Oracle® Banking APIs

Connector Credential Store Guide

Patchset Release 22.2.6.0.0
G28186-01
April 2025

Oracle Banking APIs Connector Credential Store Guide, Patchset Release 22.2.6.0.0

G28186-01

Contents

Preface

Purpose

This guide is designed to help acquaint you with the Oracle Banking Digital Experience application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

Audience

This document is intended for the following audience:

- Customers
- Partners

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 customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and

Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

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.

Related Resources

For more information on any related features, refer to the following documents:

- Oracle Banking APIs Installation Manuals

Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes.

Acronyms and Abbreviations

The list of the acronyms and abbreviations used in this guide are as follows:

Table 1 Acronyms and Abbreviations

Abbreviation	Description
OBAPI	Oracle Banking APIs

1

Creating Credential Mapping

This topic provides the systematic instructions to create credential mapping.

Credential Store Mapping

The OBAPI system utilizes external integrations to facilitate seamless communication with various services. To establish these connections, credentials are required to authenticate and authorize access. These credentials are not hardcoded but rather initialized post-installation. They are subsequently encrypted and stored within the database, ensuring confidentiality and integrity. Upon application startup, the credentials undergo decryption, enabling secure loading into the system. This subsequent section outlines the procedures and guidelines for configuring and managing these credentials within the OBAPI environment.

To utilize the credential mapping functionality, retrieve the com.ofss.digx.CredentialsStore.jar file from the designated location:

OBAPI_Installer/installables/OBAPI/BASE/22.2.6.0.0/utils/tools

Running the Credential Mapping Application

Execute the application using the following command:

java -jar com.ofss.digx.CredentialsStore.jar <csv_file> <DataBaseCredentials> <DataSeedFlag> <AES_KEY>

Command Parameters :

1. <csv_file>
   Provide the path to your CSV file containing user credentials by replacing <csv_file> with the actual file location.

   CSV File Format Requirements

   The CSV file must adhere to the following structure:

   - Contain exactly three columns: type, username, and password
   - Include a header row with column names: type,username,password
   - Subsequent rows should contain individual credential entries, with each row representing a distinct set of credentials
   - Ensure that the value in the type column is unique for each credential entry

Table 1-1 Example CSV File

type	username	password
MERCHANT	OBAPI	PASSWORD111

2. <DataBaseCredentials>
   Specify the <DataBaseCredentials> parameter as a comma-delimited string comprising the following components:

   - Database username
   - Password
   - JDBC URL (in the format jdbc:oracle:thin:@host:port/service_id)

   The expected format for <DataBaseCredentials> is: username,password,jdbc_url.

   Example: User,Password123,jdbc:oracle:thin:@host:port/service_id

   Ensure accurate input of these values to establish a successful connection to the database.

3. <DataSeedFlag>
   To control the seeding of data into the digx_fw_credentials table, set the <DataSeedFlag> parameter to 'Y' to populate the table with the generated credentials. Alternatively, specify 'N' to simply display the credentials without persisting them to the database.

4. <AES_KEY>
   The <AES_KEY> parameter is the AES encryption key used to secure sensitive data. If data is already encrypted with a previous key, use the same key here. This avoids decrypting and re-encrypting existing data. Enter the key in plaintext. Handle this key securely to prevent unauthorized access.

   Example:

   password

   Example command to run this

```
java -jar com.ofss.digx.CredentialsStore.jar data.csv
        DB_USER,DB_PASSWORD,jdbc:oracle:thin:@//HOST:PORT/SERVICE_ID Y
password
```

Upon executing this utility, you will obtain an encrypted password, which can then be utilized in conjunction with other credentials. Subsequently, these credentials will be populated into the database.

Extensibility:

To leverage custom credentials inserted into the system, utilize the following code snippet:

```
ICredentialStore store =

CredentialStoreFactory.getCredentials(CredentialStoreKeys.CREDENTIAL_IPMLEM
ENTATION);
Credential credentials = store.getCredentials(<custom_type>);
```

Replace <custom_type> with the desired type associated with the custom credentials.

Import:

```
Import the jar implementation


"com.ofss.digx.infra:com.ofss.digx.infra.crypto.impl:$libs_digxVersion"
```

into your gradle project

To ensure proper configuration, verify that the entry in the digx_fw_config_all_b table has a prop_id of “credential_impl”, a category_id of “CredentialStore”, and a PROP_VALUE of “com.ofss.digx.infra.cred.DatabaseCredentialsStore”. Confirm that these values match exactly to guarantee correct functionality. If discrepancies are found, update the entry accordingly to reflect the specified values.

Note:

```
The AES key is no longer stored in the Credential Store but inside
a keystore
          in DIGX_FW_KEYSTORE.
For any encryption operations that require the use of the AES key,
utilize
          the SymmetricCryptographyProviderFactory class, which is
available in the same
          JAR, instead of relying on the credential. This approach
streamlines the encryption
          process and enhances overall
security.SymmetricCryptographyProviderFactory.getInstance().getLate
stProvider()
.encrypt(data);SymmetricCryptographyProviderFactory.getInstance().g
etLatestProvider().decrypt(data);
```

Index

C