx.Account.getAccountDetailsByCustomAccountId(custom_account_id,
org_id);
  }
```

Parameters:

- `custom_account_id: string` – The bank account ID of the account, a unique random alphanumeric identifier.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Return Value Example:

```
[
    {
        "bapAccountVersion": 0,
        "assetType": "oaccount",
        "account_id":
"oaccount~c44ffac4c46718e9744cb0aae2016d26a87a5bef5e2d1c0d1abc7d8782f0ba61",
        "org_id": "CentralBank",
        "token_type": "fungible",
        "token_id": "USD",
        "token_name": "cbdc",
        "balance": "0",
        "onhold_balance": "0",
        "onhold_burn_balance": "0",
        "application_groups": [
            "SYSTEM_ADMINS"
        ],
        "user_id": "cb_admin_demo",
        "custom_account_id": "10101234000123"
    }
]
```

`getAccountTransactionHistory`
This method returns an array of account transaction history details for a specified account. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string())
  public async getAccountTransactionHistory(account_id: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistory",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountTransactionHistory(account_id,
account.org_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:


```
[
    {
        "transaction_id":
"otransaction~4514aa229ebcc4d2fedcaa47c4301615e30c4a9bae45cf0256a5b80d75b3697a
",
        "transacted_amount": 1000,
        "timestamp": "2025-08-25T13:20:50.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 21000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~1982f73495060e0eef4d78282a91c41e27e8c95572739b0677a1e404a0d20aa9
",
        "transacted_amount": 200,
        "timestamp": "2025-08-25T13:12:43.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~fedd714cf1509f7517819d7cd4c0921d0b2f5d1ff6a25dcb08ab411defd6b5f3
",
        "transacted_amount": 2000,
        "timestamp": "2025-08-21T05:23:25.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~f33b47234f3ee0b636962c8c31c01d06523b789ca16b3b342d5080b71268bcc3
",
        "transacted_amount": 1000,
        "timestamp": "2025-08-21T05:23:07.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 20000,
        "onhold_balance": 0
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "transaction_id":
"otransaction~cf934527149bc24f62a8ddeeea7f74a19a0f84d8f161535a771be49d2520d5b3
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-13T06:12:41.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~f5c0e11ca61d9adc843658929e6de2a738ad586304f9e020f75bf4aac5e42a2c
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-13T06:12:04.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 10000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~862aa9d9e877d3ea209b87299ab5b12c13ed5ce43d1cf1b934043c1dd02f58f6
",
        "transacted_amount": 50000,
        "timestamp": "2025-08-12T21:04:22.000Z",
        "token_id": "USD",
        "category": "transfer",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transaction_type": "DEBIT",
        "balance": 10000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~8a74c6d87ca74a613aab9db5d40386f8d5b534f9800503af8ca27e8946d7616d
",
        "transacted_amount": 40000,
        "timestamp": "2025-08-12T21:01:27.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REJECT_MINT",
        "balance": 60000,
        "onhold_balance": 0
    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    {
        "transaction_id":
"otransaction~28ac66ba33f7ad0648448964b2b74525c9e3f0c9908c7a0484690b9baa56c2db
",
        "transacted_amount": 30000,
        "timestamp": "2025-08-12T21:01:16.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 60000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~7e32ad8f365ff59814e112f27602f30ab599fb9c1638784496c66a61a6277c22
",
        "transacted_amount": 20000,
        "timestamp": "2025-08-12T21:01:05.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 30000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~1477050bb9e55f4f471872b31fce0d2097f5d5e57d89a842070df5e36d7ab0da
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:01:03.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 10000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~0e76c6931b7ee134e967e847d9730b867a0fd191d39697d83d36dd15745c02e3
",
        "transacted_amount": 40000,
        "timestamp": "2025-08-12T21:00:20.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~07bbf9c190694371626da59ded5d87434d26f612891e13bb15bdd28f6086e760
",
        "transacted_amount": 30000,
        "timestamp": "2025-08-12T21:00:01.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~8721175c6cbbce17b6c4bb6a444e475d07f52352dfd0d990679f342215153513
",
        "transacted_amount": 20000,
        "timestamp": "2025-08-12T20:59:41.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~dc24c24d43a6525e807a39edcf8c6a2b6ccb81f0d755958f509509687eacee84
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T20:59:13.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~396e6ca5a11a9609632d0864026409d46a708fb95e3e21b39fa5f3fb78f90872
",
        "transacted_amount": 0,
        "timestamp": "2025-08-12T20:43:20.000Z",
        "token_id": "",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "CREATE_ACCOUNT",
        "balance": 0,
        "onhold_balance": 0
    }
]
```


`getAccountTransactionHistoryWithFilters`
This method returns a filtered array of account transaction history details for a specified user and token. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org`

Copyright © 2025, Oracle and/or its affiliates.

`Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account. This method can be called only when connected to a remote Oracle Blockchain Platform network.

```
@GetMethod()
  @Validator(yup.string(), yup.object().nullable())
  public async getAccountTransactionHistoryWithFilters(account_id: string,
filters?: Filters) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistoryWithFilt
ers", "TOKEN", { account_id });
    return await
this.Ctx.Account.getAccountTransactionHistoryWithFilters(account_id,
account.org_id, filters);
  }
```

Parameters:

- `account_id: string` – The unique ID of the account.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
           {
               "transaction_id":
"otransaction~ccc2c2e89ab7887af4fdad3dc9918ea057a8aa834a0ed8d0f23271049f084952
",
               "transacted_amount": 10,
               "timestamp": "2025-08-20T23:22:53.000Z",
               "token_id": "token",
               "category": "category value",
               "description": "description value",
               "holding_id": "ohold~cbdc~token~hold1",
               "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
               "to_org_id": "Org1",
               "to_user_id": "cb",
               "to_custom_account_id": "1234567jh",
               "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
               "from_org_id": "CentralBank",
               "from_user_id": "Not Available",
               "from_custom_account_id": "Not Available",
               "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
               "transaction_type": "CREDIT",
               "balance": 18,
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "onhold_balance": 0,
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
            {
                "transaction_id":
"otransaction~6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a
",
                "transacted_amount": 22,
                "timestamp": "2025-08-20T23:21:45.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "to_account_id":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
                "to_org_id": "Org1",
                "to_user_id": "cb1",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "from_org_id": "Org1",
                "from_user_id": "cb",
                "from_custom_account_id": "1234567jh",
                "transacted_account":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
                "transaction_type": "DEBIT",
                "balance": 8,
                "onhold_balance": 0,
                "transacted_org_id": "Org1",
                "transacted_user_id": "cb1",
                "transacted_custom_account_id": "1234567jh"
            },
            {
                "transaction_id":
"otransaction~3212811cd7c0ff265434f5921b93621cc5b0e0450b07c52b42ad9d8ce73cf063
",
                "transacted_amount": 10,
                "timestamp": "2025-08-20T23:18:53.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "holding_id": "ohold~cbdc~token~hold1",
                "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_org_id": "Org1",
                "to_user_id": "cb",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "from_org_id": "CentralBank",
                "from_user_id": "Not Available",
                "from_custom_account_id": "Not Available",
                "transacted_account":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "transaction_type": "CREDIT",
                "balance": 30,
                "onhold_balance": 0,
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
            {
                "transaction_id":
"otransaction~af2a1b0d56778e4ff5f4dda4e81bbcbf02c0592416ee2f105ff0b764e515fd0a
",
                "transacted_amount": 10,
                "timestamp": "2025-08-20T23:17:29.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "holding_id": "ohold~cbdc~token~hold1",
                "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_org_id": "Org1",
                "to_user_id": "cb",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "from_org_id": "CentralBank",
                "from_user_id": "Not Available",
                "from_custom_account_id": "Not Available",
                "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "transaction_type": "CREDIT",
                "balance": 20,
                "onhold_balance": 0,
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
            {
                "transaction_id":
"otransaction~6c0452a58f13db93f1c308d60eab1fde6c911921b350c0bbbe0ce7dab394721d
",
                "transacted_amount": 10,
                "timestamp": "2025-08-20T23:04:53.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "holding_id": "ohold~cbdc~token~hold1",
                "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_org_id": "Org1",
                "to_user_id": "cb",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "from_org_id": "CentralBank",
                "from_user_id": "Not Available",
                "from_custom_account_id": "Not Available",
                "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "transaction_type": "CREDIT",
                "balance": 10,
                "onhold_balance": 0,
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
            {
                "transaction_id":
"otransaction~df98e24a3b90661b54b54289b600b13a1051c922db0a4d20e3705a9820ae9c20
",
                "transacted_amount": 0,
                "timestamp": "2025-08-20T22:55:09.000Z",
                "category": "",
                "description": "",
                "from_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "from_org_id": "Org1",
                "from_user_id": "cb",
                "from_custom_account_id": "1234567jh",
                "transacted_account":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "transaction_type": "CREATE_ACCOUNT",
                "balance": 0,
                "onhold_balance": 0,
                "transacted_org_id": "Org1",
                "transacted_user_id": "cb",
                "transacted_custom_account_id": "1234567jh"
            }
        ]
```

`getAccountTransactionHistoryWithFiltersFromRichHistDB`
This method fetches transaction history from the rich history database. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string(), yup.string(), yup.string(),
yup.object().nullable())
  public async
getAccountTransactionHistoryWithFiltersFromRichHistDB(account_id: string,
custom_endpoint: string, bearer_token: string, filters?: Filters) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountTransactionHistoryWithFilt
ersFromRichHistDB", "TOKEN", { account_id });
    return await
```


Copyright © 2025, Oracle and/or its affiliates.

```
this.Ctx.Account.getAccountTrxHistoryWithFiltersFromRichHistDB(account_id,
account.org_id, custom_endpoint, bearer_token, filters);
  }
```

Parameters:

- `account_id: string` – The unique ID of the account.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
    {
        "transaction_id":
"otransaction~3140569a4ecb3c3f141cc2468fe21276640b7fd79013d951d9104b79072d8f9c
",
        "transacted_amount": 200,
        "timestamp": "2025-08-25T13:16:55.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "EXECUTEHOLD",
        "category": "transfer",
        "balance": 26800,
        "onhold_balance": 300
    },
    {
        "transaction_id":
"otransaction~2b75b3e8531a651f07c2d048d8546ad70ac49c66f0b82ed7626c1739090842ce
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:16:06.000Z",
        "token_id": "USD",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~e26f11da",
        "category": "transfer",
        "balance": 26800,
        "onhold_balance": 500
    },
    {
        "transaction_id":
"otransaction~9e7bf14cf96c5f90170da9455b1318687785e936192f60b7cbeb1c8bfabc41d2
",
        "transacted_amount": 100,
        "timestamp": "2025-08-21T06:57:19.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_retirer_demo",
        "transacted_custom_account_id": "10109999006543",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_retirer_demo",
        "from_custom_account_id": "10109999006543",
        "transaction_type": "CREDIT",
        "category": "transfer",
        "balance": 26900,
        "onhold_balance": 400
    },
    {
        "transaction_id":
"otransaction~b3901b4754920a9c75e36069dc55024ad505e4c127f334eedf65ef6703dc6b86
",
        "transacted_amount": 200,
        "timestamp": "2025-08-21T05:39:25.000Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "to_org_id": "org2",
        "to_user_id": "fi2_org_officer_demo",
        "to_custom_account_id": "30300617202404",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~77b75873",
        "category": "issuance",
        "balance": 26800,
        "onhold_balance": 400
    },
    {
        "transaction_id":
"otransaction~d55c9dfc9feacb353544b5d8b2ae694162ade3890bcaaf715503fd1d6a73cd1a
",
        "transacted_amount": 200,
        "timestamp": "2025-08-21T05:39:01.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~81d7c4ac",
        "category": "transfer",
        "balance": 27000,
        "onhold_balance": 200
    },
    {
        "transaction_id":
"otransaction~751eaedbe4311edd5d17cae53d283caf397d0cb09f18d57a5e3fe61266875ff9
",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "transacted_amount": 200,
            "timestamp": "2025-08-13T09:59:22.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
            "transacted_org_id": "org2",
            "transacted_user_id": "fi2_org_officer_demo",
            "transacted_custom_account_id": "30300617202404",
            "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "to_org_id": "CentralBank",
            "to_user_id": "cb_issuer_demo",
            "to_custom_account_id": "10109999001234",
            "from_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
            "from_org_id": "org2",
            "from_user_id": "fi2_org_officer_demo",
            "from_custom_account_id": "30300617202404",
            "transaction_type": "CREDIT",
            "category": "transfer",
            "balance": 27200,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~70155a8f4e388cc9395dbd03bedaf5a878705f5ad02302c8e9163218a5c3875a
",
            "transacted_amount": 1000,
            "timestamp": "2025-08-13T06:22:56.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
            "transacted_org_id": "Org1",
            "transacted_user_id": "fi1_org_officer_demo",
            "transacted_custom_account_id": "20200222221111",
            "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "to_org_id": "CentralBank",
            "to_user_id": "cb_issuer_demo",
            "to_custom_account_id": "10109999001234",
            "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
            "from_org_id": "Org1",
            "from_user_id": "fi1_org_officer_demo",
            "from_custom_account_id": "20200222221111",
            "transaction_type": "CREDIT",
            "category": "transfer",
            "balance": 27000,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~e595f3f0cc03fa5f58a546b8abbfaf155592e492f850581db2b8fed9a529c9e2
",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:09:25.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "transacted_org_id": "org2",
        "transacted_user_id": "fi2_org_officer_demo",
        "transacted_custom_account_id": "30300617202404",
        "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "to_org_id": "org2",
        "to_user_id": "fi2_org_officer_demo",
        "to_custom_account_id": "30300617202404",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "EXECUTEHOLD",
        "category": "issuance",
        "balance": 26000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~da92402859d87ae3069722d8e39cb0da448e9a5f67468233ee9b1fe7a4ebeef8
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:09:17.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "EXECUTEHOLD",
        "category": "issuance",
        "balance": 26000,
        "onhold_balance": 10000
    },
    {
        "transaction_id":
"otransaction~6915145aaf09fbf4d96456febddc2aa87b48c08ddd8ff17a6bab5d310f67bb36
",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "transacted_amount": 1000,
            "timestamp": "2025-08-12T21:07:11.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb_retirer_demo",
            "transacted_custom_account_id": "10109999006543",
            "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "to_org_id": "CentralBank",
            "to_user_id": "cb_issuer_demo",
            "to_custom_account_id": "10109999001234",
            "from_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
            "from_org_id": "CentralBank",
            "from_user_id": "cb_retirer_demo",
            "from_custom_account_id": "10109999006543",
            "transaction_type": "CREDIT",
            "category": "transfer",
            "balance": 26000,
            "onhold_balance": 20000
        },
        {
            "transaction_id":
"otransaction~244d7172d1dc90a142e1f22204c76614c7eea814b3d61f33016b786f1b347784
",
            "transacted_amount": 10000,
            "timestamp": "2025-08-12T21:05:39.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb_issuer_demo",
            "transacted_custom_account_id": "10109999001234",
            "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
            "to_org_id": "org2",
            "to_user_id": "fi2_org_officer_demo",
            "to_custom_account_id": "30300617202404",
            "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "from_org_id": "CentralBank",
            "from_user_id": "cb_issuer_demo",
            "from_custom_account_id": "10109999001234",
            "transaction_type": "ONHOLD",
            "holding_id": "ohold~cbdc~USD~ed815e20",
            "category": "issuance",
            "balance": 25000,
            "onhold_balance": 20000
        },
        {
            "transaction_id":
"otransaction~c63ec37966264493bde6fa666527b9cca11695c15611c32e89af49a2246f13f6
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:05:20.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~12d87129",
        "category": "issuance",
        "balance": 35000,
        "onhold_balance": 10000
    },
    {
        "transaction_id":
"otransaction~5112f576c94c2d23c342479bfa37e34612414b3258a64b43cf51b920f4ff5868
",
        "transacted_amount": 5000,
        "timestamp": "2025-08-12T21:05:02.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_retirer_demo",
        "transacted_custom_account_id": "10109999006543",
        "to_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_retirer_demo",
        "to_custom_account_id": "10109999006543",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "DEBIT",
        "category": "burn",
        "balance": 45000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~862aa9d9e877d3ea209b87299ab5b12c13ed5ce43d1cf1b934043c1dd02f58f6
",
        "transacted_amount": 50000,
        "timestamp": "2025-08-12T21:04:22.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb__creator_demo",
        "transacted_custom_account_id": "10105678004567",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "from_org_id": "CentralBank",
        "from_user_id": "cb__creator_demo",
        "from_custom_account_id": "10105678004567",
        "transaction_type": "CREDIT",
        "category": "transfer",
        "balance": 50000,
        "onhold_balance": 0
    }
]
```

`getOrgAccountsTransactionHistoryWithFiltersFromRichHistDB`
This method fetches transaction history from the rich history database for the specified organization. This method can be called only by the `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string(), yup.string(), yup.string(),
yup.object().nullable())
  public async
getOrgAccountsTransactionHistoryWithFiltersFromRichHistDB(org_id: string,
custom_endpoint: string, bearer_token: string, filters?: Filters) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getOrgAccountsTransactionHistoryWith
FiltersFromRichHistDB", "TOKEN", { org_id });
    return await
this.Ctx.Account.getOrgAccountsTrxHistoryWithFiltersFromRichHistDB(org_id,
custom_endpoint, bearer_token, filters);
  }
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.

Copyright © 2025, Oracle and/or its affiliates.

- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:


```
[
    {
        "transaction_id":
"otransaction~4793f3907eefce2f9fca7ef107405b0f116efb3afbf83fa0e61fe763690c8235
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:47:56.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~2ac01689",
        "category": "issuance"
    },
    {
        "transaction_id":
"otransaction~5177f7560d32838242a26ac74f2a90c6ff9b47aae0d0988f28d9b4cf7e27c097
",
        "transacted_amount": 10,
        "timestamp": "2025-08-25T13:22:23.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user1_demo",
        "transacted_custom_account_id": "20200198765432",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "from_custom_account_id": "20200198765432",
            "transaction_type": "CREDIT",
            "holding_id": "ohold~cbdc~USD~454f4bf6",
            "category": "transfer"
        },
        {
            "transaction_id":
"otransaction~5177f7560d32838242a26ac74f2a90c6ff9b47aae0d0988f28d9b4cf7e27c097
",
            "transacted_amount": 10,
            "timestamp": "2025-08-25T13:22:23.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
            "transacted_org_id": "Org1",
            "transacted_user_id": "fi1_org_user2_demo",
            "transacted_custom_account_id": "20200211112222",
            "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
            "to_org_id": "Org1",
            "to_user_id": "fi1_org_user2_demo",
            "to_custom_account_id": "20200211112222",
            "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
            "from_org_id": "Org1",
            "from_user_id": "fi1_org_user1_demo",
            "from_custom_account_id": "20200198765432",
            "transaction_type": "EXECUTEHOLD",
            "holding_id": "ohold~cbdc~USD~454f4bf6",
            "category": "transfer"
        },
        {
            "transaction_id":
"otransaction~32137cd7e16c560f4d96d0f8999f6a267c1e95be2f5b11ef0541e574a6cd7275
",
            "transacted_amount": 200,
            "timestamp": "2025-08-25T13:16:55.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "transacted_org_id": "CentralBank",
            "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
            "to_org_id": "Org1",
            "to_user_id": "fi1_org_officer_demo",
            "to_custom_account_id": "20200222221111",
            "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "from_org_id": "CentralBank",
            "transaction_type": "CREDIT",
            "category": "transfer"
        },
        {
            "transaction_id":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~67844e07c06ab4c3d94d8ff36173b9dbcc2f1b9c99f4e7070f1097fd89f3f514
",
        "transacted_amount": 10,
        "timestamp": "2025-08-21T08:49:40.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user1_demo",
        "transacted_custom_account_id": "20200198765432",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
        "from_custom_account_id": "20200198765432",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~b6e4925b",
        "category": "transfer"
    },
    {
        "transaction_id":
"otransaction~b7c97d737fb978651c9132276ab677c0bcf795b9db13d0d39cba5243615c7389
",
        "transacted_amount": 10,
        "timestamp": "2025-08-21T08:46:09.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user1_demo",
        "to_custom_account_id": "20200198765432",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~4fbbb846",
        "category": "transfer"
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

`getAllAccountsTransactionHistoryWithFiltersFromRichHistDB`
This method fetches transaction history from the rich history database for all organizations. This method can be called only by the `Token Admin` or `Token Auditor`.

```
@GetMethod()
  @Validator(yup.string(), yup.string(), yup.object().nullable())
  public async
getAllAccountsTransactionHistoryWithFiltersFromRichHistDB(custom_endpoint:
string, bearer_token: string, filters?: Filters) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getAllAccountsTransactionHistoryWith
FiltersFromRichHistDB", "TOKEN");
    return await
this.Ctx.Account.getAllAccountsTrxHistoryWithFiltersFromRichHistDB(custom_endp
oint, bearer_token, filters);
  }
```

Parameters:

- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
    {
        "transaction_id":
"otransaction~62eb436be7c29fc2ed9cae221e874d9a31b163fa10374e7da09bf5e09a96c3ff
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-25T13:57:58.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "from_org_id": "CentralBank",
        "from_user_id": "cb__creator_demo",
        "from_custom_account_id": "10105678004567",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "transaction_type": "DEBIT",
            "category": "issuance"
        },
        {
            "transaction_id":
"otransaction~62eb436be7c29fc2ed9cae221e874d9a31b163fa10374e7da09bf5e09a96c3ff
",
            "transacted_amount": 10000,
            "timestamp": "2025-08-25T13:57:58.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb__creator_demo",
            "transacted_custom_account_id": "10105678004567",
            "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "to_org_id": "CentralBank",
            "to_user_id": "cb_issuer_demo",
            "to_custom_account_id": "10109999001234",
            "from_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "from_org_id": "CentralBank",
            "from_user_id": "cb__creator_demo",
            "from_custom_account_id": "10105678004567",
            "transaction_type": "CREDIT",
            "category": "issuance"
        },
        {
            "transaction_id":
"otransaction~c628fb7738222ed969295ccc8d21b4be95d96e3aada4f14570f7820a7051b5f7
",
            "transacted_amount": 200,
            "timestamp": "2025-08-25T13:51:18.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb_issuer_demo",
            "transacted_custom_account_id": "10109999001234",
            "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
            "to_org_id": "org2",
            "to_user_id": "fi2_org_officer_demo",
            "to_custom_account_id": "30300617202404",
            "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "from_org_id": "CentralBank",
            "from_user_id": "cb_issuer_demo",
            "from_custom_account_id": "10109999001234",
            "transaction_type": "RELEASEHOLD",
            "holding_id": "ohold~cbdc~USD~77b75873"
        },
        {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transaction_id":
"otransaction~af2cc8e43ecb4c5520d90a8d7955b5a47623a29b13eef47e31c16eb48cc0adec
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:50:45.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_retirer_demo",
        "transacted_custom_account_id": "10109999006543",
        "from_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_retirer_demo",
        "from_custom_account_id": "10109999006543",
        "transaction_type": "REJECT_BURN",
        "holding_id": "ohold~cbdc~USD~8d34",
        "description": "Burn",
        "to_account": ""
    },
    {
        "transaction_id":
"otransaction~182b99bb2ed753994a8c638ab9b08c3a4e73ac8159a3173a2a1f56b651d2eeac
",
        "transacted_amount": 2000,
        "timestamp": "2025-08-25T13:50:19.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb__creator_demo",
        "transacted_custom_account_id": "10105678004567",
        "to_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "to_org_id": "CentralBank",
        "to_user_id": "cb__creator_demo",
        "to_custom_account_id": "10105678004567",
        "transaction_type": "REJECT_MINT",
        "holding_id": "ohold~cbdc~USD~89ce",
        "description": "Minting 2000 tokens",
        "from_account": ""
    },
    {
        "transaction_id":
"otransaction~4793f3907eefce2f9fca7ef107405b0f116efb3afbf83fa0e61fe763690c8235
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:47:56.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~2ac01689",
        "category": "issuance"
    },
    {
        "transaction_id":
"otransaction~5177f7560d32838242a26ac74f2a90c6ff9b47aae0d0988f28d9b4cf7e27c097
",
        "transacted_amount": 10,
        "timestamp": "2025-08-25T13:22:23.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user2_demo",
        "transacted_custom_account_id": "20200211112222",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
        "from_custom_account_id": "20200198765432",
        "transaction_type": "EXECUTEHOLD",
        "holding_id": "ohold~cbdc~USD~454f4bf6",
        "category": "transfer"
    }
]
```


`getAccountBalance`

This method returns the current balance for a specified account. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string())
  public async getAccountBalance(account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountBalance",
```


Copyright © 2025, Oracle and/or its affiliates.

```
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountBalance(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a JSON representation of the current account balance.

Return Value Example:

```
{
    "msg": "Current Balance is: 100",
    "user_balance": 100
}
```

`deleteHistoricalTransactions`
This method deletes older transactions from the state database. This method can be called only by a `Token Admin`.

```
@Validator(yup.date())
  public async deleteHistoricalTransactions(time_to_expiration: Date) {
    await this.Ctx.Auth.checkAuthorization("TRANSACTION.deleteTransactions",
"TOKEN");
    await
this.Ctx.Model.createEvent(EVENT_NAME.DELETE_HISTORICAL_TRANSACTIONS,
{ time_to_expiration });
    return await this.Ctx.Transaction.deleteTransactions(time_to_expiration);
  }
```

Parameters:

- `time_to_expiration Date` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted.

Return Value Example:

```
{
    "msg": "Successfully deleted transaction older than date: Thu Aug 19 2025
11:19:36 GMT+0000 (Coordinated Universal Time).",
    "transactions": [

"otransaction~ec3366dd48b4ce2838f820f2f138648e6e55a07226713e59b411ff31b0d21058
"
     ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

`getTransactionById`
This method returns the history of a `Transaction` asset. This method can be called only by a `Token Admin` or `Token Auditor`, by an `Org Admin` or `Org Auditor` of the specified organization, or by a transaction participant (sender, recipient, or notary).

```
@GetMethod()
  @Validator(yup.string())
  public async getTransactionById(transaction_id: string) {
    await this.Ctx.Auth.checkAuthorization("TRANSACTION.getTransactionById",
"TOKEN", { transaction_id });
    return await this.Ctx.Transaction.getTransactionById(transaction_id);
  }
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Returns:

- On success, an JSON array of the history for the transaction.

Return Value Example:

```
{
    "transaction_id":
"otransaction~6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a
",
    "history": [
        {
            "trxId":
"6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a",
            "timeStamp": "2025-08-20T23:21:45.000Z",
            "value": {
                "assetType": "otransaction",
                "transaction_id":
"otransaction~6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a
",
                "token_id": "token",
                "from_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_account_id":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
                "transaction_type": "TRANSFER",
                "amount": 22,
                "timestamp": "2025-08-20T23:21:45.000Z",
                "number_of_sub_transactions": 0,
                "holding_id": "",
                "sub_transaction": "false",
                "category": "category value",
                "description": "description value"
            }
        }
    ],
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "sub_transactions": []
}
```

getAccountStatus
This method gets the current status of the token account. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner.

```
@GetMethod()
  @Validator(yup.string())
  public async getAccountStatus(account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.get", "TOKEN",
{ account_id });
    try {
      return await this.Ctx.AccountStatus.getAccountStatus(account_id);
    } catch (err) {
      return await this.Ctx.AccountStatus.getDefaultAccountStatus(account_id);
    }
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a message that includes details of the token account status.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```

getAccountStatusHistory
This method gets the history of the account status. This method can be called by the `Token Admin` of the chaincode, an `Org Admin` of the specified organization, or by the token account owner.

```
@GetMethod()
  @Validator(yup.string())
  public async getAccountStatusHistory(account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.history", "TOKEN",
{ account_id });
    const status_id = await
```


Copyright © 2025, Oracle and/or its affiliates.

```
this.Ctx.AccountStatus.generateAccountStatusId(account_id);
    let account_status_history: any;
    try {
      account_status_history = await
this.Ctx.AccountStatus.history(status_id);
    } catch (err) {
      return [];
    }
    return account_status_history;
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a message that includes details of the account status history.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2025-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2025-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```


Copyright © 2025, Oracle and/or its affiliates.

activateAccount
This method activates a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. Deleted accounts cannot be activated.

```
@Validator(yup.string())
  public async activateAccount(account_id: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.activateAccount",
"TOKEN", { org_id: account.org_id });
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.ACTIVATE_ACCOUNT,
{ account_id }, token_asset);
    return await this.Ctx.Account.activateAccount(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

suspendAccount
This method suspends a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
@Validator(yup.string())
  public async suspendAccount(account_id: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.suspendAccount",
"TOKEN", { org_id: account.org_id });
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.SUSPEND_ACCOUNT, { account },
token_asset);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Account.suspendAccount(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

`deleteAccount`
This method deletes a token account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance and the on-hold balance must be zero.

```
@Validator(yup.string())
  public async deleteAccount(account_id: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT_STATUS.deleteAccount",
"TOKEN", { org_id: account.org_id });
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.DELETE_ACCOUNT,
{ account_id }, token_asset);
    return await this.Ctx.Account.deleteAccount(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a JSON representation of the account status object for the specified token account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

setMaxDailyAmount
This method sets the `max_daily_amount` value for a specified account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string().optional())
  public async setMaxDailyAmount (account_id: string, max_daily_amount?:
string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization('ACCOUNT.setMaxDailyAmount',
'TOKEN', { org_id: account.org_id });
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.SET_MAX_DAILY_AMOUNT,
{ account_id, max_daily_amount }, token_asset);
    return this.Ctx.Account.setMaxDailyAmount (account_id, max_daily_amount);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `max_daily_amount?: string` – (Optional) The maximum amount of tokens the user can transfer daily. If not specified, the user can transfer any amount of tokens daily.

Return Value Example:

```
{
           "msg": "Successfully set max daily amount for account id
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a to
1000000"
       }
```

setMaxDailyTransactionCount
This method sets the `max_daily_transactions` value for a specified account. This method can be called only by a `Token Admin` of the chaincode or an `Org Admin` of the specified organization.

```
@Validator(yup.string(), yup.string().optional())
  public async setMaxDailyTransactionCount (account_id: string,
max_daily_transactions?: string) {
    const account: FungibleAccount = await
```


Copyright © 2025, Oracle and/or its affiliates.

```
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await
this.Ctx.Auth.checkAuthorization('ACCOUNT.setMaxDailyTransactionCount',
'TOKEN', { org_id: account.org_id });
    const token_asset = await this.getTokenObject(account.token_id);
    await
this.Ctx.Model.createEvent(EVENT_NAME.SET_MAX_DAILY_TRANSACTION_COUNT,
{ account_id, max_daily_transactions}, token_asset);
    return this.Ctx.Account.setMaxDailyTransactionCount (account_id,
max_daily_transactions);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `max_daily_transactions?: string` – (Optional) The maximum number of transactions the user can complete daily. If not specified, the user can complete any number of transactions daily.

Return Value Example:

```
{
            "msg": "Successfully set max daily transactions for account id
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a to
100000"
        }
```

`getMaxDailyAmount`
This method gets the `max_daily_amount` value for a specified account. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode or an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string())
  public async getMaxDailyAmount (account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization('ACCOUNT.getMaxDailyAmount',
'TOKEN', { account_id });
    return this.Ctx.Account.getMaxDailyAmount (account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
           "max_daily_amount": "10000"
       }
```


Copyright © 2025, Oracle and/or its affiliates.

getMaxDailyTransactionCount
This method gets the `max_daily_transactions` value for a specified account. This method can be called only by a `Token Admin` or `Token Auditor` of the chaincode or an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string())
  public async getMaxDailyTransactionCount (account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await
this.Ctx.Auth.checkAuthorization('ACCOUNT.getMaxDailyTransactionCount',
'TOKEN', { account_id });
    return this.Ctx.Account.getMaxDailyTransactionCount (account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
           "max_daily_transactions": "100"
       }
```

consolidateRunningBalanceInTransactions
This method calculates the running account balances saves the updated values in the transaction key/value pairs. Typically the system administrators call this method by using a REST proxy scheduler. This method can be called only by a `Token Admin` or `Org Admin`.

```
@Validator(yup.string())
  public async consolidateRunningBalanceInTransactions() {
    await
this.Ctx.Auth.checkAuthorization("TRANSACTION.consolidateRunningBalanceInTrans
actions", "TOKEN");
    return this.Ctx.Transaction.consolidateRunningBalanceInTransactions(true);
  }
```

Return Value Example:

```
{ msg: "Successfully updated account running balance for pending
transactions."}
```

processSendersAndReceivers
This method calculates and update the token account balances that are spread in the sender and receiver key/value pairs that were created during inter-organization transfers, and then deletes the sender and receiver key/value pairs, which are no longer of use. Typically the

Copyright © 2025, Oracle and/or its affiliates.

system administrators call this method by using a REST proxy scheduler. This method can be called only by a `Token Admin` or `Org Admin`.

```
@Validator(yup.string())
  public async processSendersAndReceivers() {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.processSendersAndReceivers",
"TOKEN");
    return this.Ctx.Account.processSendersAndReceivers(true);
  }
```

Return Value Example:

```
{ msg: "Successfully updated balance for accounts from pending receivers."}
```

`addRole`
This method adds a role to a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also holds the specified role.

```
@Validator(yup.string(), yup.string())
  public async addRole(account_id: string, role: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.addRoleMember", "TOKEN",
{ token_id: account.token_id, org_id: account.org_id, role });
    await this.Ctx.Model.createEvent(EVENT_NAME.ADD_ROLE, { role,
account_id }, token_asset);
    return await this.Ctx.Token.addRoleMember(role, account_id, token_asset);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `role: string` – The name of the role to add to the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.

Returns:

- On success, a message with account details.

Return Value Example:

```
{
     "msg": "Successfully added role 'notary' to Account Id:
oaccount~2eb5f8a9bc561f8f41a4ea3be9511958cc6684ef14f2337ca396efc301b627d8"
}
```


Copyright © 2025, Oracle and/or its affiliates.

removeRole
This method removes a role from a specified user and token. This method can be called only by a `Token Admin` of the chaincode or by an `Org Admin` of the specified organization who also holds the specified role.

```
@Validator(yup.string(), yup.string())
  public async removeRole(account_id: string, role: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.removeRoleMember", "TOKEN",
{ token_id: account.token_id, org_id: account.org_id, role });
    await this.Ctx.Model.createEvent(EVENT_NAME.REMOVE_ROLE, { role,
account_id }, token_asset);
    return await this.Ctx.Token.removeRoleMember(role, account_id,
token_asset);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `role: string` – The name of the role to remove from the specified user. The `mintable` and `burnable` behaviors correspond to the `minter_role_name` and `burner_role_name` properties of the specification file. Similarly, the `notary` role corresponds to the `notary_role_name` property of the specification file.

Returns:

- On success, a message with account details.

Return Value Example:

```
{
   "msg": "Successfully removed role 'notary' from Account Id:
oaccount~2eb5f8a9bc561f8f41a4ea3be9511958cc6684ef14f2337ca396efc301b627d8"
}
```

getAccountsByRole
This method returns a list of all account IDs for a specified role and token. This method can be called only by a `Token Admin` or `Token Auditor`.

```
@GetMethod()
  @Validator(yup.string(), yup.string())
  public async getAccountsByRole(token_id: string, role: string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getAccountsByRole", "TOKEN");
    return await this.Ctx.Role.getAccountsByRole(token_id, role);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
           "accounts": [

"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75"
           ]
       }
```

`getOrgAccountsByRole`
This method returns a list of all account IDs for a specified role, token, and organization. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string(), yup.string(), yup.string())
  public async getOrgAccountsByRole(token_id: string, role: string, org_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getOrgAccountsByRole",
"TOKEN", { org_id });
    return await this.Ctx.Role.getOrgAccountsByRole(token_id, role, org_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Returns:

- On success, a JSON array of account IDs.

Return Value Example:

```
{
           "accounts": [

"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75"
           ]
       }
```

`getUsersByRole`
This method returns a list of all users for a specified role and token. This method can be called only by a `Token Admin` or `Token Auditor`.

```
@GetMethod()
  @Validator(yup.string(), yup.string())
```


Copyright © 2025, Oracle and/or its affiliates.

```
  public async getUsersByRole(token_id: string, role: string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getUsersByRole", "TOKEN");
    return await this.Ctx.Role.getUsersByRole(token_id, role);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Return Value Example:

```
{
          "users": [
              {
                  "token_id": "token",
                  "org_id": "CentralBank"
              }
          ]
      }
```

`getOrgUsersByRole`
This method returns a list of all users for a specified role, token, and organization. This method can be called only by a `Token Admin` or `Token Auditor`, or by an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string(), yup.string(), yup.string())
  public async getOrgUsersByRole(token_id: string, role: string, org_id:
string) {
    await this.Ctx.Auth.checkAuthorization("ROLE.getOrgUsersByRole", "TOKEN",
{ org_id });
    return await this.Ctx.Role.getOrgUsersByRole(token_id, role, org_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Return Value Example:

```
{
           "users": [
               {
                   "token_id": "token",
                   "org_id": "CentralBank",
                   "user_id": "cb1",
                   "custom_account_id": "1234567jh"
               }
```


Copyright © 2025, Oracle and/or its affiliates.

```
            ]
        }
```

isInRole
This method returns a Boolean value to indicate if a user has a specified role. This method can be called only by a `Token Admin` or `Token Auditor`, the `AccountOwner` of the account, or an `Org Admin` or `Org Auditor` of the specified organization.

```
@GetMethod()
  @Validator(yup.string(), yup.string())
  public async isInRole(account_id: string, role: string) {
    const account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    const token_asset = await this.getTokenObject(account.token_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.isInRole", "TOKEN",
{ account_id });
    return { result: await this.Ctx.Token.isInRole(role, account_id,
token_asset) };
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `role: string` – The name of the role to search for.

Returns:

- On success, a JSON string of the Boolean result.

Return Value Example:

```
{
            "result": "true"
        }
```

getTotalMintedTokens
This method returns the total number of minted tokens for a specified token. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin` or `Org Auditor`.

```
@GetMethod()
  @Validator(yup.string())
  public async getTotalMintedTokens(token_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.getTotalMintedTokens",
"TOKEN");
    const totalMintedTokens = await
this.Ctx.Token.getTotalMintedTokens(token_asset);
    return {
      msg: `Total minted token for Token Id: ${token_id} is $
{totalMintedTokens} tokens.`,
      quantity: totalMintedTokens,
```


Copyright © 2025, Oracle and/or its affiliates.

```
    };
  }
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON string indicating the total number of tokens.

Return Value Example:

```
{
    "msg": "Total minted token for Token Id: USD is 910 tokens.",
    "quantity": 910
}
```

`getNetTokens`
This method returns the total net number of tokens available in the system for a specified token. The net token total is the amount of tokens remaining after tokens are burned. In equation form: net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens. This method can be called only by a `Token Admin`, `Token Auditor`, `Org Admin` or `Org Auditor`.

```
@GetMethod()
  @Validator(yup.string())
  public async getNetTokens(token_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Auth.checkAuthorization("TOKEN.getNetTokens", "TOKEN");
    const netTokens = await this.Ctx.Token.getNetTokens(token_asset);
    return {
      msg: `Net supply of token for Token Id: ${token_id} is ${netTokens}
tokens.`,
      quantity: netTokens,
    };
  }
```

Parameters:

- `token_id: string` – The ID of the token.

Returns:

- On success, a JSON string indicating the net number of tokens.

Return Value Example:

```
{
    "msg": "Net supply of token for Token Id: USD is 878 tokens.",
    "quantity": 878
}
```


Copyright © 2025, Oracle and/or its affiliates.

`requestMint`
This method can be called by a minter to send a request to the minter notary to create a specified amount of tokens.

```
@Validator(
    yup.string(),
    yup.string(),
    yup.number().positive(),
    yup.date(),
    yup.object().nullable()
  )
  public async requestMint(
      operation_id: string,
      notary_account_id: string,
      quantity:number,
      time_to_expiration: Date,
      info_details ?: InfoDetails
  ) {
    const notary_account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(notary_account_id);
    const token_asset = await this.getTokenObject(notary_account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.REQUEST_MINT, { operation_id,
notary_account_id, quantity, time_to_expiration, info_details }, token_asset);
    return await this.Ctx.Hold.hold(operation_id, null, notary_account_id,
quantity, time_to_expiration, token_asset, HoldOperationType.MINT,
info_details);
  }
```

Parameters:

- `operation_id: string` – The unique operation ID that represents the mint request.
- `notary_account_id: string` – The unique account ID of the minter notary who will process the request.
- `quantity: number` – The amount of tokens to mint.
- `time_to_expiration` – The time after which the minting request expires and is no longer valid.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
           "msg": "AccountId
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41 has
```


Copyright © 2025, Oracle and/or its affiliates.

```
successfully submitted request to mint 200 tokens"
       }
```

approveMint
This method can be called by a minter notary to approve a minting request.

```
@Validator(yup.string(), yup.string())
  public async approveMint(token_id: string, operation_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.APPROVE_MINT, { token_asset,
operation_id }, token_asset);
    return await this.Ctx.Hold.executeHold(operation_id, token_asset,
HoldOperationType.MINT);
  }
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.

Return Value Example:

```
{
           "msg": "Successfully minted 1000 tokens to Account Id:
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41"
       }
```

rejectMint
This method can be called by a minter notary to reject a minting request.

```
@Validator(yup.string(), yup.string())
  public async rejectMint(token_id: string, operation_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.REJECT_MINT, { token_asset,
operation_id }, token_asset);
    return await this.Ctx.Hold.releaseHold(operation_id, token_asset,
HoldOperationType.MINT);
  }
```

Parameters:

- `token_id: string` – The ID of the token to mint.
- `operation_id: string` – The unique operation ID that represents the mint request.

Return Value Example:

```
{
           "msg": "Successfully rejected mint request with Operation Id
'89ce' to mint 2000 tokens of token id USD"
       }
```


Copyright © 2025, Oracle and/or its affiliates.

issueTokens
This method mints tokens, which are then owned by the caller of the method.

```
@Validator(yup.string(), yup.number().positive(), yup.object().nullable())
  public async issueTokens(token_id: string, quantity: number, info_details?:
InfoDetails) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.ISSUE_TOKENS, { quantity,
token_asset, info_details }, token_asset);
    return await this.Ctx.Token.mint(quantity, token_asset, info_details);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to mint.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
          "msg": "Successfully minted 1000 tokens to Account Id:
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41"
      }
```

transferTokens
This method transfers tokens from the caller to a specified account.

```
@Validator(yup.string(), yup.number(), yup.object().nullable())
  public async transferTokens(to_account_id: string, quantity: number,
info_details ?: InfoDetails) {
    if (quantity <= 0) {
      throw new Error("The quantity should be a positive number.")
    }
    const to_account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(to_account_id);
    const token_asset = await this.getTokenObject(to_account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.TRANSFER_TOKENS,
{ to_account_id, quantity, info_details }, token_asset);
    let isNotary = await this.Ctx.Token.checkNotary(to_account_id,
token_asset);
    if(isNotary) {
      throw new Error(`To Account Id '${to_account_id}' can not have a notary
```


Copyright © 2025, Oracle and/or its affiliates.

```
role!`);
    }
    return await this.Ctx.Token.transfer(to_account_id, quantity,
token_asset, info_details);
  }
```

Parameters:

- `to_account_id: string` – The unique account ID of the receiver (payee).
- `quantity: number` – The number of tokens to transfer.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
          "msg": "Successfully transferred 10000 tokens from account id:
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41  to
account id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a."
      }
```

`getAccountOnHoldBalance`
This method returns the current on-hold balance for a specified account. This method can be called only by a `Token Admin` or `Token Auditor`, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string())
  public async getAccountOnHoldBalance(account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getAccountOnHoldBalance",
"TOKEN", { account_id });
    return await this.Ctx.Account.getAccountOnHoldBalance(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a JSON representation of the current on-hold balance.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
           "msg": "Total Holding Balance is: 0",
           "holding_balance": 0
       }
```

`getOnHoldDetailsWithOperationId`
This method returns the on-hold transaction details for a specified operation ID and token. This method can be called by a `Token Admin` or `Token Auditor` of the chaincode, or by a transaction participant (sender, recipient, notary).

```
@GetMethod()
  @Validator(yup.string(), yup.string())
  public async getOnHoldDetailsWithOperationId(token_id: string,
operation_id: string) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getOnHoldDetailsWithOperationId",
"TOKEN", { token_id, operation_id });
    return await this.Ctx.Hold.getOnHoldDetailsWithOperationId(token_id,
operation_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Return Value Example:

```
{
          "assetType": "ohold",
          "holding_id": "ohold~cbdc~token~hold1",
          "operation_id": "hold1",
          "token_name": "cbdc",
          "from_org_id": "CentralBank",
          "operation_type": "transfer",
          "status": "approved",
          "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
          "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
          "notary_account_id":
"oaccount~1da238c1491c919b151f777a615fb5779032c34b3ef3adeca6afe591bac10aaf",
          "token_id": "token",
          "quantity": "0",
          "time_to_expiration": "2029-01-04T00:00:00.000Z",
          "category": "category value",
          "description": "description value"
      }
```


Copyright © 2025, Oracle and/or its affiliates.

`getOnHoldIds`
This method returns a list of all of the holding IDs for a specified account. This method can be called by a `Token Admin` or `Token Auditor` of the chaincode, an `Org Admin` or `Org Auditor` of the specified organization, or the `AccountOwner` of the account.

```
@GetMethod()
  @Validator(yup.string())
  public async getOnHoldIds(account_id: string) {
    await this.Ctx.Account.getAccountWithTokenIdValidation(account_id);
    await this.Ctx.Auth.checkAuthorization("ACCOUNT.getOnHoldIds", "TOKEN",
{ account_id });
    return await this.Ctx.Account.getOnHoldIds(account_id);
  }
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Returns:

- On success, a JSON list of holding IDs.

Return Value Example:

```
{
           "msg": "Holding Ids are: ",
           "holding_ids": ["ohold~cbdc~token~hold1"]
       }
```

`getOnHoldBalanceWithOperationId`
This method returns the on-hold balance for a specified operation ID and token. This method can be called by a `Token Admin` or `Token Auditor` of the chaincode, or by a transaction participant (sender, recipient, notary).

```
@GetMethod()
  @Validator(yup.string(), yup.string())
  public async getOnHoldBalanceWithOperationId(token_id: string,
operation_id: string) {
    await
this.Ctx.Auth.checkAuthorization("ACCOUNT.getOnHoldBalanceWithOperationId",
"TOKEN", { token_id, operation_id });
    return await this.Ctx.Hold.getOnHoldBalanceWithOperationId(token_id,
operation_id);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

Copyright © 2025, Oracle and/or its affiliates.

- On success, a JSON string indicating the holding balance.

Return Value Example:

```
{
           "msg": "Current Holding Balance of Operation 'hold1' for token
'token' is: 0",
           "holding_balance": 0
       }
```

`holdTokens`
This method creates a hold on behalf of the caller (the owner of the tokens) with the `to_account_id` account. The specified notary account is responsible for either completing or releasing the hold. When the hold is created, the specified token balance from the payer is put on hold. A held balance cannot be transferred until the hold is either completed or released.

```
@Validator(
    yup.string(),
    yup.string(),
    yup.string(),
    yup.number().positive(),
    yup.date(),
    yup.object().nullable()
  )
  public async holdTokens(
    operation_id: string,
    to_account_id: string,
    notary_account_id: string,
    quantity: number,
    time_to_expiration: Date,
    info_details ?: InfoDetails,
  ) {
    const to_account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(to_account_id);
    await this.Ctx.Account.getAccountWithTokenIdValidation(notary_account_id);
    const token_asset = await this.getTokenObject(to_account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.HOLD_TOKENS, {
      operation_id,
      to_account_id,
      notary_account_id,
      quantity,
      time_to_expiration,
      info_details
    }, token_asset);
    return await this.Ctx.Hold.hold(operation_id, to_account_id,
notary_account_id, quantity, time_to_expiration, token_asset,
HoldOperationType.TRANSFER, info_details);
  }
```

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Copyright © 2025, Oracle and/or its affiliates.

- `to_account_id: string` – The unique account ID of the receiver.
- `notary_account_id: string` – The unique account ID of the notary.
- `quantity: number` – The number of tokens to put on hold.
- `time_to_expiration` – The time when the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12:46:06Z`.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
           "msg": "AccountId
oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3 is
successfully holding 100 tokens"
       }
```

`executeHoldTokens`

This method completes a hold on a token. A quantity of tokens previously held by a token owner is transferred to a receiver. If the `quantity` value is less than the actual hold value, then the remaining amount is available again to the original owner of the tokens. This method can be called only by the `AccountOwner` ID with the `notary` role for the specified operation ID. The hold can only be completed by the notary.

```
@Validator(yup.string(), yup.string(), yup.number().positive())
  public async executeHoldTokens(token_id: string, operation_id: string,
quantity: number) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.EXECUTE_HOLD_TOKENS,
{ token_asset, operation_id, quantity }, token_asset);
    return await this.Ctx.Hold.executeHold(operation_id, token_asset,
HoldOperationType.TRANSFER, quantity);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `quantity: number` – The number of on-hold tokens to transfer.

Returns:

- On success, a message with the caller's account ID and the quantity of the transaction.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
          "msg": "Account Id:
oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111 is
successfully executed '10' tokens from Operation Id '454f4bf6'."
       }
```

`releaseHoldTokens`
This method releases a hold on tokens. The transfer is not completed and all held tokens are available again to the original owner. This method can be called by the `AccountOwner` ID with the `notary` role within the specified time limit or by the payer, payee, or notary after the specified time limit.

```
@Validator(yup.string(), yup.string())
  public async releaseHoldTokens(token_id: string, operation_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.RELEASE_HOLD_TOKENS,
{ token_asset, operation_id }, token_asset);
    return await this.Ctx.Hold.releaseHold(operation_id, token_asset,
HoldOperationType.TRANSFER);
  }
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.

Returns:

- On success, a message indicating that the hold was released.

Return Value Example:

```
{
          "msg": "Successfully released '200' tokens from Operation Id
'77b75873' to Account Id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a."
       }
```

`requestBurn`
This method can be called by a burner to send a request to the burner notary to destroy a specified amount of tokens.

```
@Validator(
    yup.string(),
    yup.string(),
    yup.number().positive(),
    yup.date(),
    yup.object().nullable()
  )
```


Copyright © 2025, Oracle and/or its affiliates.

```
  public async requestBurn(
      operation_id: string,
      notary_account_id: string,
      quantity:number,
      time_to_expiration: Date,
      info_details ?: InfoDetails,
  ) {
    const notary_account: FungibleAccount = await
this.Ctx.Account.getAccountWithTokenIdValidation(notary_account_id);
    const token_asset = await this.getTokenObject(notary_account.token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.REQUEST_BURN, { operation_id,
notary_account_id, quantity, time_to_expiration, info_details }, token_asset);
    return await this.Ctx.Hold.hold(operation_id, null, notary_account_id,
quantity, time_to_expiration, token_asset, HoldOperationType.BURN,
info_details);
  }
```

Parameters:

- `operation_id: string` – The unique operation ID that represents the burn request.
- `notary_account_id: string` – The unique account ID of the burner notary who will process the request.
- `quantity: number` – The amount of tokens to burn.
- `time_to_expiration` – The time after which the burning request expires and is no longer valid.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
           "msg": "AccountId
oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98 has
successfully submitted request to burn 100 tokens"
       }
```

approveBurn
This method can be called by a burner notary to approve a burning request.

```
@Validator(yup.string(), yup.string())
  public async approveBurn(token_id: string, operation_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.APPROVE_BURN, { token_asset,
operation_id }, token_asset);
```


Copyright © 2025, Oracle and/or its affiliates.

```
    return await this.Ctx.Hold.executeHold(operation_id, token_asset,
HoldOperationType.BURN);
  }
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.

Return Value Example:

```
{
           "msg": "Successfully burned 200 tokens from account id:
oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98"
       }
```

`rejectBurn`
This method can be called by a burner notary to reject a burning request.

```
@Validator(yup.string(), yup.string())
  public async rejectBurn(token_id: string, operation_id: string) {
    const token_asset = await this.getTokenObject(token_id);
    await this.Ctx.Model.createEvent(EVENT_NAME.REJECT_BURN, { token_asset,
operation_id }, token_asset);
    return await this.Ctx.Hold.releaseHold(operation_id, token_asset,
HoldOperationType.BURN);
  }
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the burn request.

Return Value Example:

```
{
           "msg": "Successfully rejected burn request with Operation Id
'8d34' to burn 100 tokens of token id USD"
       }
```

TypeScript Methods for Inter-Organizational Transfers

Inter-organization transfers have two parts: first the amount is debited from the sender and then the amount is credited to the receiver. The following two APIs are called together as an atomic transaction using the two-phase commit API of the Oracle Blockchain Platform REST proxy. Confidential transfers are done in this manner because the private data collections of the two organizations are distinct and cannot be accessed in a single transaction.

Both transactions are sent using the two-phase commit API of the REST proxy. For more information, see Invoke Atomic Transaction.

You can create a POST request in the Postman collection where the URL is similar to: `https://test-xyz-abc.blockchain.ocp.oraclecloud.com:7443/`

Copyright © 2025, Oracle and/or its affiliates.

`restproxy/api/v2/atomicTransactions`. You must send all essential headers including the following ones.

```
- Content-Type: application/json
- Confidential-Transaction: true
```

In the request body, include the `executeHoldTokensSender`and `executeHoldTokensReceiver` methods as shown in the following request.

```
{
 "transactions": [
   {
        "chaincode": "<chaincode>",
        "args": [
            "executeHoldTokensSender"
        ],
        "timeout": 6000,
        "sync": true,
        "transientMap": {
            "args": "[ \"bc-token-id value\",\"operation_id
value\",\"quantity value\"]"
        },
        "endorsers" : <Sender's endorsers List>,
        "channel":"<channel name>"
   },
   {
        "chaincode": "<chaincode name>",
        "args": [
            "executeHoldTokensReceiver"
        ],
        "timeout": 6000,
        "sync": true,
        "transientMap": {
            "args": "[ \"bc-token-id value\",\"operation_id
value\",\"quantity value\"]"
        },
        "endorsers" : <Recipient endorsers List>,
        "channel":"<channelName>"
    }
 ],
 "isolationLevel": "readCommitted",
 "prepareTimeout": 10000,
 "copyInputsToTransientMap": true,
 "sync": true,
 "parallelPrepare": true
}
```

The following example shows a request body.

```
{
 "transactions": [
   {
        "chaincode": "WholesaleCBDCConfidential",
        "args": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "executeHoldTokensSender"
        ],
        "timeout": 6000,
        "sync": true,
        "transientMap": {
            "args": "[ \"bc-token-id value\",\"operation_id
value\",\"quantity value\"]"
        },
        "endorsers" : ["org1-xyz-abc.blockchain.ocp.oraclecloud.com:20009",
"org1-xyz-abc.blockchain.ocp.oraclecloud.com:20009"],
        "channel":"default"
   },
   {
        "chaincode": "WholesaleCBDCConfidential",
        "args": [
            "executeHoldTokensReceiver"
        ],
        "timeout": 6000,
        "sync": true,
        "transientMap": {
            "args": "[ \"bc-token-id value\",\"operation_id
value\",\"quantity value\"]"
        },
        "endorsers" : ["org2-xyz-abc.blockchain.ocp.oraclecloud.com:20009",
"org2-xyz-abc.blockchain.ocp.oraclecloud.com:20009"],
        "channel":"default"
    }
 ],
 "isolationLevel": "readCommitted",
 "prepareTimeout": 10000,
 "copyInputsToTransientMap": true,
 "sync": true,
 "parallelPrepare": true
}
```

executeHoldTokensSender
This method can be called only by the user with the notary role for the specified operation ID. This method is the first part of an approval of an transfer between organizations. The specified amount is deducted from the sender's account.

```
@ExcludeFromPostmanCollection()
  @Validator(yup.string(), yup.string(), yup.number().positive(),
yup.string(), yup.string())
  public async executeHoldTokensSender(token_id: string, operation_id:
string, quantity: number,  isolationLevel: string, globalIdInput:string) {
    if (!this.Ctx.Utils.isValidInterOrgTransferRequest()) {
      throw new Error("The request for the inter-org transfer is invalid.")
    }
    const globalTxId =
this.Ctx.Utils.extractGlobalTransactionId(globalIdInput);
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Hold.executeHoldSender(operation_id, quantity,
```


Copyright © 2025, Oracle and/or its affiliates.

```
globalTxId, token_asset);
  }
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the hold request to approve.
- `quantity: number` – The amount of held tokens to transfer.
- `isolationLevel: string` – The isolation level of the atomic transaction, either `serializable` or `readCommitted`.
- `globalIdInput: string`

Return Value Example:

```
{
      "msg":
        "Account Id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a is
successfully executed '10' tokens from Operation Id 'hold1'."
    }
```

`executeHoldTokensReceiver`
This method can be called only by the user with the notary role for the specified operation ID. This method is the second part of an approval of an transfer between organizations. The specified amount is credited to the receiver's account.

```
@ExcludeFromPostmanCollection()
  @Validator(yup.string(), yup.string(), yup.number().positive(),
yup.string(), yup.string())
  public async executeHoldTokensReceiver(token_id: string, operation_id:
string, quantity: number,  isolationLevel: string, globalIdInput:string) {
    if (!this.Ctx.Utils.isValidInterOrgTransferRequest()) {
      throw new Error("The request for the inter-org transfer is invalid.")
    }
    const globalTxId =
this.Ctx.Utils.extractGlobalTransactionId(globalIdInput);
    const token_asset = await this.getTokenObject(token_id);
    return await this.Ctx.Hold.executeHoldReceiver(operation_id, quantity,
globalTxId, token_asset);
  }
```

Parameters:

- `token_id: string` – The ID of the token to burn.
- `operation_id: string` – The unique operation ID that represents the hold request to approve.
- `quantity: number` – The amount of held tokens to transfer.
- `isolationLevel: string` – The isolation level of the atomic transaction, either `serializable` or `readCommitted`.

Copyright © 2025, Oracle and/or its affiliates.

- `globalIdInput: string`

Return Value Example:

```
{
      "msg":
        "Account Id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a is
successfully executed '10' tokens from Operation Id 'hold1, receiverId
oreceiver~72d9bfcf-2c68-4c33-b8c3-fe3374983bf2'."
    }
```

Modified TypeScript SDK Methods for Confidential Chaincode

If you include the `confidential: true` parameter in the specification file for tokens that use the extended Token Taxonomy Framework standard, Blockchain App Builder generates modified versions of the standard SDK methods as well as additional methods.

`isUserTokenAdmin`
This method returns the Boolean value `true` if the caller of the function is a `Token Admin`. Otherwise the method returns `false`.

```
Ctx.Auth.isUserTokenAdmin(org_id: string, user_id: string);
```

Parameters:

- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.
- `user_id` – The user name or email ID of the user.

Return Value Example:

```
{
    "result": true
}
```

`addTokenAdmin`
This method adds a user as a `Token Admin` of the token chaincode. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
Ctx.Admin.addTokenAdmin(org_id: string, user_id: string);
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "msg": "Successfully added Token Admin (Org_Id: CentralBank, User_Id: cb1)"
}
```

removeTokenAdmin
This method removes a user as a `Token Admin` of the token chaincode.

```
Ctx.Admin.removeTokenAdmin(org_id: string, user_id: string);
```

Parameters:

- `user_id` – The user name or email ID of the user.
- `org_id` – The membership service provider (MSP) ID of the user in the current network organization.

Return Value Example:

```
{"msg": "Successfully removed Admin (Org_Id: Org1MSP, User_Id: User1)"}
```

getAllTokenAdmins
This method returns a list of all users who are a `Token Admin` of the chaincode.

```
Ctx.Admin.getAllTokenAdmins();
```

Parameters:

- none

Return Value Example:

```
{"admins":[{"org_id":"Org1MSP","user_id":"admin"}]}
```

addOrgAdmin
This method adds a user as an `Org Admin` of the organization. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
 Ctx.Admin.addOrgAdmin(org_id: string, user_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
    "msg": "Successfully added Org Admin (Org_Id: Org1MSP, User_Id: orgAdmin)"
}
```

`removeOrgAdmin`
This method removes a user as an `Org Admin` of the organization.

```
Ctx.Admin.removeOrgAdmin(org_id: string, user_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
  "msg": "Successfully removed Org Admin (Org_Id Org1MSP User_Id orgAdmin)"
}
```

`getAllOrgAdmins`
This method returns a list of all users who are an `Org Admin` of an organization.

```
Ctx.Admin.getAllOrgAdmins();
```

Parameters:

- none

Return Value Example:

```
{
    "admins": [
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin1"
        },
        {
            "org_id": "Org1MSP",
            "user_id": "orgadmin2"
        }
    ]
}
```


Copyright © 2025, Oracle and/or its affiliates.

addTokenAuditor
This method adds a user as a `Token Auditor` of the chaincode. Because the `user_id` value is stored on the public ledger, do not use any personally identifiable information (PII) for the `user_id` value.

```
Ctx.Admin.addTokenAuditor(org_id: string, user_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Returns:

- On success, a message that includes details of the user who was added as a `Token Auditor` of the chaincode.

Return Value Example:

```
{
  "msg": "Successfully added Token Auditor (Org_Id: CentralBank, User_Id:
cb_admin_demo)"
}
```

removeTokenAuditor
This method removes a user as a `Token Auditor` of the chaincode.

```
Ctx.Admin.removeTokenAuditor(org_id: string, user_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
          "msg": "Successfully removed Token Auditor (Org_Id: CB, User_Id:
cb)"
      }
```

getAllTokenAuditors
This method returns all `Token Auditors` of the chaincode.

```
this.Ctx.Admin.getAllTokenAuditors();
```


Copyright © 2025, Oracle and/or its affiliates.

addOrgAuditor
This method adds a user as a `Org Auditor` of the chaincode.

```
Ctx.Admin.addOrgAuditor(org_id: string, user_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
          "msg": "Successfully added Org Auditor (Org_Id: CentralBank,
User_Id: cb_admin_demo)"
      }
```

removeOrgAuditor
This method removes a user as a `Org Auditor` of the chaincode.

```
this.Ctx.Admin.removeOrgAuditor(org_id: string, user_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization.
- `user_id: string` – The user name or email ID of the user.

Return Value Example:

```
{
          "msg": "Successfully removed Org Auditor (Org_Id: CB, User_Id: cb)"
      }
```

getAllOrgAuditors
This method returns all `Org Auditors` of the chaincode.

```
this.Ctx.Admin.getAllOrgAuditors()
```

getAllTokens
This method returns all the token assets saved in the state database. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
this.Ctx.Token.getAllTokens();
```

Parameters:

- none

Copyright © 2025, Oracle and/or its affiliates.

Returns:

- On success, it returns a promise with all the token assets. On error, it returns an error message.

`getDecimals`
This method returns the number of decimal places available for a fractional token. If the `divisible` behavior is not specified, then the default value is 0.

```
this.Ctx.Token.getDecimals(token_asset)
```

Parameters:

- `token_id: string` – The ID of the token.

Return Value Example:

```
1
```

`history`
This method returns history for the specified token.

```
Ctx.Token.history(tokenId: string);
```

Parameters:

- `token_id: string` – The ID of the token.

Return Value Example:

```
[
    {
        "trxId":
"0d75f09446a60088afb948c6aca046e261fddcd43df416076201cdc5565f1a35",
        "timeStamp": "2023-09-01T16:48:41.000Z",
        "value": {
            "assetType": "otoken",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "token_desc": "updatedDesc",
            "token_standard": "ttf+",
            "token_type": "fungible",
            "token_unit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "mintable": {
                "max_mint_quantity": 1000
            },
            "divisible": {
                "decimal": 2
            }
        }
    },
    {
        "trxId":
"3666344878b043b65d5b821cc79c042ba52aec467618800df5cf14eac69f72fa",
        "timeStamp": "2023-08-31T20:24:55.000Z",
        "value": {
            "assetType": "otoken",
            "token_id": "token",
            "token_name": "fiatmoneytok",
            "token_standard": "ttf+",
            "token_type": "fungible",
            "token_unit": "fractional",
            "behaviors": [
                "divisible",
                "mintable",
                "transferable",
                "burnable",
                "roles"
            ],
            "roles": {
                "minter_role_name": "minter"
            },
            "mintable": {
                "max_mint_quantity": 1000
            },
            "divisible": {
                "decimal": 2
            }
        }
    }
]
```

`getTokensByName`
This method returns all the token assets with the specified name. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Token.getTokensByName(token_name: string);
```

Parameters:

- `token_name: string` – The name of the token, which corresponds to the `Token_name` property of the model. The value is the class name of the token.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
[
 {
    "assetType": "otoken",
    "token_id": "digiCurr101",
    "token_name": "digicur",
    "token_desc": "Digital Currency equiv of dollar",
    "token_type": "fungible",
    "behaviors": [
        "divisible",
        "mintable",
        "transferable",
        "burnable",
        "roles"
    ],
    "roles": {
        "minter_role_name": "minter"
        "burner_role_name": "burner",
        "notary_role_name": "notary"
    },
    "mintable": {
        "max_mint_quantity": 2000
    },
    "divisible": {
        "decimal": 1
    },
    "currency_name": "DOLLAR",
    "token_to_currency_ratio": 1
 }
]
```

`createAccount`
This method creates an account for a specified user and token. Every user who has tokens at any point must have an account.

```
Ctx.Account.createAccount(org_id: string, user_id: string, token_type:
string, dailyLimits);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the user in the current organization. The ID must begin with an alphanumeric character and can include letters, numbers, and special characters such as underscores (_), periods (.), at signs (@), and hyphens (-).
- `user_id: string` – The user name or email ID of the user. The ID must begin with an alphanumeric character and can include letters, numbers, and special characters such as underscores (_), periods (.), at signs (@), and hyphens (-).
- `token_type: string` – The type of the token, which must be `fungible`.

Copyright © 2025, Oracle and/or its affiliates.

- `daily_limits: DailyLimits` – A JSON object of the following type.

```
{
     "max_daily_amount": 100000
     "max_daily_transactions": 10000
 }
```

  In the example, the `max_daily_amount` value is the maximum amount of tokens that can be transacted daily and `max_daily_transactions` value is the maximum number of transactions that can be completed daily.

Return Value Example:

```
{
    "bapAccountVersion": 0,
    "assetType": "oaccount",
    "account_id":
"oaccount~b53cb2c19c92d1d5c8cb9f6e988e7761c34e03e014e6c4b889565fc0abf46c8a",
    "org_id": "CentralBank",
    "token_type": "fungible",
    "token_id": "",
    "token_name": "",
    "balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_burn_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6"
}
```

`associateToken`
This method associates a fungible token with an account.

```
Ctx.Account.associateToken(account_id: string, token_id: string,
custom_account_id?: string);
```

Parameters:

- `account_id: string` – The ID of the account.
- `token_id: string` – The ID of the token.
- `custom_account_id: string` – For confidential mode, the bank account ID of the account, a unique random alphanumeric identifier. By default, 14 characters long.

Return Value Example:

```
{
    "bapAccountVersion": 0,
    "assetType": "oaccount",
    "account_id":
"oaccount~b53cb2c19c92d1d5c8cb9f6e988e7761c34e03e014e6c4b889565fc0abf46c8a",
    "org_id": "CentralBank",
```


Copyright © 2025, Oracle and/or its affiliates.

```
    "token_type": "fungible",
    "token_id": "USD",
    "token_name": "cbdc",
    "balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "onhold_burn_balance":
"028b72742c8aa9a0395c828fe4f0e46226a3e40d4e731d0b994c8028c8b7bd4df6",
    "application_groups": [
        "CENTRAL_BANK_USERS"
    ]
}
```

`getAllAccounts`
This method returns a list of all accounts. This method uses Berkeley DB SQL rich queries and can only be called when connected to the remote Oracle Blockchain Platform network.

```
this.Ctx.Account.getAllAccounts();
```

Parameters:

- none

Return Value Example:

```
[
    {
        "key":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
        "valueJson": {
            "bapAccountVersion": 3,
            "assetType": "oaccount",
            "account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance":
"028a27c752e9b518d156e61da6589e723b179b6d7351dc34ec92bbb27b783ed1dc",
            "onhold_balance":
"02d53a7eda9484bda7252212714f647b70d32663cfa416f93f488ef6b6bc8fb2eb",
            "onhold_burn_balance":
"02d53a7eda9484bda7252212714f647b70d32663cfa416f93f488ef6b6bc8fb2eb",
            "application_groups": [
                "application_groups value"
            ]
        }
    },
    {
        "key":
"oaccount~1da238c1491c919b151f777a615fb5779032c34b3ef3adeca6afe591bac10aaf",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "valueJson": {
            "bapAccountVersion": 0,
            "assetType": "oaccount",
            "account_id":
"oaccount~1da238c1491c919b151f777a615fb5779032c34b3ef3adeca6afe591bac10aaf",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance":
"020da01cd57249c6470b6dd03c2ee4c49de58c71ccc64f2de1203e835967a0846d",
            "onhold_balance":
"020da01cd57249c6470b6dd03c2ee4c49de58c71ccc64f2de1203e835967a0846d",
            "onhold_burn_balance":
"020da01cd57249c6470b6dd03c2ee4c49de58c71ccc64f2de1203e835967a0846d",
            "application_groups": [
                "application_groups value"
            ]
        }
    }
]
```

`getAllOrgAccounts`
This method returns a list of all token accounts that belong to a specified organization.

```
Ctx.Account.getAllOrgAccounts(org_id: string);
```

Parameters:

- `org_id: string` – The membership service provider (MSP) ID of the organization.

Return Value Example:

```
[
    {
        "key":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
        "valueJson": {
            "bapAccountVersion": 2,
            "assetType": "oaccount",
            "account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance": "18",
            "onhold_balance": "0",
            "onhold_burn_balance": "0",
            "application_groups": [
                "application_groups value"
            ],
            "user_id": "cb",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "custom_account_id": "1234567jh"
        }
    },
    {
        "key":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
        "valueJson": {
            "bapAccountVersion": 1,
            "assetType": "oaccount",
            "account_id":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
            "org_id": "CentralBank",
            "token_type": "fungible",
            "token_id": "token",
            "token_name": "cbdc",
            "balance": "22",
            "onhold_balance": "0",
            "onhold_burn_balance": "0",
            "application_groups": [
                "application_groups value"
            ],
            "user_id": "cb1",
            "custom_account_id": "1234567jh"
        }
    }
]
```

getAccountsByUser
This method returns a list of all account IDs for a specified user and organization.

```
Ctx.Account.getAccountsByUser(org_id: string, user_id: string);
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the organization.
- `user_id string` – The user name or email ID of the user.

Return Value Example:

```
[
    {
        "bapAccountVersion": 2,
        "assetType": "oaccount",
        "user_id": "cb",
        "custom_account_id": "1234567jh",
        "account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
        "org_id": "CentralBank",
        "token_type": "fungible",
        "token_id": "token",
        "token_name": "cbdc",
        "balance": "18",
        "onhold_balance": "0",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "onhold_burn_balance": "0",
        "application_groups": [
            "application_groups value"
        ]
    }
]
```

getUserByAccountId
This method returns the user details for a specified account.

```
this.Ctx.Account.getUserByAccountId(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
    "token_id": "USD",
    "org_id": "CentralBank",
    "user_id": "cb_admin_demo",
    "custom_account_id": "10101234000123"
}
```

getAccountWithStatus
This method returns account details for a specified user and token.

```
Ctx.Account.getAccountWithStatus(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
    "bapAccountVersion": 2,
    "assetType": "oaccount",
    "user_id": "cb",
    "custom_account_id": "1234567jh",
    "status": "active",
    "account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
    "org_id": "CentralBank",
    "token_type": "fungible",
    "token_id": "token",
    "token_name": "cbdc",
    "balance": "18",
    "onhold_balance": "0",
    "onhold_burn_balance": "0",
    "application_groups": [
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "application_groups value"
    ]
}
```

getAccountDetailsByCustomAccountId
This method returns all token account details for a specified organization and account ID.

```
Ctx.Account.getAccountDetailsByCustomAccountId(custom_account_id: string,
org_id: string)
```

Parameters:

- `custom_account_id: string` – The bank account ID of the account, a unique random alphanumeric identifier.
- `org_id string` – The membership service provider (MSP) ID of the organization.

Return Value Example:

```
[
    {
        "bapAccountVersion": 0,
        "assetType": "oaccount",
        "account_id":
"oaccount~c44ffac4c46718e9744cb0aae2016d26a87a5bef5e2d1c0d1abc7d8782f0ba61",
        "org_id": "CentralBank",
        "token_type": "fungible",
        "token_id": "USD",
        "token_name": "cbdc",
        "balance": "0",
        "onhold_balance": "0",
        "onhold_burn_balance": "0",
        "application_groups": [
            "SYSTEM_ADMINS"
        ],
        "user_id": "cb_admin_demo",
        "custom_account_id": "10101234000123"
    }
]
```

getAccountTransactionHistory
This method returns an array of the transaction history details for a specified account.

```
Ctx.Account.getAccountTransactionHistory(account_id: string)
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
[
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transaction_id":
"otransaction~4514aa229ebcc4d2fedcaa47c4301615e30c4a9bae45cf0256a5b80d75b3697a
",
        "transacted_amount": 1000,
        "timestamp": "2025-08-25T13:20:50.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 21000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~1982f73495060e0eef4d78282a91c41e27e8c95572739b0677a1e404a0d20aa9
",
        "transacted_amount": 200,
        "timestamp": "2025-08-25T13:12:43.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~fedd714cf1509f7517819d7cd4c0921d0b2f5d1ff6a25dcb08ab411defd6b5f3
",
        "transacted_amount": 2000,
        "timestamp": "2025-08-21T05:23:25.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~f33b47234f3ee0b636962c8c31c01d06523b789ca16b3b342d5080b71268bcc3
",
        "transacted_amount": 1000,
        "timestamp": "2025-08-21T05:23:07.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~cf934527149bc24f62a8ddeeea7f74a19a0f84d8f161535a771be49d2520d5b3
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-13T06:12:41.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "APPROVE_MINT",
        "balance": 20000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~f5c0e11ca61d9adc843658929e6de2a738ad586304f9e020f75bf4aac5e42a2c
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-13T06:12:04.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 10000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~862aa9d9e877d3ea209b87299ab5b12c13ed5ce43d1cf1b934043c1dd02f58f6
",
        "transacted_amount": 50000,
        "timestamp": "2025-08-12T21:04:22.000Z",
        "token_id": "USD",
        "category": "transfer",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transaction_type": "DEBIT",
        "balance": 10000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~8a74c6d87ca74a613aab9db5d40386f8d5b534f9800503af8ca27e8946d7616d
",
        "transacted_amount": 40000,
        "timestamp": "2025-08-12T21:01:27.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REJECT_MINT",
        "balance": 60000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~28ac66ba33f7ad0648448964b2b74525c9e3f0c9908c7a0484690b9baa56c2db
",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "transacted_amount": 30000,
            "timestamp": "2025-08-12T21:01:16.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transaction_type": "APPROVE_MINT",
            "balance": 60000,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~7e32ad8f365ff59814e112f27602f30ab599fb9c1638784496c66a61a6277c22
",
            "transacted_amount": 20000,
            "timestamp": "2025-08-12T21:01:05.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transaction_type": "APPROVE_MINT",
            "balance": 30000,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~1477050bb9e55f4f471872b31fce0d2097f5d5e57d89a842070df5e36d7ab0da
",
            "transacted_amount": 10000,
            "timestamp": "2025-08-12T21:01:03.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transaction_type": "APPROVE_MINT",
            "balance": 10000,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~0e76c6931b7ee134e967e847d9730b867a0fd191d39697d83d36dd15745c02e3
",
            "transacted_amount": 40000,
            "timestamp": "2025-08-12T21:00:20.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transaction_type": "REQUEST_MINT",
            "balance": 0,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~07bbf9c190694371626da59ded5d87434d26f612891e13bb15bdd28f6086e760
",
            "transacted_amount": 30000,
            "timestamp": "2025-08-12T21:00:01.000Z",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~8721175c6cbbce17b6c4bb6a444e475d07f52352dfd0d990679f342215153513
",
        "transacted_amount": 20000,
        "timestamp": "2025-08-12T20:59:41.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~dc24c24d43a6525e807a39edcf8c6a2b6ccb81f0d755958f509509687eacee84
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T20:59:13.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "REQUEST_MINT",
        "balance": 0,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~396e6ca5a11a9609632d0864026409d46a708fb95e3e21b39fa5f3fb78f90872
",
        "transacted_amount": 0,
        "timestamp": "2025-08-12T20:43:20.000Z",
        "token_id": "",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transaction_type": "CREATE_ACCOUNT",
        "balance": 0,
        "onhold_balance": 0
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

`getAccountTransactionHistoryWithFilters`
This method returns an array of the transaction history details for a specified account. This method can only be called when connected to the remote Oracle Blockchain Platform network.

```
Ctx.Account.getAccountTransactionHistoryWithFilters(account_id: string,
org_id: string, filters);
```

Parameters:

- `account_id: string` – The ID of the account.
- `org_id: string` – The membership service provider (MSP) ID of the organization.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
          {
              "transaction_id":
"otransaction~ccc2c2e89ab7887af4fdad3dc9918ea057a8aa834a0ed8d0f23271049f084952
",
              "transacted_amount": 10,
              "timestamp": "2025-08-20T23:22:53.000Z",
              "token_id": "token",
              "category": "category value",
              "description": "description value",
              "holding_id": "ohold~cbdc~token~hold1",
              "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
              "to_org_id": "Org1",
              "to_user_id": "cb",
              "to_custom_account_id": "1234567jh",
              "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
              "from_org_id": "CentralBank",
              "from_user_id": "Not Available",
              "from_custom_account_id": "Not Available",
              "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
              "transaction_type": "CREDIT",
              "balance": 18,
              "onhold_balance": 0,
              "transacted_org_id": "CentralBank",
              "transacted_user_id": "Not Available",
              "transacted_custom_account_id": "Not Available"
          },
          {
              "transaction_id":
"otransaction~6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
                "transacted_amount": 22,
                "timestamp": "2025-08-20T23:21:45.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "to_account_id":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
                "to_org_id": "Org1",
                "to_user_id": "cb1",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "from_org_id": "Org1",
                "from_user_id": "cb",
                "from_custom_account_id": "1234567jh",
                "transacted_account":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
                "transaction_type": "DEBIT",
                "balance": 8,
                "onhold_balance": 0,
                "transacted_org_id": "Org1",
                "transacted_user_id": "cb1",
                "transacted_custom_account_id": "1234567jh"
            },
            {
                "transaction_id":
"otransaction~3212811cd7c0ff265434f5921b93621cc5b0e0450b07c52b42ad9d8ce73cf063
",
                "transacted_amount": 10,
                "timestamp": "2025-08-20T23:18:53.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "holding_id": "ohold~cbdc~token~hold1",
                "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_org_id": "Org1",
                "to_user_id": "cb",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "from_org_id": "CentralBank",
                "from_user_id": "Not Available",
                "from_custom_account_id": "Not Available",
                "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "transaction_type": "CREDIT",
                "balance": 30,
                "onhold_balance": 0,
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
```


Copyright © 2025, Oracle and/or its affiliates.

```
            {
                "transaction_id":
"otransaction~af2a1b0d56778e4ff5f4dda4e81bbcbf02c0592416ee2f105ff0b764e515fd0a
",
                "transacted_amount": 10,
                "timestamp": "2025-08-20T23:17:29.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "holding_id": "ohold~cbdc~token~hold1",
                "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_org_id": "Org1",
                "to_user_id": "cb",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "from_org_id": "CentralBank",
                "from_user_id": "Not Available",
                "from_custom_account_id": "Not Available",
                "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "transaction_type": "CREDIT",
                "balance": 20,
                "onhold_balance": 0,
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
            {
                "transaction_id":
"otransaction~6c0452a58f13db93f1c308d60eab1fde6c911921b350c0bbbe0ce7dab394721d
",
                "transacted_amount": 10,
                "timestamp": "2025-08-20T23:04:53.000Z",
                "token_id": "token",
                "category": "category value",
                "description": "description value",
                "holding_id": "ohold~cbdc~token~hold1",
                "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_org_id": "Org1",
                "to_user_id": "cb",
                "to_custom_account_id": "1234567jh",
                "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "from_org_id": "CentralBank",
                "from_user_id": "Not Available",
                "from_custom_account_id": "Not Available",
                "transacted_account":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
                "transaction_type": "CREDIT",
                "balance": 10,
                "onhold_balance": 0,
```


Copyright © 2025, Oracle and/or its affiliates.

```
                "transacted_org_id": "CentralBank",
                "transacted_user_id": "Not Available",
                "transacted_custom_account_id": "Not Available"
            },
            {
                "transaction_id":
"otransaction~df98e24a3b90661b54b54289b600b13a1051c922db0a4d20e3705a9820ae9c20
",
                "transacted_amount": 0,
                "timestamp": "2025-08-20T22:55:09.000Z",
                "category": "",
                "description": "",
                "from_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "from_org_id": "Org1",
                "from_user_id": "cb",
                "from_custom_account_id": "1234567jh",
                "transacted_account":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "transaction_type": "CREATE_ACCOUNT",
                "balance": 0,
                "onhold_balance": 0,
                "transacted_org_id": "Org1",
                "transacted_user_id": "cb",
                "transacted_custom_account_id": "1234567jh"
            }
        ]
```

`getAccountTrxHistoryWithFiltersFromRichHistDB`
This method returns an array of the transaction history details from the rich history database.

```
Ctx.Account.getAccountTrxHistoryWithFiltersFromRichHistDB(account_id: string,
org_id: string, custom_endpoint: string, bearer_token: string, filters);
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `org_id: string` – The membership service provider (MSP) ID of the organization.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
    {
        "transaction_id":
```


Copyright © 2025, Oracle and/or its affiliates.

```
"otransaction~3140569a4ecb3c3f141cc2468fe21276640b7fd79013d951d9104b79072d8f9c
",
        "transacted_amount": 200,
        "timestamp": "2025-08-25T13:16:55.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "EXECUTEHOLD",
        "category": "transfer",
        "balance": 26800,
        "onhold_balance": 300
    },
    {
        "transaction_id":
"otransaction~2b75b3e8531a651f07c2d048d8546ad70ac49c66f0b82ed7626c1739090842ce
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:16:06.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~e26f11da",
        "category": "transfer",
        "balance": 26800,
        "onhold_balance": 500
    },
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transaction_id":
"otransaction~9e7bf14cf96c5f90170da9455b1318687785e936192f60b7cbeb1c8bfabc41d2
",
        "transacted_amount": 100,
        "timestamp": "2025-08-21T06:57:19.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_retirer_demo",
        "transacted_custom_account_id": "10109999006543",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_retirer_demo",
        "from_custom_account_id": "10109999006543",
        "transaction_type": "CREDIT",
        "category": "transfer",
        "balance": 26900,
        "onhold_balance": 400
    },
    {
        "transaction_id":
"otransaction~b3901b4754920a9c75e36069dc55024ad505e4c127f334eedf65ef6703dc6b86
",
        "transacted_amount": 200,
        "timestamp": "2025-08-21T05:39:25.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "to_org_id": "org2",
        "to_user_id": "fi2_org_officer_demo",
        "to_custom_account_id": "30300617202404",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~77b75873",
        "category": "issuance",
        "balance": 26800,
        "onhold_balance": 400
    },
```


Copyright © 2025, Oracle and/or its affiliates.

```
    {
        "transaction_id":
"otransaction~d55c9dfc9feacb353544b5d8b2ae694162ade3890bcaaf715503fd1d6a73cd1a
",
        "transacted_amount": 200,
        "timestamp": "2025-08-21T05:39:01.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~81d7c4ac",
        "category": "transfer",
        "balance": 27000,
        "onhold_balance": 200
    },
    {
        "transaction_id":
"otransaction~751eaedbe4311edd5d17cae53d283caf397d0cb09f18d57a5e3fe61266875ff9
",
        "transacted_amount": 200,
        "timestamp": "2025-08-13T09:59:22.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "transacted_org_id": "org2",
        "transacted_user_id": "fi2_org_officer_demo",
        "transacted_custom_account_id": "30300617202404",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "from_org_id": "org2",
        "from_user_id": "fi2_org_officer_demo",
        "from_custom_account_id": "30300617202404",
        "transaction_type": "CREDIT",
        "category": "transfer",
        "balance": 27200,
        "onhold_balance": 0
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "transaction_id":
"otransaction~70155a8f4e388cc9395dbd03bedaf5a878705f5ad02302c8e9163218a5c3875a
",
        "transacted_amount": 1000,
        "timestamp": "2025-08-13T06:22:56.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "CREDIT",
        "category": "transfer",
        "balance": 27000,
        "onhold_balance": 0
    },
    {
        "transaction_id":
"otransaction~e595f3f0cc03fa5f58a546b8abbfaf155592e492f850581db2b8fed9a529c9e2
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:09:25.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "transacted_org_id": "org2",
        "transacted_user_id": "fi2_org_officer_demo",
        "transacted_custom_account_id": "30300617202404",
        "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "to_org_id": "org2",
        "to_user_id": "fi2_org_officer_demo",
        "to_custom_account_id": "30300617202404",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "EXECUTEHOLD",
        "category": "issuance",
        "balance": 26000,
        "onhold_balance": 0
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "transaction_id":
"otransaction~da92402859d87ae3069722d8e39cb0da448e9a5f67468233ee9b1fe7a4ebeef8
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:09:17.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "EXECUTEHOLD",
        "category": "issuance",
        "balance": 26000,
        "onhold_balance": 10000
    },
    {
        "transaction_id":
"otransaction~6915145aaf09fbf4d96456febddc2aa87b48c08ddd8ff17a6bab5d310f67bb36
",
        "transacted_amount": 1000,
        "timestamp": "2025-08-12T21:07:11.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_retirer_demo",
        "transacted_custom_account_id": "10109999006543",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_retirer_demo",
        "from_custom_account_id": "10109999006543",
        "transaction_type": "CREDIT",
        "category": "transfer",
        "balance": 26000,
        "onhold_balance": 20000
```


Copyright © 2025, Oracle and/or its affiliates.

```
    },
    {
        "transaction_id":
"otransaction~244d7172d1dc90a142e1f22204c76614c7eea814b3d61f33016b786f1b347784
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:05:39.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "to_org_id": "org2",
        "to_user_id": "fi2_org_officer_demo",
        "to_custom_account_id": "30300617202404",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~ed815e20",
        "category": "issuance",
        "balance": 25000,
        "onhold_balance": 20000
    },
    {
        "transaction_id":
"otransaction~c63ec37966264493bde6fa666527b9cca11695c15611c32e89af49a2246f13f6
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-12T21:05:20.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "from_user_id": "cb_issuer_demo",
        "from_custom_account_id": "10109999001234",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~12d87129",
        "category": "issuance",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "balance": 35000,
            "onhold_balance": 10000
        },
        {
            "transaction_id":
"otransaction~5112f576c94c2d23c342479bfa37e34612414b3258a64b43cf51b920f4ff5868
",
            "transacted_amount": 5000,
            "timestamp": "2025-08-12T21:05:02.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb_retirer_demo",
            "transacted_custom_account_id": "10109999006543",
            "to_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
            "to_org_id": "CentralBank",
            "to_user_id": "cb_retirer_demo",
            "to_custom_account_id": "10109999006543",
            "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "from_org_id": "CentralBank",
            "from_user_id": "cb_issuer_demo",
            "from_custom_account_id": "10109999001234",
            "transaction_type": "DEBIT",
            "category": "burn",
            "balance": 45000,
            "onhold_balance": 0
        },
        {
            "transaction_id":
"otransaction~862aa9d9e877d3ea209b87299ab5b12c13ed5ce43d1cf1b934043c1dd02f58f6
",
            "transacted_amount": 50000,
            "timestamp": "2025-08-12T21:04:22.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb__creator_demo",
            "transacted_custom_account_id": "10105678004567",
            "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "to_org_id": "CentralBank",
            "to_user_id": "cb_issuer_demo",
            "to_custom_account_id": "10109999001234",
            "from_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "from_org_id": "CentralBank",
            "from_user_id": "cb__creator_demo",
            "from_custom_account_id": "10105678004567",
            "transaction_type": "CREDIT",
            "category": "transfer",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "balance": 50000,
        "onhold_balance": 0
    }
]
```

getOrgAccountsTrxHistoryWithFiltersFromRichHistDB
This method returns an array of the transaction history details for a specified organization from the rich history database.

```
Ctx.Account.getOrgAccountsTrxHistoryWithFiltersFromRichHistDB(org_id: string,
custom_endpoint: string, bearer_token: string, filters);
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the organization.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
    {
        "transaction_id":
"otransaction~4793f3907eefce2f9fca7ef107405b0f116efb3afbf83fa0e61fe763690c8235
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:47:56.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~2ac01689",
        "category": "issuance"
    },
    {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transaction_id":
"otransaction~5177f7560d32838242a26ac74f2a90c6ff9b47aae0d0988f28d9b4cf7e27c097
",
        "transacted_amount": 10,
        "timestamp": "2025-08-25T13:22:23.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user1_demo",
        "transacted_custom_account_id": "20200198765432",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
        "from_custom_account_id": "20200198765432",
        "transaction_type": "CREDIT",
        "holding_id": "ohold~cbdc~USD~454f4bf6",
        "category": "transfer"
    },
    {
        "transaction_id":
"otransaction~5177f7560d32838242a26ac74f2a90c6ff9b47aae0d0988f28d9b4cf7e27c097
",
        "transacted_amount": 10,
        "timestamp": "2025-08-25T13:22:23.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user2_demo",
        "transacted_custom_account_id": "20200211112222",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
        "from_custom_account_id": "20200198765432",
        "transaction_type": "EXECUTEHOLD",
        "holding_id": "ohold~cbdc~USD~454f4bf6",
        "category": "transfer"
    },
    {
        "transaction_id":
"otransaction~32137cd7e16c560f4d96d0f8999f6a267c1e95be2f5b11ef0541e574a6cd7275
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
        "transacted_amount": 200,
        "timestamp": "2025-08-25T13:16:55.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "to_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_officer_demo",
        "to_custom_account_id": "20200222221111",
        "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "from_org_id": "CentralBank",
        "transaction_type": "CREDIT",
        "category": "transfer"
    },
    {
        "transaction_id":
"otransaction~67844e07c06ab4c3d94d8ff36173b9dbcc2f1b9c99f4e7070f1097fd89f3f514
",
        "transacted_amount": 10,
        "timestamp": "2025-08-21T08:49:40.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user1_demo",
        "transacted_custom_account_id": "20200198765432",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
        "from_custom_account_id": "20200198765432",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~b6e4925b",
        "category": "transfer"
    },
    {
        "transaction_id":
"otransaction~b7c97d737fb978651c9132276ab677c0bcf795b9db13d0d39cba5243615c7389
",
        "transacted_amount": 10,
        "timestamp": "2025-08-21T08:46:09.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user1_demo",
        "to_custom_account_id": "20200198765432",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~4fbbb846",
        "category": "transfer"
    }
]
```

`getAllAccountsTrxHistoryWithFiltersFromRichHistDB`
This method returns an array of the transaction history details for all organizations from the rich history database.

```
Ctx.Account.getAllAccountsTrxHistoryWithFiltersFromRichHistDB(custom_endpoint:
 string, bearer_token: string, filters);
```

Parameters:

- `org_id string` – The membership service provider (MSP) ID of the organization.
- `custom_endpoint` – The RESTful service endpoint of the rich history database.
- `bearer_token` – The access authorization token for the RESTful service endpoint.
- `filters: string` – An optional parameter. If empty, all records are returned. The `PageSize` property determines the number of records to return. If `PageSize` is 0, the default page size is 20. The `Bookmark` property determines the starting index of the records to return. For more information, see the Hyperledger Fabric documentation. The `StartTime` and `EndTime` properties must be specified in RFC-3339 format.

Return Value Example:

```
[
    {
        "transaction_id":
"otransaction~62eb436be7c29fc2ed9cae221e874d9a31b163fa10374e7da09bf5e09a96c3ff
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-25T13:57:58.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "from_org_id": "CentralBank",
        "from_user_id": "cb__creator_demo",
        "from_custom_account_id": "10105678004567",
        "transaction_type": "DEBIT",
        "category": "issuance"
    },
    {
        "transaction_id":
"otransaction~62eb436be7c29fc2ed9cae221e874d9a31b163fa10374e7da09bf5e09a96c3ff
",
        "transacted_amount": 10000,
        "timestamp": "2025-08-25T13:57:58.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb__creator_demo",
        "transacted_custom_account_id": "10105678004567",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
        "from_org_id": "CentralBank",
        "from_user_id": "cb__creator_demo",
        "from_custom_account_id": "10105678004567",
        "transaction_type": "CREDIT",
        "category": "issuance"
    },
    {
        "transaction_id":
"otransaction~c628fb7738222ed969295ccc8d21b4be95d96e3aada4f14570f7820a7051b5f7
",
        "transacted_amount": 200,
        "timestamp": "2025-08-25T13:51:18.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "transacted_org_id": "CentralBank",
        "transacted_user_id": "cb_issuer_demo",
        "transacted_custom_account_id": "10109999001234",
        "to_account":
"oaccount~3954f54a8bc7acdd0c3d0960104240f60d56c26c8a179430267359cd80ce3709",
        "to_org_id": "org2",
        "to_user_id": "fi2_org_officer_demo",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "to_custom_account_id": "30300617202404",
            "from_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
            "from_org_id": "CentralBank",
            "from_user_id": "cb_issuer_demo",
            "from_custom_account_id": "10109999001234",
            "transaction_type": "RELEASEHOLD",
            "holding_id": "ohold~cbdc~USD~77b75873"
        },
        {
            "transaction_id":
"otransaction~af2cc8e43ecb4c5520d90a8d7955b5a47623a29b13eef47e31c16eb48cc0adec
",
            "transacted_amount": 100,
            "timestamp": "2025-08-25T13:50:45.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb_retirer_demo",
            "transacted_custom_account_id": "10109999006543",
            "from_account":
"oaccount~cea6080858337b1575d6a76ed0bd07a0eacd8871e3f2f7f793729a0e4b0e8e98",
            "from_org_id": "CentralBank",
            "from_user_id": "cb_retirer_demo",
            "from_custom_account_id": "10109999006543",
            "transaction_type": "REJECT_BURN",
            "holding_id": "ohold~cbdc~USD~8d34",
            "description": "Burn",
            "to_account": ""
        },
        {
            "transaction_id":
"otransaction~182b99bb2ed753994a8c638ab9b08c3a4e73ac8159a3173a2a1f56b651d2eeac
",
            "transacted_amount": 2000,
            "timestamp": "2025-08-25T13:50:19.000Z",
            "token_id": "USD",
            "transacted_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "transacted_org_id": "CentralBank",
            "transacted_user_id": "cb__creator_demo",
            "transacted_custom_account_id": "10105678004567",
            "to_account":
"oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41",
            "to_org_id": "CentralBank",
            "to_user_id": "cb__creator_demo",
            "to_custom_account_id": "10105678004567",
            "transaction_type": "REJECT_MINT",
            "holding_id": "ohold~cbdc~USD~89ce",
            "description": "Minting 2000 tokens",
            "from_account": ""
        },
        {
```


Copyright © 2025, Oracle and/or its affiliates.

```
        "transaction_id":
"otransaction~4793f3907eefce2f9fca7ef107405b0f116efb3afbf83fa0e61fe763690c8235
",
        "transacted_amount": 100,
        "timestamp": "2025-08-25T13:47:56.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_officer_demo",
        "transacted_custom_account_id": "20200222221111",
        "to_account":
"oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a",
        "to_org_id": "CentralBank",
        "to_user_id": "cb_issuer_demo",
        "to_custom_account_id": "10109999001234",
        "from_account":
"oaccount~76687c724ddbc2d6e6664d9618b2bf1c2a9fe10f84887462447e4caba6aaaff3",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_officer_demo",
        "from_custom_account_id": "20200222221111",
        "transaction_type": "ONHOLD",
        "holding_id": "ohold~cbdc~USD~2ac01689",
        "category": "issuance"
    },
    {
        "transaction_id":
"otransaction~5177f7560d32838242a26ac74f2a90c6ff9b47aae0d0988f28d9b4cf7e27c097
",
        "transacted_amount": 10,
        "timestamp": "2025-08-25T13:22:23.000Z",
        "token_id": "USD",
        "transacted_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "transacted_org_id": "Org1",
        "transacted_user_id": "fi1_org_user2_demo",
        "transacted_custom_account_id": "20200211112222",
        "to_account":
"oaccount~d08ff55a040d5e5dcf406009bab1b3398e06c374356efb1bddbf2f17fc37f949",
        "to_org_id": "Org1",
        "to_user_id": "fi1_org_user2_demo",
        "to_custom_account_id": "20200211112222",
        "from_account":
"oaccount~1e31495a0c149b08cb9d02bdcac5e83d88c0f1557d954dda12bb807d7f6fc111",
        "from_org_id": "Org1",
        "from_user_id": "fi1_org_user1_demo",
        "from_custom_account_id": "20200198765432",
        "transaction_type": "EXECUTEHOLD",
        "holding_id": "ohold~cbdc~USD~454f4bf6",
        "category": "transfer"
    }
]
```


Copyright © 2025, Oracle and/or its affiliates.

`getAccountBalance`
This method returns the account balance for a specified account.

```
Ctx.Account.getAccountBalance(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
    "msg": "Current Balance is: 100",
    "user_balance": 100
}
```

`deleteTransactions`
This method deletes older transactions from the state database.

```
Ctx.Transaction.deleteTransactions(time_to_expiration: Date);
```

Parameters:

- `time_to_expiration: Date` – A time stamp that indicates when to delete transactions. Transaction assets that are older than the specified time will be deleted..

Return Value Example:

```
{
    "msg": "Successfully deleted transaction older than date: Thu Aug 19 2025
11:19:36 GMT+0000 (Coordinated Universal Time).",
    "transactions": [

"otransaction~ec3366dd48b4ce2838f820f2f138648e6e55a07226713e59b411ff31b0d21058
"
     ]
}
```

`getTransactionById`
This method returns the history of a `Transaction` asset.

```
Ctx.Transaction.getTransactionById(transaction_id: string);
```

Parameters:

- `transaction_id string` – The ID of the transaction asset.

Return Value Example:

```
{
    "transaction_id":
"otransaction~6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a
```


Copyright © 2025, Oracle and/or its affiliates.

```
",
    "history": [
        {
            "trxId":
"6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a",
            "timeStamp": "2025-08-20T23:21:45.000Z",
            "value": {
                "assetType": "otransaction",
                "transaction_id":
"otransaction~6523597253e45b3c976aecdc54e2c6316d0a4f88ad5d7f3fff48c69213e4375a
",
                "token_id": "token",
                "from_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
                "to_account_id":
"oaccount~5e3b5a3306d7ca3f3e3fe7cc27b6ae547e94dba1c5af58a69a771d7a619b6a66",
                "transaction_type": "TRANSFER",
                "amount": 22,
                "timestamp": "2025-08-20T23:21:45.000Z",
                "number_of_sub_transactions": 0,
                "holding_id": "",
                "sub_transaction": "false",
                "category": "category value",
                "description": "description value"
            }
        }
    ],
    "sub_transactions": []
}
```

`getAccountStatus`
This method gets the current status of the token account.

```
Ctx.AccountStatus.getAccountStatus(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "active"
}
```


Copyright © 2025, Oracle and/or its affiliates.

history (Account Status)
This method gets the history of the account status.

```
Ctx.AccountStatus.history(status_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
[
  {
    "trxId":
"d5c6d6f601257ba9b6edaf5b7660f00adc13c37d5321b8f7d3a35afab2e93e63",
    "timeStamp": "2025-12-02T10:39:14.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "suspended"
    }
  },
  {
    "trxId":
"e6c850cfa084dc20ad95fb2bb8165eef3a3bd62a0ac867cccee57c2003125183",
    "timeStamp": "2025-12-02T10:37:50.000Z",
    "value": {
      "assetType": "oaccountStatus",
      "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
      "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
      "status": "active"
    }
  }
]
```

activateAccount
This method activates a token account. Deleted accounts cannot be activated.

```
Ctx.Account.activateAccount(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "active"
}
```

suspendAccount
This method suspends a token account. After an account is suspended, you cannot complete any operations that update the account. A deleted account cannot be suspended.

```
Ctx.Account.suspendAccount(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
    "assetType": "oaccountStatus",
    "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
d7",
    "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
    "status": "suspended"
}
```

deleteAccount
This method deletes a token account. After an account is deleted, you cannot complete any operations that update the account. The deleted account is in a final state and cannot be changed to any other state. To delete an account, the account balance and the on-hold balance must be zero.

```
Ctx.Account.deleteAccount(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
  "assetType": "oaccountStatus",
  "status_id":
"oaccountStatus~5a0b0d8b1c6433af9fedfe0d9e6580e7cf6b6bb62a0de6267aaf79f79d5e96
```


Copyright © 2025, Oracle and/or its affiliates.

```
d7",
  "account_id":
"oaccount~1c568151c4acbcd1bd265c766c677145760a61c47fc8a3ba681a4cfbe287f9c1",
  "status": "deleted"
}
```

setMaxDailyAmount
This method sets the `max_daily_amount` value for a specified account.

```
Ctx.Account.setMaxDailyAmount(account_id: string, max_daily_amount?: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `max_daily_amount?: string` – (Optional) The maximum amount of tokens the user can transfer daily. If not specified, the user can transfer any amount of tokens daily.

Return Value Example:

```
{
           "msg": "Successfully set max daily amount for account id
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a to
1000000"
       }
```

setMaxDailyTransactionCount
This method sets the `max_daily_transactions` value for a specified account.

```
Ctx.Account.setMaxDailyTransactionCount(account_id: string,
max_daily_transactions?: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.
- `max_daily_transactions?: string` – (Optional) The maximum number of transactions the user can complete daily. If not specified, the user can complete any number of transactions daily.

Return Value Example:

```
{
            "msg": "Successfully set max daily transactions for account id
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a to
100000"
        }
```


Copyright © 2025, Oracle and/or its affiliates.

getMaxDailyAmount
This method gets the `max_daily_amount` value (the maximum amount that can be transferred daily) for a specified account.

```
Ctx.Account.getMaxDailyAmount(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
            "max_daily_amount": "10000"
       }
```

getMaxDailyTransactionCount
This method gets the `max_daily_transactions` value (the maximum number of transactions allowed per day) for a specified account.

```
Ctx.Account.getMaxDailyTransactionCount(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
            "max_daily_transactions": "100"
       }
```

consolidateRunningBalanceInTransactions
This method calculates the running account balances saves the updated values in the transaction key/value pairs. Typically the system administrators call this method by using a REST proxy scheduler.

```
Ctx.Transaction.consolidateRunningBalanceInTransactions(save: boolean);
```

Return Value Example:

```
{ msg: "Successfully updated account running balance for pending
transactions."}
```

processSendersAndReceivers
This method calculates and update the token account balances that are spread in the sender and receiver key/value pairs that were created during inter-organization transfers, and then

Copyright © 2025, Oracle and/or its affiliates.

deletes the sender and receiver key/value pairs, which are no longer of use. Typically the system administrators call this method by using a REST proxy scheduler.

```
Ctx.Account.processSendersAndReceivers(save: boolean);
```

Return Value Example:

```
{ msg: "Successfully updated balance for accounts from pending receivers."}
```

addRoleMember
This method adds a role to a specified account.

```
Ctx.Token.addRoleMember(role: string, account_id: string, token_asset);
```

Parameters:

- `account_id: number` – The account ID to add the role to.
- `role: string` – The name of the role to add to the specified user.
- `token_asset: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.

Return Value Example:

```
{
     "msg": "Successfully added role 'notary' to Account Id:
oaccount~2eb5f8a9bc561f8f41a4ea3be9511958cc6684ef14f2337ca396efc301b627d8"
}
```

removeRoleMember
This method removes a role from a specified account.

```
Ctx.Token.removeRoleMember(role: string, account_id: string, token_asset);
```

Parameters:

- `account_id: number` – The account ID to remove the role from.
- `role: string` – The name of the role to remove from to the specified user.
- `token_asset: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.

Return Value Example:

```
{
   "msg": "Successfully removed role 'notary' from Account Id:
oaccount~2eb5f8a9bc561f8f41a4ea3be9511958cc6684ef14f2337ca396efc301b627d8"
}
```


Copyright © 2025, Oracle and/or its affiliates.

getAccountsByRole
This method returns a list of all accounts for a specified role and token.

```
Ctx.Role.getAccountsByRole(token_id: string, role: string);
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Return Value Example:

```
{
          "accounts": [

"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75"
          ]
      }
```

getOrgAccountsByRole
This method returns information about all accounts that have a specified role in a specified organization.

```
Ctx.Role.getOrgAccountsByRole(token_id: string, role: string, org_id: string);
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Return Value Example:

```
{
          "accounts": [

"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75"
          ]
      }
```

getUsersByRole
This method returns a list of all users for a specified role and token.

```
Ctx.Role.getUsersByRole(token_id: string, role: string);
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to search for.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
          "users": [
              {
                  "token_id": "token",
                  "org_id": "CentralBank"
              }
          ]
      }
```

`getOrgUsersByRole`
This method returns information about all users that have a specified role in a specified organization.

```
 Ctx.Role.getOrgUsersByRole(token_id: string, role: string, org_id: string);
```

Parameters:

- `token_id: string` – The ID of the token.
- `role: string` – The name of the role to check for.
- `org_id: string` – The membership service provider (MSP) ID of the organization.

Return Value Example:

```
{
           "users": [
               {
                   "token_id": "token",
                   "org_id": "CentralBank",
                   "user_id": "cb1",
                   "custom_account_id": "1234567jh"
               }
           ]
       }
```

`isInRole`
This method indicates whether a user has a specified role.

```
Ctx.Token.isInRole(role: string, account_id: string, token_asset)
```

Parameters:

- `account_id: number` – The unique ID of the token account.
- `role: string` – The name of the role to check.
- `token_asset: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
            "result": "true"
       }
```

getTotalMintedTokens
This method returns the total number of tokens minted.

```
Ctx.Token.getTotalMintedTokens(token_asset);
```

Parameters:

- `token_asset: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.

Return Value Example:

```
{
    "msg": "Total minted token for Token Id: USD is 910 tokens.",
    "quantity": 910
}
```

getNetTokens
This method returns the net quantity of tokens that are available in the system. The net tokens are the amount of tokens remaining after tokens are burned. In equation form: net tokens = total minted tokens - total burned tokens. If no tokens are burned, then the number of net tokens is equal to the total minted tokens.

```
Ctx.Token.getNetTokens(token_asset);
```

Parameters:

- `token_asset: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.

Return Value Example:

```
{
    "msg": "Net supply of token for Token Id: USD is 878 tokens.",
    "quantity": 878
}
```

hold
This method is used by the minter to send a request to the minter notary to create a specified amount of tokens. The specified notary account is responsible for either completing or releasing the hold.

```
Ctx.Hold.hold(operation_id: string, to_account_id: string, notary_account_id:
string, quantity: number, time_to_expiration: Date, token, type:
HoldOperationType, info_details?: InfoDetails)
```


Copyright © 2025, Oracle and/or its affiliates.

Parameters:

- `operation_id: string` – A unique ID to identify the hold operation. Typically this ID is passed by the client application.
- `to_account_id: string` – The ID of the recipient account.
- `notary__account_id: string` – The ID of the notary account.
- `quantity: number` – The total number of tokens to put on hold.
- `time_to_expiration: Date` – The duration until the hold expires. Specify `0` for a permanent hold. Otherwise use the RFC-3339 format. For example, `2021-06-02T12`.
- `token: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.
- `type: HoldOperationType` – The type of hold operation to request. It must be one of three possible values, `MINT`, `BURN`, or `TRANSFER`.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
           "msg": "AccountId
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41 has
successfully submitted request to mint 200 tokens"
       }
```

`executeHold`
Notaries can call this method to approve a hold operation, which can be one of three types: mint, burn, or transfer.

```
Ctx.Hold. executeHold(operation_id: string, token, operation_type:
HoldOperationType, quantity? :number)
```

Parameters:

- `operation_type: HoldOperationType` – The type of hold operation to approve. It must be one of three possible values, `MINT`, `BURN`, or `TRANSFER`.
- `quantity: number` – (Optional) For transfer operations only, the amount that is approved to be transferred to the recipient.
- `token: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.
- `operation_id: string` – A unique ID of the request.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
           "msg": "Successfully minted 1000 tokens to Account Id:
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41"
       }
```

`releaseHold`
Notaries can call this method to reject a hold operation, which can be one of three types: mint, burn, or transfer.

```
Ctx.Hold.releaseHold(operation_id: string, token, operation_type:
HoldOperationType)
```

Parameters:

- `operation_type: HoldOperationType` – The type of hold operation to reject. It must be one of three possible values, `MINT`, `BURN`, or `TRANSFER`.
- `token: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.
- `operation_id: string` – A unique ID of the request.

Return Value Example:

```
{
           "msg": "Successfully rejected mint request with Operation Id
'89ce' to mint 2000 tokens of token id USD"
       }
```

`issueTokens`
Minters can call this method to create a specified amount of tokens.

```
this.Ctx.Token.mint(quantity, token_asset, info_details)
```

Parameters:

- `token_id: string` – The ID of the token.
- `quantity` – The number of tokens to mint.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
          "msg": "Successfully minted 1000 tokens to Account Id:
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41"
      }
```

`transferTokens`
This method transfers tokens from the caller to a specified account. The caller of the method must have an account. The quantity must be within the decimal values specified by the `decimal` parameter of the `divisible` behavior in the specification file.

```
this.Ctx.Token.transfer(to_account_id: string, quantity, token_asset,
info_details);
```

Parameters:

- `to_account_id: string` – The account ID of the receiver (payee).
- `quantity: number` – The number of tokens to transfer.
- `info_details: JSON` – An object specifying the category (`category`) and description (`description`) of the request.

  You specify the `info_details` parameter in a different format if you are using Visual Studio Code versus the CLI or a Postman collection.

  Visual Studio Code: `{ "category": "category value", "description": "description value" }`

  CLI / Postman: `"{\"category\":\"category value\",\"description\":\"description value\"}"`

Return Value Example:

```
{
          "msg": "Successfully transferred 10000 tokens from account id:
oaccount~da6e14466a0ba9b48ebc18fa672addb92dffc371bf953c3229a95b2ff2d9cd41  to
account id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a."
      }
```

`getAccountOnHoldBalance`
This method returns the on-hold balance for a specified account.

```
Ctx.Account.getAccountOnHoldBalance(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
           "msg": "Total Holding Balance is: 0",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "holding_balance": 0
        }
```

getOnHoldDetailsWithOperationId
This method returns the on-hold transaction details for a specified operation ID and token.

```
Ctx.Hold.getOnHoldDetailsWithOperationId(token_id: string, operation_id:
string);
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID that identifies the hold operation.

Return Value Example:

```
{
          "assetType": "ohold",
          "holding_id": "ohold~cbdc~token~hold1",
          "operation_id": "hold1",
          "token_name": "cbdc",
          "from_org_id": "CentralBank",
          "operation_type": "transfer",
          "status": "approved",
          "from_account_id":
"oaccount~5abb4155f4b245bb1732321e3174ac81999b03495b5c74f8e1f270bafbadef75",
          "to_account_id":
"oaccount~22776ada3efff6aaf4c68c204c1f063b139adfa1bca79ed53199117c34036cfc",
          "notary_account_id":
"oaccount~1da238c1491c919b151f777a615fb5779032c34b3ef3adeca6afe591bac10aaf",
          "token_id": "token",
          "quantity": "0",
          "time_to_expiration": "2029-01-04T00:00:00.000Z",
          "category": "category value",
          "description": "description value"
      }
```

getOnHoldIds
This method returns a list of all of the holding IDs for a specified account.

```
Ctx.Account.getOnHoldIds(account_id: string);
```

Parameters:

- `account_id: string` – The unique ID of the token account.

Return Value Example:

```
{
           "msg": "Holding Ids are: ",
```


Copyright © 2025, Oracle and/or its affiliates.

```
            "holding_ids": ["ohold~cbdc~token~hold1"]
        }
```

`getOnHoldBalanceWithOperationId`
This method returns the on-hold balance for a specified operation ID and token. This method can be called by anyone.

```
Ctx.Hold.getOnHoldBalanceWithOperationId(token_id: string, operation_id:
string);
```

Parameters:

- `token_id: string` – The ID of the token.
- `operation_id: string` – A unique ID that identifies the hold operation.

Return Value Example:

```
{
            "msg": "Current Holding Balance of Operation 'hold1' for token
'token' is: 0",
            "holding_balance": 0
        }
```

TypeScript SDK Methods for Inter-Organizational Transfers

Inter-organization transfers have two parts: first the amount is debited from the sender and then the amount is credited to the receiver. The following two APIs are called together as an atomic transaction using the two-phase commit API of the Oracle Blockchain Platform REST proxy. Confidential transfers are done in this manner because the private data collections of the two organizations are distinct and cannot be accessed in a single transaction.

`executeHoldTokensSender`
This method can be called only by the user with the notary role for the specified operation ID. This method is the first part of an approval of an transfer between organizations. The specified amount is deducted from the sender's account.

```
Ctx.Hold.executeHoldSender(operation_id: string, quantity, globalTxId:
string, token: any);
```

Parameters:

- `token: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.
- `operation_id: string` – The unique operation ID that represents the hold request to approve.
- `quantity: number` – The amount of held tokens to transfer.
- `globalTxid: string` – The global transactin ID generated by the REST proxy for a two-phase commit request, which in this case is used as the common reference that binds the debit and credit operations in an inter-organization transfer.

Copyright © 2025, Oracle and/or its affiliates.

Return Value Example:

```
{
      "msg":
        "Account Id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a is
successfully executed '10' tokens from Operation Id 'hold1'."
    }
```

`executeHoldTokensReceiver`
This method can be called only by the user with the notary role for the specified operation ID. This method is the second part of an approval of an transfer between organizations. The specified amount is credited to the receiver's account.

```
Ctx.Hold.executeHoldReceiver(operation_id: string, quantity, globalTxId:
string, token: any);
```

Parameters:

- `token: any` – The token asset, passed as a parameter. The properties of the token asset are described in the model file.
- `operation_id: string` – The unique operation ID that represents the hold request to approve.
- `quantity: number` – The amount of held tokens to transfer.
- `globalTxid: string` – The global transactin ID generated by the REST proxy for a two-phase commit request, which in this case is used as the common reference that binds the debit and credit operations in an inter-organization transfer.

Return Value Example:

```
{
      "msg":
        "Account Id:
oaccount~68d67712f500e9dac8c314c19744003a993250271d960e9b0d25267bb18dfe9a is
successfully executed '10' tokens from Operation Id 'hold1, receiverId
oreceiver~72d9bfcf-2c68-4c33-b8c3-fe3374983bf2'."
    }
```

Application Models

The enhanced version of Blockchain App Builder supports model attributes that generate additional methods for application scenarios.

If you include the `model: wcbdc` parameter in the specification file for tokens that use the extended Token Taxonomy Framework standard, Blockchain App Builder generates application-specific chaincode including additional methods and functionality for use with the wholesale CBDC application. If you also include `confidential: true` parameter, Blockchain App Builder generates the confidential mode version of the application-specific chaincode (TypeScript only).

If you include the `model: bond` parameter in the specification file for tokens that use the extended ERC-1155 standard, Blockchain App Builder generates application-specific chaincode, including additional methods for use with the bond marketplace application.

Copyright © 2025, Oracle and/or its affiliates.

For more information, see the Oracle Blockchain Platform Digital Assets Edition guide.

Enhancements to Postman Collections

Oracle Blockchain Platform Digital Assets Edition adds support for endorsement parameters, confidential chaincodes, and OAuth 2.0 to the Postman collections generated by Blockchain App Builder.

Blockchain App Builder enables you to create a Postman collection that includes example payloads for all of your chaincode controller APIs. For more information, see Generate a Postman Collection Using the CLI and Generate a Postman Collection Using Visual Studio Code.

Oracle Blockchain Platform Digital Assets Edition extends that functionality by including an additional parameter in the request payload for all setter methods. For generic (as opposed to confidential mode) chaincode, the parameter is either `endorsers` or `sameOrgEndorser`. The `sameOrgEndorser` parameter, if `true`, indicates that transaction endorsements must be from the same organization as the requester. The `endorsers` parameter specifies a list of peers that must endorse the transaction.

The `sameOrgEndorserOptionInWrapperAPI` parameter in the `.ochain.json` file in the chaincode specifies which APIs require a `sameOrgEndorser` value. APIs that are associated with the `sameOrgEndorserOptionInWrapperAPI` parameter have the `sameOrgEndorser` parameter set to `true` in their payloads. All other APIs include the `endorsers` parameter instead of the `sameOrgEndorser` parameter.

The following snippet shows the `sameOrgEndorserOptionInWrapperAPI` parameter in the `.ochain.json` file in the wholesale CBDC chaincode package.

```
"sameOrgEndorserOptionInWrapperAPI":
["addConversionRate","addTokenAdmin","addTokenAuditor","approveBurn","approveM
int","burnTokens","createExchangePoolAccounts","deleteHistoricalTransactions",
"initializeCBDCToken","initializeExchangePoolUser","issueTokens","mintWithFund
ingExchangePool","rejectBurn","rejectMint","removeTokenAdmin","removeTokenAudi
tor","requestBurn","requestMint","updateCBDCToken","updateConversionRate"]
```

You can customize this parameter as needed. When the wrapper API is generated, the specified APIs will have the `sameOrgEndorser` parameter set to `true` in their payloads.

The following example payloads show these endorsement parameters.

`addOrgAdmin`

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "addOrgAdmin",
        "{{bc-org-id}}",
        "{{bc-user-id}}"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}},
    "endorsers": {{endorsers}}
}
```


Copyright © 2025, Oracle and/or its affiliates.

addTokenAdmin

```
{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "addTokenAdmin",
        "{{bc-org-id}}",
        "{{bc-user-id}}"
    ],
    "timeout": {{bc-timeout}},
    "sync": {{bc-sync}},
    "sameOrgEndorser": true
}
```

Confidential Chaincode Support

Chaincodes generated in confidential mode handle parameters differently from generic mode. In confidential chaincodes, API arguments are passed by using a transient map in the request payload. This ensures that arguments remain confidential and are not exposed in logs. For more information about confidential chaincode, see Confidential Payments Overview.

All getter methods include a `peer` parameter (not present in generic mode chaincode).

All setter methods include the `endorsers` parameter, which specifies a list of peers that must endorse the transaction. The `sameOrgEndorserOptionInWrapperAPI` parameter in the `.ochain.json` file is set to an empty array:

```
"sameOrgEndorserOptionInWrapperAPI": []
```

Setter methods require a `Confidential-Transaction` header, which indicates that the blinding factor required for confidential transactions is retrieved from the Oracle Blockchain Platform instance. This header is required in Postman requests for confidential transactions.

To maintain the security of confidential mode chaincode, pass arguments in a `transientMap` object. You can pass arguments directly in the `args` parameter in the transient map, as shown in the following example for the `activateAccount` method:

```
"transientMap": { "args": "[\"account_id value\"]" }
```

For the wrapper API Postman collection only (not the generic Postman collection), you can also use the `transientMapArgsFlag` flag, which if set to `true` automatically passes all arguments to the transient map, as shown in the following example for the `activateAccount` method:

```
{
   "accountId": "account_id value",
   "transientMapArgsFlag": true
}
```

The following example Postman requests show these parameters and headers.

Copyright © 2025, Oracle and/or its affiliates.

`createAccount` (setter method)

```
curl --location 'https://test-xyz-abc.blockchain.ocp.oraclecloud.com:7443/
restproxy/api/v2/channels/default/transactions' \
--header 'Content-Type: application/json' \
--header 'Confidential-Transaction: true' \
--header 'Authorization: Bearer {{access_token}}' \
--data '{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "activateAccount"
    ],
    "timeout": 6000,
    "sync": true,
    "endorsers": ["org1-xyz-abc.blockchain.ocp.oraclecloud.com:20009", "org2-
xyz-abc.blockchain.ocp.oraclecloud.com:20009"],
    "transientMap": {
        "args": "[\"account_id value\"]"
    }
}'
```

`getAllActiveAccounts` (getter method)

```
curl --location 'https://test-xyz-abc.blockchain.ocp.oraclecloud.com:7443/
restproxy/api/v2/channels/default/chaincode-queries' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer {{access_token}}' \
--data '{
    "chaincode": "{{bc-chaincode-name}}",
    "args": [
        "getAllActiveAccounts"
    ],
    "timeout": 6000,
    "sync": true,
    "peer": "org-xyz-abc.blockchain.ocp.oraclecloud.com:20009",
    "transientMap": {
        "args": "[\"bc-org-id value\",\"bc-token-id value\"]"
    }
}'
```

OAuth 2.0 Support

By default, the Postman collections generated by Blockchain App Builder are configured for OAuth 2.0 authorization. You must provide the following values in the global variables of the Postman collection to enable OAuth 2.0.

`client-id`
The client ID of the Oracle Blockchain Platform instance.

`client-secret`
The client secret of the Oracle Blockchain Platform instance.

`access-token-url`
The access token URL contains the domain URL of the Oracle Blockchain Platform instance, in the following format: `https://<domain-URL>/oauth2/v1/token`

Copyright © 2025, Oracle and/or its affiliates.

Complete the following steps to retrieve the client ID, client secret, and domain URL for an Oracle Blockchain Platform instance:

1. Sign in to your Oracle Cloud Infrastructure (OCI) account. Select the compartment that contains the Oracle Blockchain Platform instance.
2. In the console, click the **Navigation** menu, and then click **Identity & Security**.
3. Under **Identity**, select **Domains**.
4. On the Domains page, click **Oracle Identity Cloud Service**.
5. In the Oracle Identity Cloud Service navigation menu, select **Oracle Cloud Services**. Locate your Oracle Blockchain Platform instance and open that instance's details page.
6. The default page displayed is the OAuth Configuration page. The Client ID and Client Secret fields are in the General Information section.
7. Navigate back to the Domains page and then click the domain that is used by your Oracle Blockchain Platform instance.
8. The domain URL is displayed in the domain details.

These global variables are used in a pre-request script to generate an `access_token` placeholder. The script generates a token for the `bc-admin` user, so that every request runs with `bc-admin` authorization.

The first request generates a `bc-admin` token using the specified credentials. The token and the expiry value are stores in Postman collection variables. Missing or expired tokens are automatically regenerated. If the client ID or client secret changes, the script automatically detects the change, invalidates the previous token and generates a token with updated credentials. If you modify the `access_token` value manually, it can cause requests to fail. If you clear the `access_token` value, the script generates another `bc-admin` token automatically.

You can use the `Generate Access Token for User` request in the Postman collection to generate and use a token other than the default `bc-admin` token. To generate a custom access token, complete the following steps.

1. Open the `Generate Access Token for User` request. By default, the `username` and `password` values are set to `bc-admin`.
2. Update the `username` and `password` values in the collection variables.
3. Click **Generate New Access Token**. The Manage Access Tokens window opens.
4. Update the default name, review the other details, and then click **Save**. The token is now available for use, and shown in the Available Tokens section on the Authorization page.

Generated Postman collections include variables required for OAuth 2.0 support, as shown in the following table.

Variable	Description	Default Value
`client-id`	Client ID of the Oracle Blockchain Platform instance.	`client-id` value
`client-secret`	Client Secret of the Oracle Blockchain Platform instance.	`client-secret` value
`access-token-url`	OAuth 2.0 token endpoint of the domain of the Oracle Blockchain Platform instance.	`access-token-url` value
`access_token`	OAuth 2.0 access token generated for the `bc-admin` or custom user.	Not available initially; managed by script

Copyright © 2025, Oracle and/or its affiliates.

Variable	Description	Default Value
`access_token_expiry`	Expiry time of the generated access token.	Not available initially; managed by script
`admin_credentials_hash`	Hash value of credentials used for token validation and regeneration.	Not available initially; managed by script

Auditor Roles

The enhanced version of Blockchain App Builder includes support for auditor roles when using the extended Token Taxonomy Framework standard.

The version of Blockchain App Builder supplied with Oracle Blockchain Platform Digital Assets Edition supports two additional roles for chaincode projects that use the extended Token Taxonomy Framework standard: `Token Auditor` and `Org Auditor`. These roles function similarly to the `Token Admin` and `Org Admin` roles, but auditor roles are limited to read-only access. Admin roles have read-write access.

For more information, see the controller and SDK method descriptions in Scaffolded TypeScript Project for Token Taxonomy Framework and Scaffolded Go Project for Token Taxonomy Framework.

Method Level Access Controls

Auditor roles have read-only access to methods that use the extended Token Taxonomy Framework standard.

The following table shows how the auditor roles affect access to methods in the extended Token Taxonomy Framework standard.

Method name	Roles that can access the method
`isTokenAdmin`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getAllAdmins`	`Token Admin`, `Token Auditor`
`getOrgAdmins`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getAllTokens`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getTokenById`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getTokenDecimals`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getTokensByName`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`isInRole`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`

Copyright © 2025, Oracle and/or its affiliates.

Method name	Roles that can access the method
`getTotalMintedTokens`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getNetTokens`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getTokenHistory`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getAccountsByRole`	`Token Admin`, `Token Auditor`
`getOrgAccountsByRole`	`Token Admin`, `Token Auditor`, `Org Admin` of any organization, `Org Auditor` of any organization
`getUsersByRole`	`Token Admin`, `Token Auditor`
`getOrgUsersByRole`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getTransactionById`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, transaction participant (sender, recipient, notary)
`getAllAccounts`	`Token Admin`, `Token Auditor`
`getAllOrgAccounts`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization
`getAccountsByUser`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, multiple account owner
`getUserByAccountId`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization
`getAccount`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getAccountTransactionHistory`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getAccountTransactionHistoryWithFilters`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getSubTransactionsById`	`Token Admin`, `Token Auditor`, transaction caller
`getSubTransactionsById`	`Token Admin`, `Token Auditor`, transaction caller
`getAccountBalance`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getAccountOnHoldBalance`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`

Copyright © 2025, Oracle and/or its affiliates.

Method name	Roles that can access the method
`getOnHoldIds`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getOnHoldDetailsWithOperationId`	`Token Admin`, `Token Auditor`, hold transaction participant (sender, recipient, notary)
`getOnHoldBalanceWithOperationId`	`Token Admin`, `Token Auditor`, hold transaction participant (sender, recipient, notary)
`getConversionHistory`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getAccountStatus`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getAccountStatusHistory`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, `Account Owner`
`getConversionRate`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, any account owner
`getConversionRateHistory`	`Token Admin`, `Token Auditor`, `Org Admin` of the particular organization, `Org Auditor` of the particular organization, any account owner
`getExchangePoolUser`	`Token Admin`, `Token Auditor`

Copyright © 2025, Oracle and/or its affiliates.