Oracle® GoldenGate

Stream Analytics Documentation

F18429-19
November 2024

Oracle GoldenGate Stream Analytics Documentation,

F18429-19

Primary Author: Oracle Corporation

Contents

4 Manage

5 Transform

9 Reference

10 Troubleshoot

1
Overview

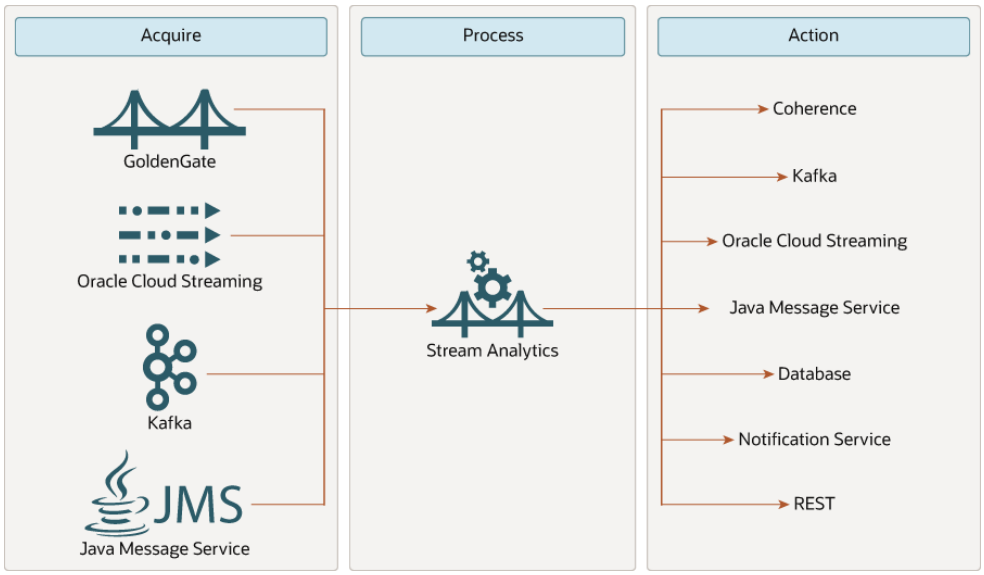
1.1 Introduction

The Oracle GoldenGate Stream Analytics (GGSA) runtime component is a complete solution platform for building applications to filter, correlate, and process events in real-time. With flexible deployment options of stand-alone Spark or Hadoop-YARN, it proves to be a versatile, high-performance event processing engine. GGSA enables Fast Data and Internet of Things (IOT) – delivering actionable insight and maximizing value on large volumes of high velocity data from varied data sources in real-time. It enables distributed intelligence and low latency responsiveness by pushing business logic to the network edge.

1.2 Key Features of GGSA

- Natively integrated with Oracle GoldenGate to process and analyze transaction streams from relational databases
- Interactive pipeline designer with live results to instantly validate your work
- Zero-code environment to build continuous ETL and analytics workflows
- Pattern library for advanced data transformation and real-time analytics
- Extensive support for processing geospatial data
- Secured connectivity to diverse data sources and sinks
- Built-in support for real-time visualizations and dashboards
- Automatic application state management
- Automatic configuration of pipelines for high availability and reliability
- Automatic configuration of pipelines for lower latency and higher throughput
- Automatic log management of pipelines for better disk space utilization

1.3 GGSA Architecture



Acquiring data

Stream Analytics can acquire data from any of the following on-premises and cloud-native data sources:

- **GoldenGate**: Natively integrated with Oracle GoldenGate, Stream Analytics offers data replication for high-availability data environments, real-time data integration, and transactional change data capture.
- **Oracle Cloud Streaming**: Ingest continuous, high-volume data streams that you can consume or process in real-time.
- **Kafka**: A distributed streaming platform used for metrics collection and monitoring, log aggregation, and so on.
- **Java Message Service**: Allows java-based applications to send, receive, and read distributed communications.

Processing data

With Stream Analytics, you can filter, correlate, and process events in real-time.

Perform actions on the data

After Stream Analytics processes the data, you can output the results to any one of the following external target data sources:

- Coherence
- Kafka
- Oracle Cloud Streaming
- Java Message Service
- Database
- Notification
- REST

1.4 Steps to build Continuous-ETL and Realtime-Analytics Pipelines

Step	Action	Description
Step 1	Create a Connection	You must create a connection to an external system, to be Supported stream sources: • Kafka • OCI Streaming Service • JMS • Oracle Advanced Queuing See Create Connections.
Step 2	Create a Stream	From the Catalog, create a Stream using the Connection from **Step 1**. Supported stream definitions: • File • Kafka • JMS • AQ • GoldenGate See Create Streams.
Step 3	Create a Pipeline	From the Catalog, create a Pipeline using the Stream from **Step 2**. See Create a Pipeline.
Step 4	Add Business Logic	Transform the input data stream, add business logic to the pipeline to analyze the input data stream. See Transform. See Analyze.
Step 5	Publish the Pipeline	See Publishing a Pipeline.

2
Install

2.1 Planning Your Installation

To plan the installation of GoldenGate Stream Analytics (GGSA) 19.1.0.0.* efficiently, ensure that you have the required hardware and software. You should also perform the prerequisite procedures before starting the installation process.

You can use the information in the certification matrix before installing GoldenGate Stream Analytics 19.1.0.0.*. The certification matrix provides you useful links to support pages, supported software, and system requirements in general. The following software is required for operation of GGSA:

- Oracle JDK 8 Update 131 and higher versions
- Repository Database
  - Oracle Database versions 12.2.0.1 or higher, 12.1.0.1 or higher, and 11.2.0.4 or higher
  - Else, you can use MySQL version 5.6 or 5.7
- A running Kafka cluster:
  - Version 0.10.2 to 2.2.1 for releases **19.1.0.0.1 to 19.1.0.0.7**.
  - Version 0.10.2 to 3.4.0 for release **19.1.0.0.8**.
  - Versions 3.2.0 to 3.7.0 for release **19.1.0.0.9**.
- Locally installed Spark Libraries. GGSA does not package the Spark client libraries, so you will also need locally installed Spark:

  For releases 19.1.0.0.0 to 19.1.0.0.7:

  - Spark release: `2.4.3`
  - Package type: `Pre-built for Apache Hadoop 2.7 and later`
  - Download Spark: `spark-2.4.3-bin-hadoop2.7.tgz`

  For release 19.1.0.0.8:

  - Spark release: `3.4.0`
  - Package type: `Pre-built for Apache Hadoop 3.3 and later`
  - Download Spark: `spark-3.4.0-bin-hadoop3.tgz`

  For release 19.1.0.0.9:

Spark Release	Package Type	Download Spark	Kafka Compatibility
3.4.0	Pre-built for Apache Hadoop 3.3 and later	spark-3.4.0-bin-hadoop3.tgz	– Kafka 3.7.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 – Kafka 3.6.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 – Kafka 3.5.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 – Kafka 3.2.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4
3.5.0	Pre-built for Apache Hadoop 3.3 and later	spark-3.5.0-bin-hadoop3.tgz	– Kafka 3.7.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 – Kafka 3.6.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 – Kafka 3.5.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 – Kafka 3.2.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4

- **Note:**

  Install Spark and JDK in the same node on which you plan to install Oracle Stream Analytics. See Installing GoldenGate Stream Analytics.

- Google Chrome browser with version 6.0 or higher

2.2 Installing GoldenGate Stream Analytics

After you have reviewed the above software prerequisites, please follow the steps below to install GoldenGate Stream Analytics 19.1.0.0.*:

1. Create a directory, for example, `spark-downloads`, and download Apache Spark into the newly created folder and as specified by versions below:

   For releases 19.1.0.0.0 to 19.1.0.0.7:

   - Spark release: `2.4.3`

- Package type: `Pre-built for Apache Hadoop 2.7 and later`
- Download Spark: `spark-2.4.3-bin-hadoop2.7.tgz`

For release 19.1.0.0.8 :

- Spark release: `3.4.0`
- Package type: `Pre-built for Apache Hadoop 3.3 and later`
- Download Spark: `spark-3.4.0-bin-hadoop3.tgz`

For release 19.1.0.0.9:

Spark Release	Package Type	Download Spark	Kafka Compatibility
`3.4.0`	`Pre-built for Apache Hadoop 3.3 and later`	`spark-3.4.0-bin-hadoop3.tgz`	• Kafka 3.7.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 • Kafka 3.6.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 • Kafka 3.5.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 • Kafka 3.2.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4
`3.5.0`	`Pre-built for Apache Hadoop 3.3 and later`	`spark-3.5.0-bin-hadoop3.tgz`	• Kafka 3.7.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 • Kafka 3.6.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 • Kafka 3.5.1 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4, Yarn 2.7.1 • Kafka 3.2.0 works with Spark standalone, Yarn 3.3.6, Yarn 3.3.4

2. Extract the Spark archive to a local directory.

   You can see a subfolder, `spark-*.*.*-bin-hadoop*.*`.

3. Create a new directory, for example, `OSA-19` and download `OSA-19.1.0.0.*.zip` from Oracle eDelivery and extract it into the newly created folder.

   You can find the `OSA-19.1.0.0.*-README.txt` file in the `OSA-19.1.0.0.*` zip file.

4. Extract the downloaded file. You should now see a subfolder `OSA-19.1.0.0.*`.
5. Review the file `OSA-19.1.0.0.*-README.txt` in the OSA-19 folder.
6. Set the environment variables:
   - Set the `SPARK_HOME` environment variable in the `OSA-19.1.0.0.*/osa-base/etc/osa-env.sh` file to point to the directory where you have extracted the Spark archive. For example:

     ```
     SPARK_HOME=/products/spark-downloads/spark-*.*.*-bin-hadoop*.*
     ```
7. Set the `JDK_HOME` environment variable in the `OSA-19.1.0.0.*/osa-base/etc/osa-env.sh` file to point to the directory where you have extracted the JDK archive. For example:

   ```
   JDK_HOME=/products/java-downloads/jdk1.8.0_131
   ```

2.3 Configuring the Metadata Store

Please follow steps below for configuring your metadata store.

1. Configure your data source in `OSA-19.1.0.0.*/osa-base/etc/jetty-osa-datasource.xml` as per instructions below. This step is essential for creating OSA's database schema. The OSA database user referred to in the document will be created by the installation process.
2. Uncomment and edit one of the two Data source configurations, either for Oracle Database or MySQL depending on the database you want to use as metadata store. The uncommented fragment for Oracle database is shown below: **a.**

```
<New id="osads"
class="org.eclipse.jetty.plus.jndi.Resource">
          <Arg>
          <Ref refid="wac"/>
          </Arg>
              <Arg>jdbc/OSADataSource</Arg>
            <Arg>
             <New class="oracle.jdbc.pool.OracleDataSource">
                <Set
name="URL">jdbc:oracle:thin:@myhost.example.com:1521:OSADB</Set>
                <Set name="User">OSA_USER</Set>
                <Set name="Password">
                  <Call class="org.eclipse.jetty.util.security.Password"
name="deobfuscate">
                    <Arg> OBF:OBFUSCATED_PASSWORD</Arg>
                  </Call>
                </Set>
                <Set name="connectionCachingEnabled">true</
Set>
                <Set name="connectionCacheProperties">
                 <New class="java.util.Properties">
                 <Call name="setProperty"><Arg>MinLimit</Arg><Arg>1</Arg></
Call>
                 <Call name="setProperty"><Arg>MaxLimit</Arg><Arg>15</
Arg></Call>
                 <Call name="setProperty"><Arg>InitialLimit</Arg><Arg>1</
Arg></Call>
                 </New>
               </Set>
```

```
            </New>
          </Arg>
        </New>
```

3. Decide on an OSA schema username and a plain-text password. For illustration, say **osa** as schema user name and **alphago** as password.
   Change directory to top-level folder `OSA-19.1.0.0.*` and execute the following command:`java -cp ./lib/ jetty-util-9.4.17.v20190418.jar org.eclipse.jetty.util.security.Password osa <your password>`

   For example, `java -cp ./lib/ jetty-util-9.4.17.v20190418.jar org.eclipse.jetty.util.security.Password osa alphago`

   You should see results like below on console:

   ```
   2019-06-18 14:14:45.114:INFO::main: Logging initialized @1168ms to
   org.eclipse.jetty.util.log.StdErrLogalphago
   OBF:<obfuscated password>

   MD5:34d0a556209df571d311b3f41c8200f3

   CRYPT:osX/8jafUvLwA
   ```

4. Note down the obfuscated password string that is displayed (shown in bold), by copying it to clipboard or notepad.
5. Change database host, port, SID, osa schema user name and osa schema password fields marked in bold in the code in Step 2a.
   Example - jdbc:oracle:thin:@myhost.example.com:1521:ORCL

SAMPLE JETTY-OSA-DATASOURCE.XML

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/
jetty/configure_9_3.dtd">

<!-- ============================================================= -->
<!-- Configure jdbc/OSADataSource data source                      -->
<!-- ============================================================= -->
<Configure id="Server" class="org.eclipse.jetty.server.Server">

    <!-- SAMPLE OSA DATASOURCE CONFIGURATION FOR ORACLE-->
    <New id="osads" class="org.eclipse.jetty.plus.jndi.Resource">
         <Arg>
             <Ref refid="wac"/>
         </Arg>
         <Arg>jdbc/OSADataSource</Arg>
         <Arg>
             <New class="oracle.jdbc.pool.OracleDataSource">
                 <Set
name="URL">jdbc:oracle:thin:@myhost.example.com:1521:OSADB</Set>
                 <Set name="User">osa_prod</Set>
                 <Set name="Password">
                    <Call class="org.eclipse.jetty.util.security.Password"
name="deobfuscate">

<Arg>OBF:1ggz1j1u1k8q1leq1v2h1w8v1v1x1lcs1k5g1iz01gez</Arg>
                    </Call>
                </Set>
```

```
                <Set name="connectionCachingEnabled">true</Set>
                <Set name="connectionCacheProperties">
                    <New class="java.util.Properties">
                        <Call name="setProperty"><Arg>MinLimit</Arg><Arg>1</
Arg></Call>
                        <Call name="setProperty"><Arg>MaxLimit</Arg><Arg>15</
Arg></Call>
                        <Call name="setProperty"><Arg>InitialLimit</
Arg><Arg>1</Arg></Call>
                    </New>
                </Set>
            </New>
        </Arg>
     </New>


    <!-- SAMPLE OSA DATASOURCE CONFIGURATION FOR ADW-->
    <!--
    <New id="osads" class="org.eclipse.jetty.plus.jndi.Resource">
         <Arg>
             <Ref refid="wac"/>
         </Arg>
         <Arg>jdbc/OSADataSource</Arg>
         <Arg>
             <New class="oracle.jdbc.pool.OracleDataSource" type="adw">
                 <Set name="URL">jdbc:oracle:thin:@oracletestdb_high?
TNS_ADMIN=/scratch/oracletest/Wallet_oracletestdb</Set>
                 <Set name="User">{OSA_USER}</Set>
                 <Set name="Password">
                    <Call class="org.eclipse.jetty.util.security.Password"
name="deobfuscate">
                        <Arg>{OBF:OBFUSCATE_PASSWORD}</Arg>
                    </Call>
                </Set>
                <Set name="connectionCachingEnabled">true</Set>
                <Set name="connectionCacheProperties">
                    <New class="java.util.Properties">
                        <Call name="setProperty"><Arg>MinLimit</Arg><Arg>1</
Arg></Call>
                        <Call name="setProperty"><Arg>MaxLimit</Arg><Arg>15</
Arg></Call>
                        <Call name="setProperty"><Arg>InitialLimit</
Arg><Arg>1</Arg></Call>
                    </New>
                </Set>
            </New>
         </Arg>
     </New>
     -->
    <!-- SAMPLE OSA DATASOURCE CONFIGURATION FOR MYSQL-->
    <!--

    <New id="osads" class="org.eclipse.jetty.plus.jndi.Resource">
        <Arg>
            <Ref refid="wac"/>
        </Arg>
```

```
        <Arg>jdbc/OSADataSource</Arg>
        <Arg>
            <New class="com.mysql.cj.jdbc.MysqlConnectionPoolDataSource">
                <Set name="URL">jdbc:mysql://examplehost.com:3306/OSADB</Set>
                <Set name="User">{OSA_USER}</Set>
                <Set name="Password">
                    <Call class="org.eclipse.jetty.util.security.Password"
name="deobfuscate">
                        <Arg>{OBF:OBFUSCATE_PASSWORD}</Arg>
                    </Call>
                </Set>
            </New>
        </Arg>
    </New>
-->

</Configure>
```

> **Note:**
>
> Do not use a hyphen in the OSA metadata username, in the jetty-osa-datasource.xml

.

2.3.1 Configuring ATP/ADW as Metadata Store

GoldenGate Stream Analytics creates the metadata schema, as part of initial configuration of the system, using the script: `${OSA_HOME}/osa-base/bin/configure.sh dbroot=<sys user of database> dbroot_password=<sys user password of the database>`

However, before running the above script, you must configure the datasource in the datasource configuration file at`${OSA_HOME}/osa-base/etc/jetty-osa-datasource.xml`.

To configure ATP/ADW as metadata store, first comment the Oracle and MYSQL sections, while uncommenting the ADW/APT section in `jetty-osa-datasource.xml` file.

Below is the template for the datasource configuration for ATP/ADW database:

`jetty-osa-datasource.xml`

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://www.eclipse.org/
jetty/configure_9_3.dtd">
<Configure id="Server" class="org.eclipse.jetty.server.Server">
    <New id="osads" class="org.eclipse.jetty.plus.jndi.Resource">
        <Arg>
            <Ref refid="wac"/>
        </Arg>
        <Arg>jdbc/OSADataSource</Arg>
        <Arg>
            <New class="oracle.jdbc.pool.OracleDataSource" type="adw">
                <Set name="URL">jdbc:oracle:thin:@{service_name}?
```

```
TNS_ADMIN={wallet_absolute_path}</Set>
                 <Set name="User">{osa_db_user}</Set>
                 <Set name="Password">
                    <Call class="org.eclipse.jetty.util.security.Password"
name="deobfuscate">
                        <Arg>{obfuscated_password}</Arg>
                    </Call>
                </Set>
                <Set name="connectionCachingEnabled">true</Set>
                <Set name="connectionCacheProperties">
                    <New class="java.util.Properties">
                        <Call name="setProperty"><Arg>MinLimit</Arg><Arg>1</
Arg></Call>
                        <Call name="setProperty"><Arg>MaxLimit</Arg><Arg>15</
Arg></Call>
                        <Call name="setProperty"><Arg>InitialLimit</
Arg><Arg>1</Arg></Call>
                    </New>
                </Set>
            </New>
        </Arg>
    </New>
</Configure>
```

> **Note:**
>
> In the above template, replace the variables in {} as below:
>
> - {*service_name*} - one of the service names listed in the tnsnames.ora file inside the wallet
> - {*wallet_absolute_path*} - the absolute path of wallet folder on the machine where OSA is installed
> - {*osa_db_user*} - the username to create the osa metadata. This username and schema will be created by the 'dbroot' user provided in above script.
> - {*obfuscated_password*} - the Obfuscated password for {osa_db_user}

2.4 Initializing Metadata Store

This topic applies only to Oracle user-managed services.

After installing GGSA, you need to configure the metadata store with the database admin credential details and the version of GGSA as required.

To initialize the metadata store, you need database admin credentials with sysdba privileges:

1. Change directory to `OSA-19.1.0.0.*/osa-base/bin`.
2. Execute the following command: `./start-osa.sh dbroot=<db sys user> dbroot_password=<db sys password>` For example, `./start-osa.sh dbroot=AlphaUser dbroot_password=AlphaPassword`

3. Following console messages will be displayed indicating the OSA schema was created and prompting for password for **osaadmin** user.

   The following console messages indicates that the GGSA schema is created and the metadata store is successfully initialized:

```
JAVA_HOME: JAVA_HOME: /<jdk install path>/jdk1.8.0_121.
         SPARK_HOME: /<Spark install path>/spark-*.*.*-bin-hadoop*.*
         The RELEASE file exists:
         /<Spark install path>/spark-*.*.*-bin-hadoop*.*/RELEASE
         SPARK_VERSION: Spark *.*.* built for Hadoop *.*
         2019-06-23 08:48:51.444:INFO::main:
         Logging initialized @305ms to org.eclipse.jetty.util.log.StdErrLog
         OSA DB user created:osa
         The OSA application administrative user with the predefined name
"osaadmin" is going to be created
         You have not specified a password
         for the "osaadmin" user on the command line. Please enter it
below.
```

4. Enter password:
5. Re-enter password:
6. Run `./stop-osa.sh` to complete schema creation and metadata initialization.
7. If you don't see the above messages, check the OSA-19.1.0.0.*/osa-base/logs folder to identify the cause and potential solution.

> **Note:**
>
> If you do not have the database admin credentials, ask your database administrator to create a GoldenGate Stream Analytics database user by using the SQL scripts available in the `OSA-19.1.0.0.*/osa-base/sql` folder. The GoldenGate Stream Analytics database username must match the one configured in `jetty-osa-datasource.xml`.

2.5 Jetty Properties File

Use the jetty properties available at `OSA-19.1.0.0.*/osa-base/etc/jetty.properties`, to modify certain security features.

> **Note:**
>
> It is recommended that you configure these properties at the installation stage, to avoid restarting your server, if configured at a later stage.

Following are the available properties:

- **jetty.session.timeout**
  You can set the timeout for OSA web session. This sets the timeout for OSA web session. By default the timeout is set to 30 minutes. The value can be changed to any integer greater than 1.

- **host.headers.whitelist**
  You can restrict the x-forwarded-host header values to the values defined with this property.

  Example: `host.headers.whitelist= www.oracle.com, www.microsoft.com, localhost:9080`

  Here the value of the host header can be only of these three domains listed. Commenting out this property with a # will allow all values for the header.

  > **Note:**
  >
  > If you do not specify explicitly the host header in your request, the default value is `host-server:port`, where the OSA jetty server is running. Hence you must specify the port number along with the server address.

- **xforwarded.host.headers.whitelist**
  You can restrict the x-forwarded-host header values to the values defined with this property.

  Example: `xforwarded.host.headers.whitelist= www.oracle.com, www.microsoft.com, localhost`

  Here the value of the x-forwarded-host header can be only of these three domains listed. Commenting out this property with a # will allow all values for the header. If no domain is entered, that is, if the value of the property is empty, then this header is not supported.

- **response.headers.list**
  A comma separated list of response headers, which will be sent along with response for every request.

  Example: `response.headers.list="x-frame-options: sameorigin, X-Content-Type-Options: nosniff"`

  By default the above 2 response headers are set.

  - `x-frame-options`: sameorigin will prevent clickjack attacking.
  - `X-Content-Type-Options`: nosniff will prevent sniffing of the response content by the browsers.

2.6 Adjusting Jetty Threadpool

Edit `OSA-19.1.0.0.*/etc/jetty-threadpool.xml` to change minimum and maximum thread configuration to 100 and 2000 respectively. Sample shown below.

```
<New id="threadPool"
    class="org.eclipse.jetty.util.thread.QueuedThreadPool">
 <Set name="minThreads" type="int"><Property
name="jetty.threadPool.minThreads"
      deprecated="threads.min" default="100"/></Set>
<Set name="maxThreads" type="int"><Property name="jetty.threadPool.maxThreads"
      deprecated="threads.max" default="2000"/></Set>
<Set name="reservedThreads" type="int"><Property
      name="jetty.threadPool.reservedThreads" default="-1"/></Set>
 <Set name="idleTimeout" type="int"><Property
      name="jetty.threadPool.idleTimeout" deprecated="threads.timeout"
      default="60000"/></Set>
```

```
<Set name="detailedDump" type="boolean"><Property
      name="jetty.threadPool.detailedDump" default="false"/></Set>
</New>
</Configure>
```

2.7 Integrating Stream Analytics with Oracle GoldenGate

Follow the below steps to integrate Oracle Goldengate with Stream Analytics:

1. Download and install Oracle GoldenGate Big Data. For a compatible version of Oracle GoldenGate Big Data, see the latest certification matrix.

   > **Note:**
   >
   > Install Oracle GoldenGate Big Data on the same machine and with the same user as OSA.

2. Set the following environment variables:
   - *KAFKA_HOME* – set this variable to the path where Kafka is installed.
     Example :`export KAFKA_HOME=/u01/app/kafka`.
   - *LD_LIBRARY_PATH* – set this variable to the directory path that contains JVM shared library.
     Example: `export LD_LIBRARY_PATH=/u01/app/java/jre/lib/amd64/server:$LD_LIBRARY_PATH`
   - *GGBD_HOME* – set this variable to the path where Goldengate for Bigdata is installed.
     Example: `export GGBD_HOME=/u01/app/OGG_BigData_Linux_x64_19.1.0.0.0`
3. Start the manager process on port 7801.

For installation steps, see Installing GoldenGate for Big Data.

2.8 Maven Setting for GoldenGate Big Data Handlers

Maven is required to download third-party client libraries for the GGBD handlers to work.

2.8.1 Set the Maven Home Path

To configure maven home:

Update the `OSA-19.1.0.0.*/osa-base/bin/configure-osa.sh` with the correct M2_HOME path, as below:
Change the path from

```
OSA_HOME="$( cd "$(dirname "../../")" >/dev/null 2>&1 ; pwd -P )"
```

to

```
OSA_HOME="$( cd "$(dirname "../../../")" >/dev/null 2>&1 ; pwd -P )"
```

> **Note:**
>
> Update the maven home path before initialization of the metadata store, or you will have to restart GGSA after this update.

2.8.2 Configure Maven Proxy Settings

If your GGSA installation is behind proxy, to use the GGBD handlers, you have to configure the `settings.xml` that comes with the Maven distribution.

Update the `<OSA_INSTALLATION_PATH>/apache-maven-3.6.3/conf/settings.xml` with the correct proxy entries in the `<proxies> </proxies>` section, as shown below:

```
<proxy>
      <id>optional</id>
      <active>true</active>
      <protocol>http</protocol>
      <username>proxyuser</username>
      <password>proxypass</password>
      <host>proxy.host.net</host>
      <port>80</port>
      <nonProxyHosts>local.net|some.host.com</nonProxyHosts>
    </proxy>
```

> **Note:**
>
> Username and password field is required if the proxy is protected.

> **Note:**
>
> Update the `settings.xml` before initialization of the metadata store, or you will have to restart GGSA after this update.

2.9 GoldenGate Stream Analytics Hardware Requirements for Enterprise Deployment

This chapter provides the hardware requirements for GoldenGate Stream Analytics Design and Data tiers.

Design Tier

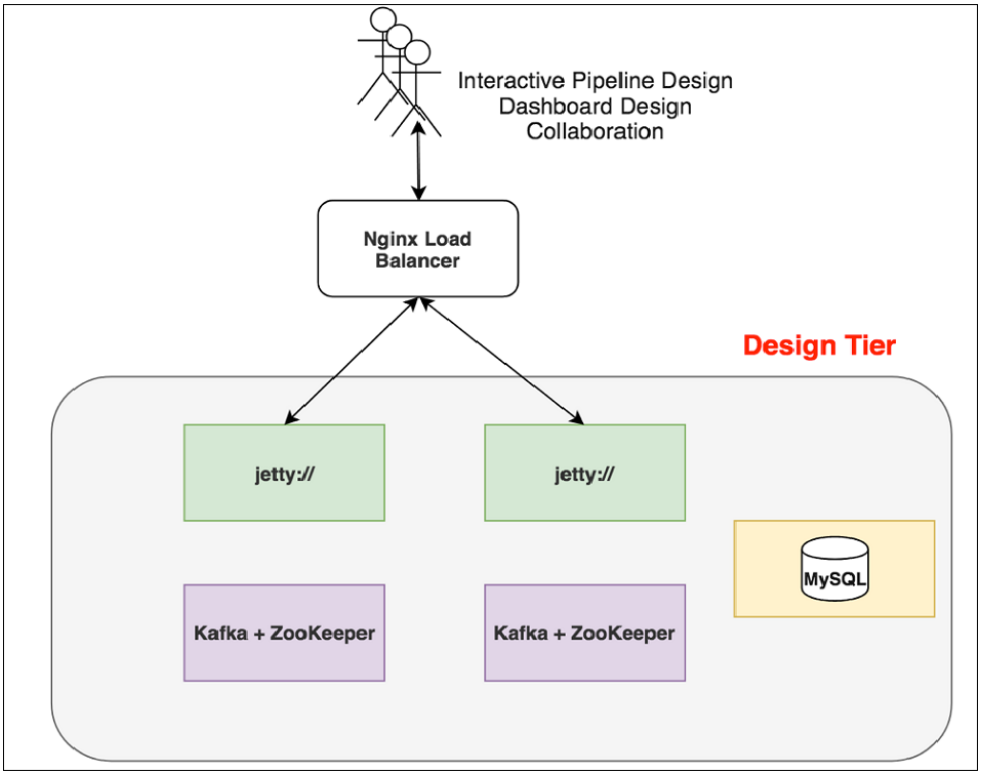
GoldenGate Stream Analytics' Design-tier is a multi-user environment that allows users to implement and test dataflow pipelines. The design-tier also serves dashboards for streaming data. Multiple users can build, test, and deploy pipelines based on the capacity of the Runtime-tier (YARN/Spark cluster) simultaneously.

GGSA uses Jetty as the web-server with support for HA. For production deployments of GGSA Design-tier, you require the minimum hardware configuration listed below:

- Web server – Jetty with High Availability (HA) support
- 2 nodes with 4+ cores and 32+ GB of RAM for running two instances of Jetty.
- 1 node with 4+ cores and 16+ GB of RAM for running MySQL or Oracle meta-store.
- 2 nodes with 4+ cores and 16+ GB of RAM for running two instances of Kafka and 3 instances of ZooKeeper. Please note this is a separate Kafka cluster for GGSA's internal use and for interactively designing pipelines. ZooKeeper end-point of this Kafka cluster must be specified in GGSA's system settings UI.

> **Note:**
>
> The two-node Kafka cluster can be avoided if customer already has a Kafka cluster in place and is fine with OSA leveraging that cluster for its internal usage.

Based on the above estimates, total cores for design-tier is 12 and approximate memory is 112 GB RAM. Jetty instances can be independently scaled as the number of users increase. Diagram below illustrates GGSA's Design-tier topology.



Data Tier

The deployed pipelines are run on the YARN or Spark cluster. You can use existing YARN/ Spark clusters if you have sufficient spare capacity.

Sizing Guidelines

Use the following sizing guidelines to run GGSA pipelines. Ensure that the pipelines are deployed on shared storage, so that the pipeline code and libraries are accessible from all

nodes in the YARN/Spark cluster. GGSA supports NFS for shared storage but if you want to use HDFS, the hardware needs two more nodes.

- – 2 nodes with 4+ cores, 16+GB RAM, and 500 GB local disk to run HDFS cluster, two instances of HDFS name and data nodes.

The Spark tier is where work happens and the Spark cluster size depends on

- Number of pipelines that will simultaneously run
- Logic in each pipeline
- Desired degree of parallelism

For each streaming pipeline the number of cores and memory gets computed based on a required degree of parallelism. As an example, consider a pipeline ingesting data from customer's Kafka topic T with 3 partitions using direct ingestion. Direct ingestion is where no Spark Receivers are used. In this case, the minimum number of processes that you need to run for optimal performance is as follows: 1 Spark Driver Process + 3 Executor processes, 1 for each Kafka Topic partition. Each Executor process needs a minimum of 2 cores.

The number of cores for a pipeline can be computed as

--executor-cores = 1 + Number of Executors * 2

In case of Receiver-based ingestion as in JMS, it is computed as

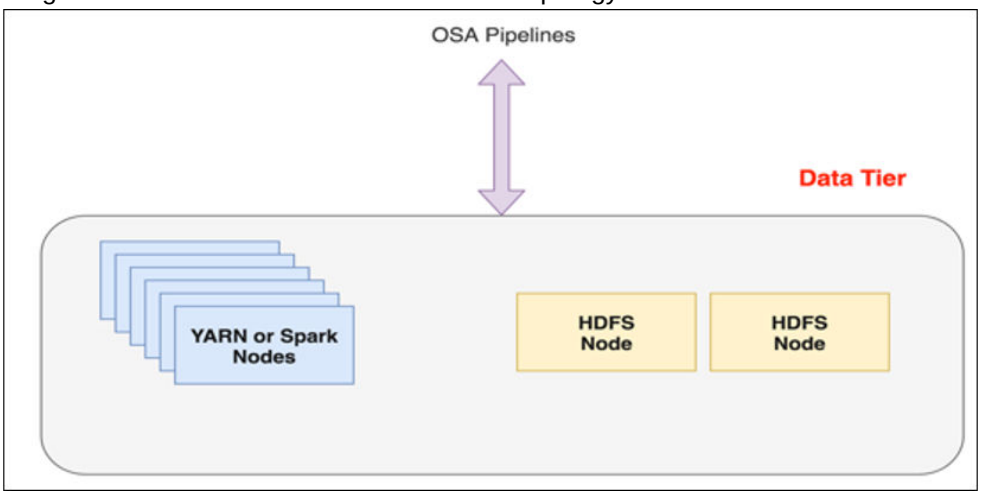
--executor-cores = 2 + Number of Executors * 2

This is rough estimates and environments where fine-grained scheduling is not available. In environments like Kubernetes, we have the luxury of more fine-grained scheduling.

The formula for sizing memory is

(Number of Windows * Average Window Range * Event Rate * Event Size) + (Number of Lookup/Reference Objects being cached * Size of Lookup Object).

Diagram below illustrates GGSA's Data-tier topology.



If you are considering GGSA for POCs and not production, then you can use the following configuration:

Design Tier

- An instance of the Jetty running on a 4+ core node with a 32+ GB of RAM.
- An instance of MySQL/Oracle for metadata store on a 4+ core node with a 16+ GB of RAM.

- A node of the Kafka cluster running on a 4+ core node with 16+ GB of RAM.

> **Note:**
>
> This is a separate Kafka cluster for GGSA's internal use and for interactively designing pipelines.

Data Tier

- A Hadoop Distributed File System (HDFS) cluster node running on 4+ core physical node with 16+ GB of RAM.
- 2 nodes of the YARN/Spark cluster each running on a 4+ core physical node with a 16+ GB of RAM.

Development Mode Configurations

Design Tier

- 1 node with 4+ cores and 16+ GB of RAM for 1 instance of Jetty, 1 instance of MySQL DB, and 1 instance of Kafka+ZooKeeper

Data Tier

- 1 node with 4+ cores and 16+ GB of RAM for 1 instance of HDFS and 1 instance of YARN/Spark.

2.10 Retaining https and Disabling http

1. By default, the GGSA web application is available on both http (port 9080) and https (port 9443). Follow the procedure below if you intend to disable http.
2. Edit file `osa-base/start.d/http.ini`.
3. Comment out as follows: `##--module=http`.
4. Start GGSA web server by running `osa-base/bin/start-osa.sh`.

2.11 Setting up Runtime for GoldenGate Stream Analytics Server

Before you start using GoldenGate Stream Analytics, you need to specify the runtime server, environment, and node details. You must do this procedure right after you launch GoldenGate Stream Analytics (GGSA) for the first time.

1. Change directory to `OSA-19.1.0.0.*.*/osa-base/bin and run ./start-osa.sh`. You should see the following message on console.

   ```
   Supported OSA schema versions are: [18.4.3, 18.1.0.1.0, 18.1.0.1.1,
   19.1.0.0.0, 19.1.0.0.1, 19.1.0.0.2, 19.1.0.0.3, 19.1.0.0.5, 19.1.0.0.6,
   19.1.0.0.7, 19.1.0.0.8]

   The schema is preconfigured and current. No changes or updates are required.
   ```

   If you do not see the above message, please check the log file in `OSA-19.1.0.0.*.*/osa-base/logs` folder.
2. the Chrome browser, enter `localhost:9080/osa` to access Oracle Stream Analytics login page, and login using your credentials.

> **Note:**
>
> The password is a plain-text password.

3. Click the user name at the top right corner of the screen.
4. Click **System Settings**.
5. Click **Environment**.
6. Select the **Runtime Server**. See the sections below for Yarn and Spark Standalone runtime configuration details.

Yarn Configuration

1.
   - **YARN Resource Manager URL**: Enter the URL where the YARN Resource Manager is configured.
   - **Storage**: Select the storage type for pipelines. To submit a GGSA pipeline to Spark, the pipeline has to be copied to a storage location that is accessible by all Spark nodes.
     - **If the storage type is WebHDFS**:
       * **Path**: Enter the WebHDFS directory (hostname:port/path), where the generated Spark pipeline will be copied to and then submitted from. This location must be accessible by all Spark nodes. The user specified in the authentication section below must have read-write access to this directory.
       * **HA Namenodes**: Set the HA namenodes. If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.
     - **If storage type is HDFS**:
       * **Path**: The path could be `<HostOrIPOfNameNode><HDFS Path>`. For example, `xxx.xxx.xxx.xxx/user/oracle/ggsapipelines`. Hadoop user must have `Write` permissions. The folder will automatically be created if it does not exist.
       * **HA Namenodes**: If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.
     - **If storage type is NFS**:
       Path: The path could be `/oracle/spark-deploy`.

   > **Note:**
   >
   > /oracle should exist and spark-deploy will automatically be created if it does not exist. You will need `Write` permissions on the /oracle directory.

2. **Hadoop Authentication:**
   - **Simple** authentication credentials:
     - **Protection Policy**: Select a protection policy from the drop-down list. This value should match the value on the cluster.

  - **Username**: Enter the user account to use for submitting Spark pipelines. This user must have read-write access to the Path specified above.

- **Kerberos** authentication credentials:
  - **Protection Policy**: Select a protection policy from the drop-down list. This value should match the value on the cluster.
  - **Kerberos Realm**: Enter the domain on which Kerberos authenticates a user, host, or service. This value is in the `krb5.conf file`.
  - **Kerberos KDC**: Enter the server on which the Key Distribution Center is running. This value is in the `krb5.conf file`.
  - **Principal**: Enter the GGSA service principal that is used to authenticate the GGSA web application against Hadoop cluster, for application deployment. This user should be the owner of the folder used to deploy the GGSA application in HDFS. You have to create this user in the yarn node manager as well.
  - **Keytab**: Enter the keytab pertaining to GGSA service principal.
  - **Yarn Resource Manager Principal**: Enter the yarn principal. When Hadoop cluster is configured with Kerberos, principals for hadoop services like hdfs, https, and yarn are created as well.

3. **Yarn master console port**: Enter the port on which the Yarn master console runs. The default port is `8088`.
4. Click **Save**.

Spark Standalone

1. Select the **Runtime Server** as **Spark Standalone**, and enter the following details:
   - **Spark REST URL**: Enter the Spark standalone REST URL. If Spark standalone is HA enabled, then you can enter comma-separated list of active and stand-by nodes.
   - **Storage**: Select the storage type for pipelines. To submit a GGSA pipeline to Spark, the pipeline has to be copied to a storage location that is accessible by all Spark nodes.
     - **If the storage type is WebHDFS**:
       * **Path**: Enter the WebHDFS directory (hostname:port/path), where the generated Spark pipeline will be copied to and then submitted from. This location must be accessible by all Spark nodes. The user specified in the authentication section below must have read-write access to this directory.
       * **HA Namenodes**: If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.
     - **If storage type is HDFS**:
       * **Path**: The path could be `<HostOrIPOfNameNode><HDFS Path>`. For example, `xxx.xxx.xxx.xxx/user/oracle/ggsapipelines`. Hadoop user must have `Write` permissions. The folder will automatically be created if it does not exist.
       * **HA Namenodes**: If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.

         This field is applicable only when the storage type is HDFS.

- **Hadoop Authentication for WebHDFS and HDFS Storage Types:**
  - **Simple** authentication credentials:
    - **Protection Policy**: Select a protection policy from the drop-down list.
    - **Username**: Enter the user account to use for submitting Spark pipelines. This user must have read-write access to the Path specified above.
  - **Kerberos** authentication credentials:
    - **Protection Policy**: Select a protection policy from the drop-down list.
    - **Kerberos Realm**: Enter the domain on which Kerberos authenticates a user, host, or service. This value is in the `krb5.conf file`.
    - **Kerberos KDC**: Enter the server on which the Key Distribution Center is running. This value is in the `krb5.conf file`.
    - **Principal**: Enter the GGSA service principal that is used to authenticate the GGSA web application against Hadoop cluster, for application deployment. This user should be the owner of the folder used to deploy the GGSA application in HDFS. You have to create this user in the yarn node manager as well.
    - **Keytab**: Enter the keytab pertaining to GGSA service principal.
    - **Yarn Resource Manager Principal**: Enter the yarn principal. When Hadoop cluster is configured with Kerberos, principals for hadoop services like hdfs, https, and yarn are created as well.
- **If storage type is NFS**:
  Path: The path could be `/oracle/spark-deploy`.

> **Note:**
>
> /oracle should exist and spark-deploy will automatically be created if it does not exist. You will need `Write` permissions on the /oracle directory.

2. **Spark standalone master console port**: Enter the port on which the Spark standalone console runs. The default port is `8080`.

> **Note:**
>
> The order of the comma-separated ports should match the order of the comma-separated spark REST URLs mentioned in the **Path**.

3. **Spark master username**: Enter your Spark standalone server username.
4. **Spark master password**: Click **Change Password**, to change your Spark standalone server password.

> **Note:**
>
> You can change your Spark standalone server username and password in this screen. The username and password fields are left blank, by default.

5. Click **Save**.

2.12 Validating Data Flow to GoldenGate Stream Analytics

After you have configured GoldenGate Stream Analytics (GGSA) with the runtime details, you need to ensure that sample data is being detected and correctly read by GGSA.

To validate data flow into GoldenGate Stream Analytics, use the following steps:

1. Copy the six lines below into a CSV file, for example `sample.csv`.

```
ProductLn,ProductType,Product,OrderMethod,CountrySold,QuantitySold,UnitSale
Price
Personal Accessories,Watches,Legend,Special,Brazil,1,240
Outdoor Protection,First Aid,Aloe Relief,E-mail,United States,3,5.23
Camping Equipment,Lanterns,Flicker Lantern,Telephone,Italy,3,35.09
Camping Equipment,Lanterns,Flicker Lantern,Fax,United States,4,35.09
Golf Equipment,Irons,Hailstorm Steel Irons,Telephone,Spain,5,461
```

2. In the **Catalog**, as shown in the image below, click **Create New Item**, and then click **Stream**. create a stream of type File.
3. In the **Type Properties** page of the **Create Stream** dialog box, provide the **Name**, **Description**, and **Tags** for the Stream, select the **Stream Type** as **File**, and then select **Create Pipeline with this source (Launch Pipeline Editor)**.
4. Click the **Next** button to navigate to the **Source Details** page of the **Create Stream** dialog box.
5. In the **Source Details** page, click **Upload file** to upload the `sample.csv` file, and then click **Next** to navigate to the **Data Format** page.
6. In the **Data Format** page, select the **CSV Predefined Format** as **Default** and select the **First record as header**, and then click **Next** to navigate to the **Shape** page.
7. In the **Shape** page, verify that the shape of the event is successfully inferred as in the following image, and then click **Save**.
8. In the **Create Pipeline** dialog box, enter the **Name**, **Description**, **Tags** of the pipeline, select the **Stream** that you created, and then click **Save**:

   You can see the pipeline editor and you can see the message `Starting Pipeline` followed by the message `Listening to Events`.

> **Note:**
>
> This is the first access of the cluster and it takes time to copy libraries, please be patient. You should eventually see the screenshot below with single node representing the stream source.

To complete your pipeline, see Create a Pipeline.

2.13 Terminating GoldenGate Stream Analytics

You can terminate GoldenGate Stream Analytics by running a simple command.

Use the following command to terminate GoldenGate Stream Analytics:

`./stop-osa.sh from OSA-19.1.0.0.*/osa-base/bin folder`

2.14 Upgrading GoldenGate Stream Analytics

To upgrading from an existing version of GGSA to newer version:

1. Backup your metadata store using any of Oracle or MySQL backup tools. The backup is required to restore the tools in case the upgrade fails.
2. Run the ./stop-osa.sh command to stop the GGSA server.
3. Create a **OSA-19** folder and download the `OSA-19.1.0.0.*.zip` file into the newly- created folder.
4. Unzip to extract the contents of the `OSA-19.1.0.0.*.zip` file.
5. Copy `<YourVersion>/osa-base/etc/osa-env.sh to OSA-19.1.0.0.*/osa-base/etc`.

> **Note:**
>
> You can skip this step, if you are upgrading to GGSA version 19.1.0.0.8.

6. Copy `<YourVersion>/osa-base/etc/jetty-osa-datasource.xml` to `OSA-19.1.0.0.*/osa-base/etc`.
7. Run the `./start-osa.sh` command to start the OSA server. This performs the schema migration.
8. To validate the upgrade, see Validating your Installation.

3
Configure

3.1 Configure Runtime Environment

3.1.1 Mandatory Configurations

The following configurations are mandatory:

3.1.1.1 Configuring Kafka

Kafka is used as an internal transport to display live output from pipelines, errors, warnings, etc.

To configure Kafka:

1. Click the user name at the top right corner of the screen.
2. Click **System Settings**.
3. Click **Environment**.
4. Enter the **Kafka bootstrap URL**. Select the one of the available authentication methods:
   - **SSL**: Select SSL to connect to an SSL enabled Kafka cluster.
     - **Truststore**: Locate and upload the truststore file. This field is applicable only to connect to an SSL enabled Kafka cluster.
     - **Truststore Password**: Enter the truststore password.
   - **MTLS**: Select MLTS to enable 2-way authentication of both the user and the Kafka broker.
     - **Truststore**: Locate and upload the truststore file. This field is applicable only to connect to an SSL enabled Kafka cluster.
     - **Truststore Password**: Enter the truststore password.
     - **Keystore**: Locate and upload the keystore file. This field is applicable only to connect to an SSL enabled Kafka cluster.
     - **Keystore Password**: Enter the keystore password.
   - **SASL**: Select SASL if Kafka broker requires authentication.
     - **User Name**: When using OCI Streaming Kafka compatibility APIs, enter the SASL username for Kafka broker, in the following format:

       ```
       tenancyName/username/stream pool id
       ```

> **Note:**
>
> Enter the tenancyName and userName, not tenancy OCID and user OCID. Similarly, enter the stream pool ID and not the stream pool name.

You can retrieve this information from the OCI console. This field is enabled only if you have checked the SASL option.

– **Password**: Enter the SASL password, which is an authentication token that you can generate on the **User Details** page, of the OCI console.

> **Note:**
>
> Copy the authentication token when you create it, and save it for future use. You can not retrieve it at a later stage.

After this configuration, GGSA creates Kafka topics and Group IDs, to be used internally.

3.1.1.1.1 Internal Kafka Topics

The internal Kafka topics and Group ID's used by GGSA are standardized to the following naming conventions:

Kafka Topics

Topic	Resource	Operations
sx_backend_notification_<UUID>	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_messages_<UUID>	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_<stage_name>_public	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_<stage_name>_draft	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_public_<offset_number>_<stage_name>_offset	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE

Group IDs

Group ID	Resource	Operations
sx_<UUID>_receiver	Group	DESCRIBE, READ
sx_<UUID>	Group	DESCRIBE, READ
sx_<application_name>_public_<offset_number>_<stage_name>	Group	DESCRIBE, READ

3.1.1.2 Configuring the Runtime Server

For runtime, you can use a Spark Standalone or Hadoop YARN cluster. Only users with the *Administrator* permissions can set the system settings in Oracle GoldenGate Analytics.

3.1.1.2.1 Configuring for Standalone Spark Runtime

To configure for Standalone Spark Runtime:

1. Click the user name at the top right corner of the screen.
2. Click **System Settings**.
3. Click **Environment**.
4. Select the **Runtime Server** as **Spark Standalone**, and enter the following details:
   - **Spark REST URL**: Enter the Spark standalone REST URL. If Spark standalone is HA enabled, then you can enter comma-separated list of active and stand-by nodes.
   - **Storage**: Select the storage type for pipelines. To submit a GGSA pipeline to Spark, the pipeline has to be copied to a storage location that is accessible by all Spark nodes.
     - **If the storage type is WebHDFS**:
       * **Path**: Enter the WebHDFS directory (hostname:port/path), where the generated Spark pipeline will be copied to and then submitted from. This location must be accessible by all Spark nodes. The user specified in the authentication section below must have read-write access to this directory.
       * **HA Namenodes**: If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.
     - **If storage type is HDFS**:
       * **Path**: The path could be `<HostOrIPOfNameNode><HDFS Path>`. For example, `xxx.xxx.xxx.xxx/user/oracle/ggsapipelines`. Hadoop user must have `Write` permissions. The folder will automatically be created if it does not exist.
       * **HA Namenodes**: If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.

         This field is applicable only when the storage type is HDFS.
     - **Hadoop Authentication for WebHDFS and HDFS Storage Types:**
       * **Simple** authentication credentials:

        - **Protection Policy**: Select a protection policy from the drop-down list.
        - **Username**: Enter the user account to use for submitting Spark pipelines. This user must have read-write access to the Path specified above.
      - **Kerberos** authentication credentials:
        - **Protection Policy**: Select a protection policy from the drop-down list.
        - **Kerberos Realm**: Enter the domain on which Kerberos authenticates a user, host, or service. This value is in the `krb5.conf file`.
        - **Kerberos KDC**: Enter the server on which the Key Distribution Center is running. This value is in the `krb5.conf file`.
        - **Principal**: Enter the GGSA service principal that is used to authenticate the GGSA web application against Hadoop cluster, for application deployment. This user should be the owner of the folder used to deploy the GGSA application in HDFS. You have to create this user in the yarn node manager as well.
        - **Keytab**: Enter the keytab pertaining to GGSA service principal.
        - **Yarn Resource Manager Principal**: Enter the yarn principal. When Hadoop cluster is configured with Kerberos, principals for hadoop services like hdfs, https, and yarn are created as well.
    - **If storage type is NFS**:
      Path: The path could be `/oracle/spark-deploy`.

      > **Note:**
      >
      > /oracle should exist and spark-deploy will automatically be created if it does not exist. You will need `Write` permissions on the /oracle directory.

5. **Spark standalone master console port**: Enter the port on which the Spark standalone console runs. The default port is `8080`.

   > **Note:**
   >
   > The order of the comma-separated ports should match the order of the comma-separated spark REST URLs mentioned in the **Path**.

6. **Spark master username**: Enter your Spark standalone server username.
7. **Spark master password**: Click **Change Password**, to change your Spark standalone server password.

   > **Note:**
   >
   > You can change your Spark standalone server username and password in this screen. The username and password fields are left blank, by default.

8. Click **Save**.

3.1.1.2.2 Configuring for Hadoop Yarn Runtime

1. Click the user name at the top right corner of the screen.
2. Click **System Settings**.
3. Click **Environment**.
4. Select the **Runtime Server** as **Yarn**, and enter the following details:
   - **YARN Resource Manager URL**: Enter the URL where the YARN Resource Manager is configured.
   - **Storage**: Select the storage type for pipelines. To submit a GGSA pipeline to Spark, the pipeline has to be copied to a storage location that is accessible by all Spark nodes.
     - **If the storage type is WebHDFS**:
       * **Path**: Enter the WebHDFS directory (hostname:port/path), where the generated Spark pipeline will be copied to and then submitted from. This location must be accessible by all Spark nodes. The user specified in the authentication section below must have read-write access to this directory.
       * **HA Namenodes**: Set the HA namenodes. If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.
     - **If storage type is HDFS**:
       * **Path**: The path could be `<HostOrIPOfNameNode><HDFS Path>`. For example, `xxx.xxx.xxx.xxx/user/oracle/ggsapipelines`. Hadoop user must have `Write` permissions. The folder will automatically be created if it does not exist.
       * **HA Namenodes**: If the hostname in the above URL refers to a logical HA cluster, specify the actual namenodes here, in the format:`Hostname1:Port, Hostname2:Port`.
     - **If storage type is NFS**:
       Path: The path could be `/oracle/spark-deploy`.

       > **Note:**
       > /oracle should exist and spark-deploy will automatically be created if it does not exist. You will need `Write` permissions on the /oracle directory.
5. **Hadoop Authentication:**
   - **Simple** authentication credentials:
     - **Protection Policy**: Select a protection policy from the drop-down list. This value should match the value on the cluster.
     - **Username**: Enter the user account to use for submitting Spark pipelines. This user must have read-write access to the Path specified above.
   - **Kerberos** authentication credentials:
     - **Protection Policy**: Select a protection policy from the drop-down list. This value should match the value on the cluster.

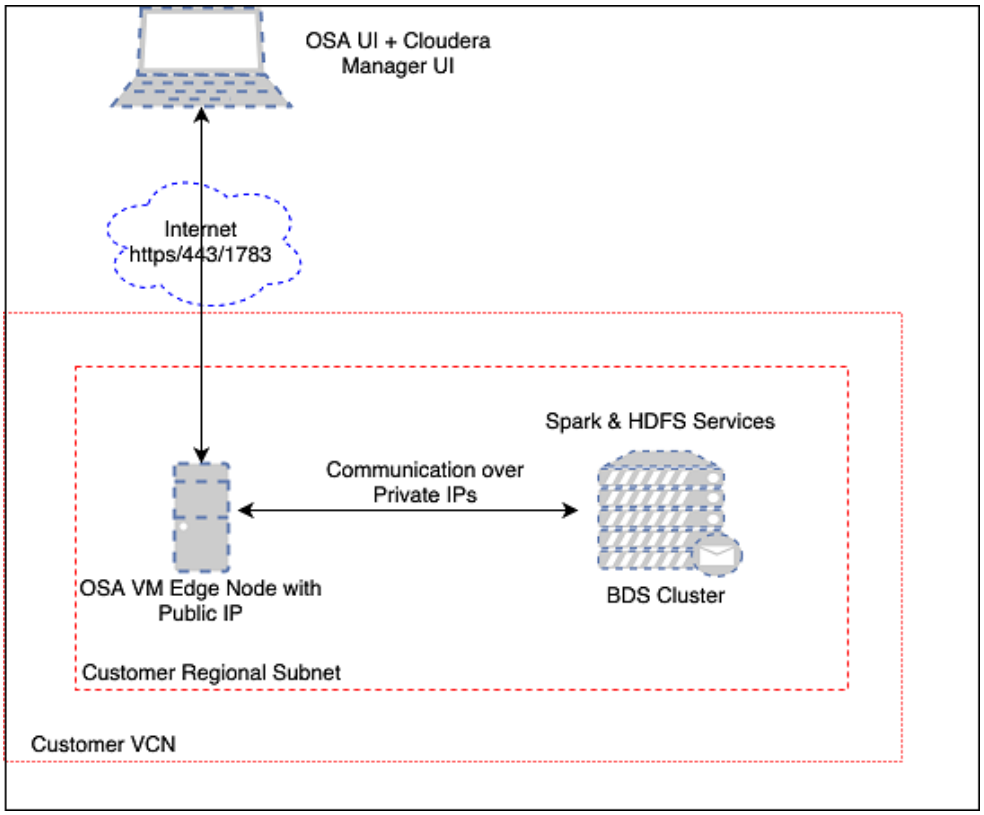
- **Kerberos Realm**: Enter the domain on which Kerberos authenticates a user, host, or service. This value is in the `krb5.conf file`.
- **Kerberos KDC**: Enter the server on which the Key Distribution Center is running. This value is in the `krb5.conf file`.
- **Principal**: Enter the GGSA service principal that is used to authenticate the GGSA web application against Hadoop cluster, for application deployment. This user should be the owner of the folder used to deploy the GGSA application in HDFS. You have to create this user in the yarn node manager as well.
- **Keytab**: Enter the keytab pertaining to GGSA service principal.
- **Yarn Resource Manager Principal**: Enter the yarn principal. When Hadoop cluster is configured with Kerberos, principals for hadoop services like hdfs, https, and yarn are created as well.

6. **Yarn master console port**: Enter the port on which the Yarn master console runs. The default port is `8088`.
7. Click **Save**.

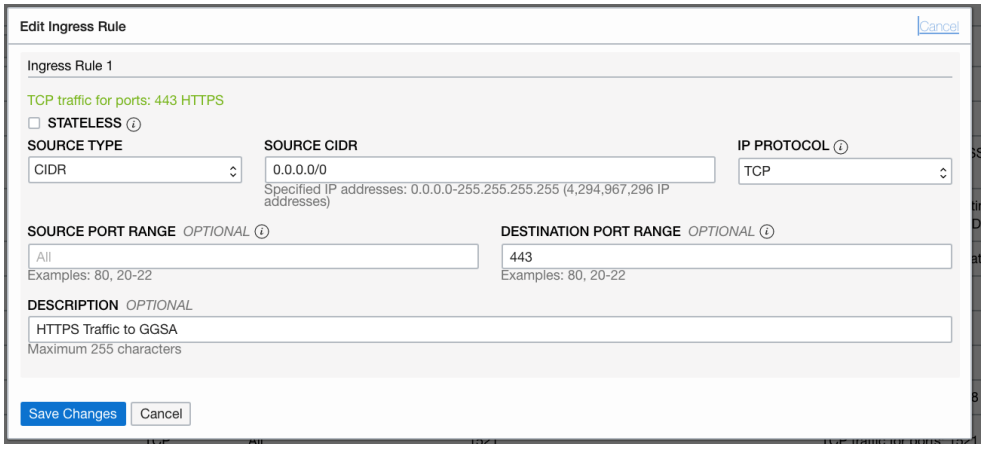
3.1.1.2.3 Configuring for OCI Big Data Service

3.1.1.2.3.1 Topology

In this example, the marketplace GGSA instance and the BDS cluster are in the same regional subnet. The GGSA instance is accessed using its public IP address. The GoldenGate Stream Analytics instance seconds as a Bastion Host to your BDS cluster. Once you ssh to GGSA instance, you can ssh to all your BDS nodes, by copying your ssh private key to GGSA instance.



The default security list for the Regional Subnet must allow bidirectional traffic to edge/OSA node so create a stateful rule for destination port 443. Also create a similar Ingress rule for port 7183 to access the BDS Cloudera Manager via the OSA edge node. An example Ingress rule is shown below.



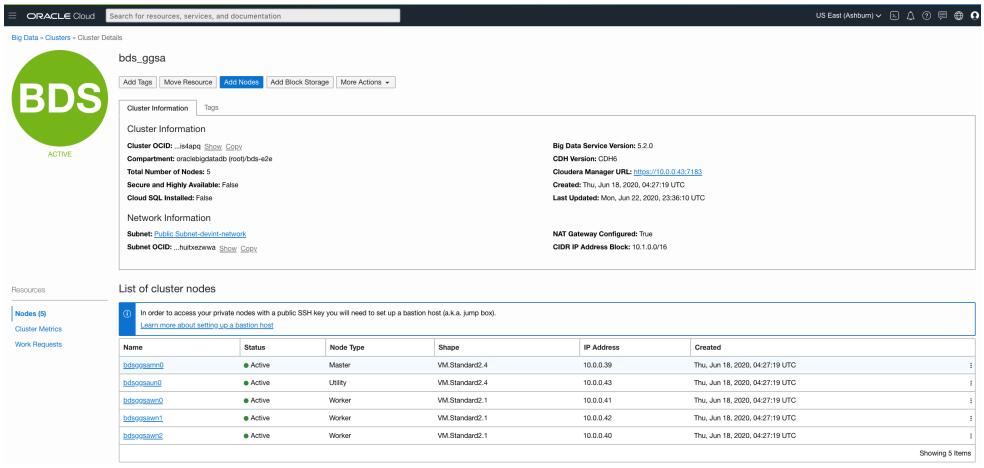
You will also be able to access your Cloudera Manager using the same public IP of GGSA instance by following steps below. Please note this is optional.SSH to GGSA box and run the following port forward commands so you can access the Cloudera Manager via the GGSA instance:

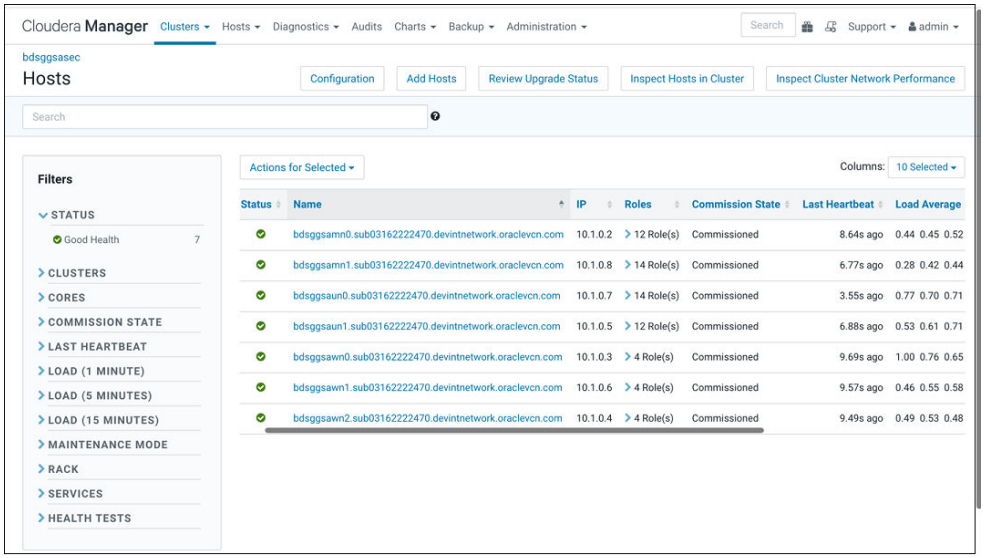
```
sudo firewall-cmd --add-forward-port=port=7183:proto=tcp:toaddr=<IP address of Utility Node running the Cloudera Manager console>

sudo firewall-cmd --runtime-to-permanent

sudo sysctl net.ipv4.ip_forward=1

sudo iptables -t nat -A PREROUTING -p tcp --dport 7183 -j DNAT --to-destination <IP address of Utility Node running the Cloudera Manager console>:7183

sudo iptables -t nat -A POSTROUTING -j MASQUERADE
```

You should now be able to access the Cloudera Manager using the URL `https://<Public IP of GGSA>:7183`.

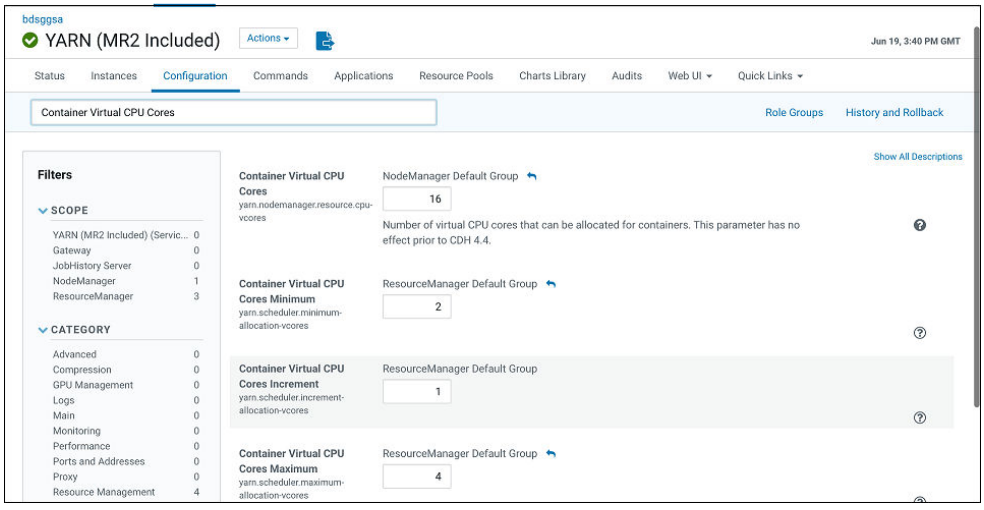
3.1.1.2.3.2 Prerequisites

1. Retrieve IP addresses of BDS cluster nodes from OCI console as shown in screenshot below. Alternatively, you can get the FQDN for BDS nodes from the Cloudera Manager as shown below.





2. Reconfigure YARN virtual cores using Cloudera Manager as shown below. This will allow many pipelines to run in the cluster and not be bound by actual physical cores.



- **Container Virtual CPU Cores**: This is the total virtual CPU cores available to YARN Node Manager for allocation to Containers. Please note this is not limited by physical cores and you can set this to a high number, say 32 even for VM standard 2.1.

- **Container Virtual CPU Cores Minimum**: This is the minimum vcores that will be allocated by YARN scheduler to a Container. Please set this to 2 since CQL engine is a long-running task and will require a dedicated vcore.

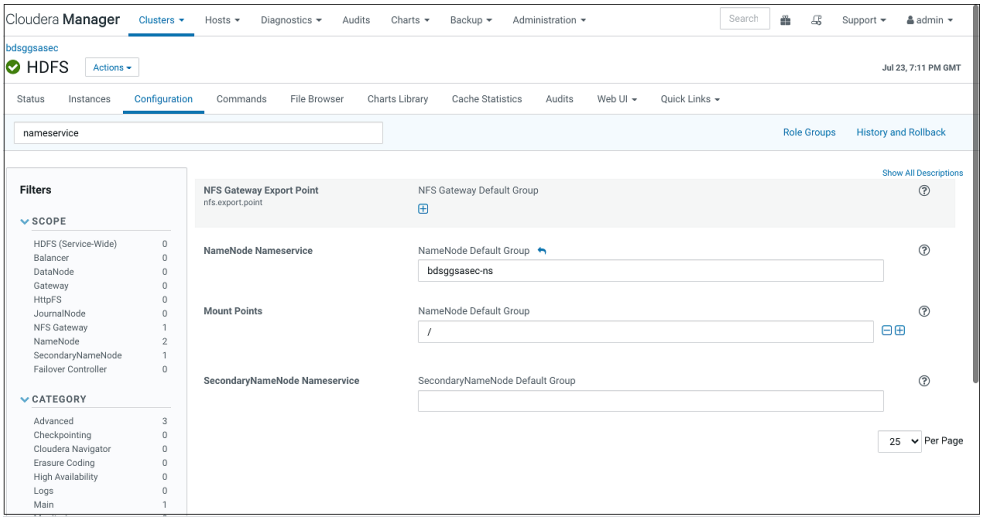
  Container Virtual CPU Cores Maximum: This is the maximum vcores that will be allocated by YARN scheduler to a Container. Please set this to a number higher than 2 say 4.

> **Note:**
>
> This change will require a restart of the YARN cluster from Cloudera Manager.

3.1.1.2.3.3 Configuring for Kerberized Big Data Service Runtime

In GGSA **System Settings** dialog, configure the following:

1. Set Kafka Zookeeper Connection to Private IP of the OSA node or list of Brokers that have been configured for GGSA's internal usage. For example, 10.0.0.59:2181.
2. Set **Runtime Server** to Yarn.
3. Set Yarn Resource Manager URL to Private IPs of all master nodes starting with the one running active Yarn Resource Manager. For example, `bdsggsamn1.sub03162222470.devintnetwork.oraclevcn.com`, `bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com`.
4. Set storage type to **HDFS**.
5. Set Path to <NameNode Nameservice><HDFS Path>. For example, `bdsggsasec-ns/user/oracle/ggsapipelines`. The path will automatically be created if it does not exist but the Hadoop User must have write permissions. NameNode Nameservice can be obtained from HDFS configuration as shown in the screenshot below. Use search string nameservice in the search field.



6. Set HA Namenode to Private IPs or hostnames of all master nodes (comma separated), starting with the one running active NameNode server. In the next version of GGSA, the ordering will not be needed. For example, `bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com`, `bdsggsamn1.sub03162222470.devintnetwork.oraclevcn.com`.
7. Set Yarn Master Console port to 8088 or as configured in BDS.

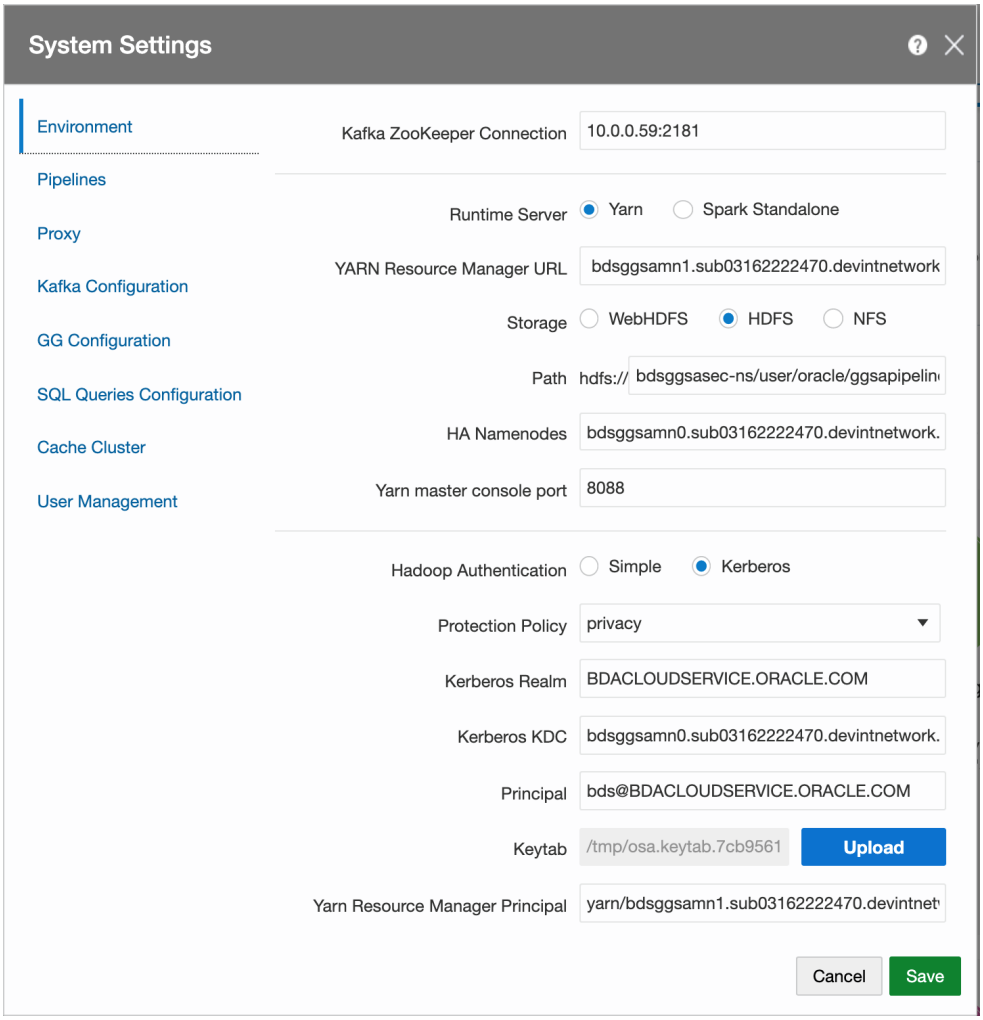
8. Set Hadoop authentication to Kerberos.
9. Set protection policy to privacy. Please note this should match the value in HDFS configuration property *hadoop.rpc.protection*.
10. Set Kerberos Realm to BDACLOUDSERVICE.ORACLE.COM.
11. Set Kerberos KDC to private IP or hostname of BDS master node 0. For example, `bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com`.
12. Set principal to bds@BDACLOUDSERVICE.ORACLE.COM. See this documentation to create a Kerberos principal (e.g. bds) and add it to hadoop admin group, starting with step **Connect to Cluster's First Master Node** and through the step **Update HDFS Supergroup**.

    > **Note:**
    >
    > You can hop/ssh to the master node using your GGSA node as the Bastion. You will need your ssh private key to be available on GGSA node though. Restart your BDS cluster as instructed in the documentation.
    >
    > ```
    > [opc@bdsggsa ~]$ ssh -i id_rsa_private_key
    > opc@bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com
    > ```

13. Make sure the newly created principal is added to Kerberos keytab file on the master node as shown:
    bdsmn0 # sudo **kadmin.local**

    kadmin.local: **ktadd -k /etc/krb5.keytab bds@BDACLOUDSERVICE.ORACLE.COM**
14. Fetch the keytab file using sftp and set Keytab field in system settings by uploading the same.
15. Set Yarn Resource Manager principal which should be in the format `yarn/<FQDN of BDS MasterNode running Active Resource Manager>@KerberosRealm`. For example, `yarn/bdsggsamn1.sub03162222470.devintnetwork.oraclevcn.com@BDACLOUDSERVICE.ORACLE.COM`.

Sample System Settings Screen:

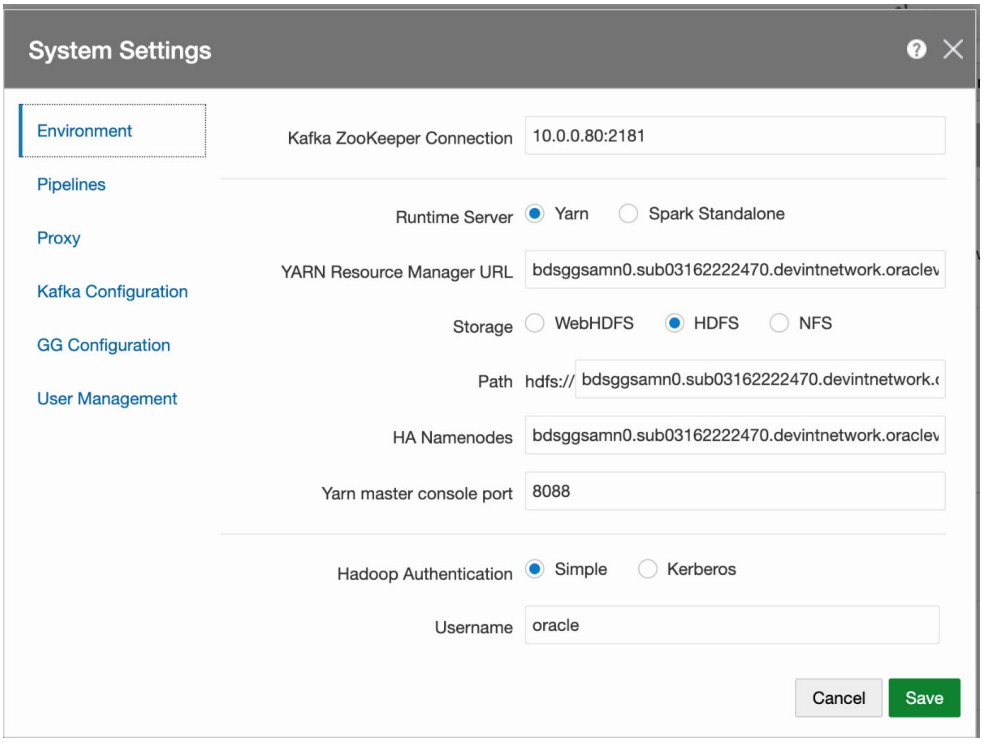


3.1.1.2.3.4 Configuring for Non Kerberized Big Data Service Runtime

In GGSA **System Settings**dialog, configure the following:

1. Set Kafka Zookeeper Connection to Private IP of the OSA node or list of Brokers that have been configured for GGSA's internal usage. For example, 10.0.0.59:2181
2. Set **Runtime Server** to Yarn.
3. Set Yarn Resource Manager URL to Private IP or Hostname of the BDS Master node. For example, `bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com`.
4. Set storage type to **HDFS**.
5. Set Path to <PrivateIP or Host Of Master><HDFS Path>. For example, `10.x.x.x/user/oracle/ggsapipelines` or `bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com/user/oracle/ggsapipelines` .
6. Set HA Namenode to Private IP or Hostname of the BDS Master node. For example, `bdsggsamn0.sub03162222470.devintnetwork.oraclevcn.com`.
7. Set Yarn Master Console port to 8088 or as configured in BDS
8. Set Hadoop authentication to Simple and leave Hadoop protection policy at authentication if available

9. Set username to **oracle**.
10. Click **Save**.



3.1.2 Optional Configurations

The following configurations are optional:

3.1.2.1 Configuring Pipeline Preferences

To set the pipeline configurations:

1. Click the user name in the top right corner of the screen and Select **System Settings** from the drop-down list.
2. Click **Pipelines** and set the following configurations:
   - **Batch Duration**: Set the default duration of the batch for each pipeline.
   - **Executor Count**: Set the default number of executors per pipeline.
   - **Cores per Executor**: Set the default number of cores. A minimum value of 2 is required.
   - **Executor Memory**: Set the default allocated memory for each executor instance in megabytes.
   - **Cores per Driver**: Set the default number of cores. A minimum value of 1 is required.
   - **Driver Memory**: Set the default allocated memory per driver instance in megabytes.
   - **Log Level**: Select a log level for unpublished pipelines, from the drop-down list.

> **Note:**
>
> Reset the default log level of draft pipelines to WARNING, and of published pipelines, to ERROR.

- **High Availability**: Set the default HA value for each pipeline
- **Pipeline Topic Retention**: Set retention period for intermediate stage topics, in milliseconds. Default is one hour.
- **Enable Pipeline Topics**: Select this option to enable creation of intermediate kafka topics. It is selected by default.
- **Input Topics Offset**: Select the Kafka topic offset value from the drop-down list. The default value is **latest**.

> **Note:**
>
> When you publish the pipeline for the first time, the input stream is read based on the offset value you have selected in this drop-down list. On a subsequent publish, the value you have selected here is not considered, and the input stream is read from where it was last left off.

- **Reset Offset**: Select this option to read the input stream based on the offset value selected in the **Input Topics Offset** drop-down list.

> **Note:**
>
> If you are using two Kafka streams as an input to the pipeline, the offset is not preserved and the pipeline starts from the current timestamp. With a single stream the offset is maintained and the pipeline can read from the previous state of it.

3. Click **Save**.

3.1.2.2 Configuring Network Proxy

To set your Network Proxy:

1. Click the user name in the top right corner of the screen and Select **System Settings** from the drop-down list.
2. Click **Proxy**, and set the following configurations:
   - **HTTP Proxy**: Set HTTP proxy server to access any resource, for example a REST target, outside your network.
   - **Port**: Enter the port of the HTTP proxy server. Enter a number between 0 and 65,535.
   - **Use this proxy server for all protocols**: Select the option to use a single proxy server.
   - **No proxy for**: Set a list of hosts that should be reached directly, bypassing the proxy. This is a list of patterns separated by the delimiter `|`. The patterns can start or end with a `*` for wildcards.

3. Click **Save**.

3.1.2.3 Configuring Kafka Preferences

To set your Kafka preferences:

1. Click the user name in the top right corner of the screen and Select **System Settings** from the drop-down list.
2. Click **Kafka Configuration**, and set the following configurations:
   - **Partition count per Topic**: By default, GGSA creates a Kafka topic with partition count of 3 and a replication factor of 1. Increasing the partition count to higher than 3 will increase throughput when consuming data from topic. For Kafka targets, you can select a column as partitioning key.
   - **Replication factor per Topic**: Increasing the replication factor increases the number of copies of your data thereby making it highly available.
3. Click **Save**.

3.1.2.4 Configuring GG Preferences

To set your GG preferences:

1. Click the user name in the top right corner of the screen and Select **System Settings** from the drop-down list.
2. Click **GG Configuration**, and set the following configurations:
   - **Begin**: By default, GGSA reads the Goldengate trail file from the latest offset (Now), but you can change it to read from the beginning.
3. Click **Save**.

3.1.2.5 Configuring SQL Preferences

When displaying tables and objects in Database reference or target, GGSA by default only shows the objects that are owned by the user. If you want the system to display all objects the user has access to, regardless of the owner, enter your own SQL query here.

To set your SQL preferences:

1. Click the user name in the top right corner of the screen and Select **System Settings** from the drop-down list.
2. Click **SQL Queries Configurations**, and set the following configurations:
   - **Oracle DB Reference and Target**:
     Enter a SQL query to fetch the table column details, for all the tables to be used for creating references or targets. The SELECT query must contain the following attributes for each column: `table_name`, `column_name`, `data_type`, `data_length`, `data_precision`, `data_scale`, `nullable` and `data_default`. `WHERE` statements can be used to filter the tables listed.

     Example: `SELECT table_name, column_name, data_type, data_length, data_precision, data_scale, nullable, data_default FROM all_tab_columns WHERE table_name LIKE DEMO ORDER BY table_name`. In this example all the tables with names starting with DEMO will be displayed when using references or targets.
   - **Oracle DB Geofence**: Enter the query for geofence tables and columns details.

Enter a SQL query to fetch the table column details, for all the tables to be used for creating Geofences. The SELECT query must contain the following attributes for each column: `table_name, column_name, data_type, data_length, data_precision, data_scale`. The tables listed in for geofence must have at least one column of the `SDO_GEOMETRY` type. Hence the following where clause is mandatory: `WHERE table_name IN (SELECT TABLE_NAME FROM all_tab_columns WHERE data_type='SDO_GEOMETRY'`.

Example: `SELECT table_name, column_name, data_type, data_length, data_precision, data_scale FROM all_tab_columns WHERE table_name IN (SELECT TABLE_NAME FROM all_tab_columns WHERE data_type='SDO_GEOMETRY') AND table_name LIKE GEO ORDER BY table_name desc`. In this example, all the tables with names starting with GEO will be listed in the descending order of the table name.

3. Click **Save**.

3.1.2.6 Changing Spark Work Directory

To change Spark work directory:

1. Stop Spark service:
   a. `sudo systemctl stop spark-slave.service`
   b. `sudo systemctl stop spark-master.service`
2. Create work directory under u02:
   a. Navigate to `/u02`
   b. `sudo mkdir spark`. Here `spark` is the work folder name, you can create folder with the name of your choice.
   c. `chmod 777 spark`, to change permission.
3. Edit `spark-env.sh`
   a. Navigate to `SPARK_HOME/conf`, and edit `spark-env.sh`
   b. Add `SPARK_WORKER_DIR=/u02/spark` at the end of the file `spark-env.sh` to point to newly created folder under `/u02`
4. Start Spark service:
   a. `sudo systemctl start spark-master.service`
   b. `sudo systemctl start spark-slave.service`

   You will see the application and driver data (files and logs) under `/u02/spark` when you publish the pipeline again.

3.1.2.7 Changing Spark Log Rollover based on Time

To change Spark log rollover based on time:

1. Stop Spark service:
   a. `sudo systemctl stop spark-slave.service`
   b. `sudo systemctl stop spark-master.service`
2. Edit `spark-env.sh`

a. Navigate to `SPARK_HOME/conf`, and comment or delete `SPARK_WORKER_OPTS` variable and its value.

3. Edit `spark-defaults.conf` by adding the below lines:
   - `spark.executor.logs.rolling.maxRetainedFiles 7`
   - `spark.executor.logs.rolling.strategy time`
   - `spark.executor.logs.rolling.time.interval minutely`

> **Note:**
>
> You can change `spark.executor.logs.rolling.time` to `daily`, `hourly`, `minutely`. This is to enable log rollover based on time.

4. Start Spark service:
   a. `sudo systemctl start spark-master.service`
   b. `sudo systemctl start spark-slave.service`

   You will see the application and driver data (files and logs) under `/u02/spark` when you publish the pipeline again.

3.2 Configure Users

Managing Users

Configuring User Preferences

3.2.1 Managing Users

After you install GoldenGate Stream Analytics, it is important to authenticate and manage users who use the application.

User details are stored in a database. When you create a GGSA schema at the time of installation, the following database tables are populated with one record in each table:

- `osa_users` — table containing the users
- `osa_user_roles` — table containing the user names and their associated roles

When you execute a query to pull in all the data from the `osa_users` table, you can see the following:

```
select * from osa_users;


+----+----------+--------------------------------------+
| id | username | pwd                                  |
+----+----------+--------------------------------------+
|  1 | osaadmin | MD5:201f00b5ca5d65a1c118e5e32431514c |
+----+----------+--------------------------------------+
```

where `osaadmin` is the pre-configured user along with the encrypted password.

When you execute a query to pull in all the data from the `osa_user_roles` table, you can see the following:

```
select * from osa_user_roles;


+---------+---------+
| user_id | role_id |
+---------+---------+
|       1 |       1 |
+---------+---------+
```

where `role_id` of value `1` indicates that the user is an administrator.

3.2.1.1 Adding Users

Though you can continue using Oracle GoldenGate Stream Analytics through the pre-configured user, it is a best practice to create your own users and delete the default pre-configured user.

When you add a user, it is highly recommended, though not mandatory, to obfuscate or encrypt the password. You can use the utility provided with the application server (Jetty) to encrypt the password.

Add Users Through User Interface

You can add/create users through the Oracle GoldenGate Stream Analytics application user interface.

To add a new user:

1. Go to **System Settings**.
2. Under the **User Management** tab, click **Add user**.
3. Enter details in the **Username**, **Password**, and **Confirm Password** fields.
4. click **Create**.
   You can see the new user along with the predefined user in the list of available users.

Repeat these steps for as many users as you need, based on your requirement. If you try a user with the same name as that of an existing user, an error `A user profile with the user name <username> already exists. Please specify another user name.` pops up.

Add Users Through Code

To add a new user:

1. Open a terminal and navigate to `OSA-19.1.0.0.*.*`.

   This is top-level folder in the folder where you have extracted your zip installer.

> **Note:**
>
> Replace *.* with the current version of GGSA.

2. Execute the following command:

```
java -cp ./lib/jetty-util-9.4.17.v20190418.jar
org.eclipse.jetty.util.security.Password NewUser <password>
```

where `NewUser` is the name of the user and `<password>` is the password that you want to obfuscate or encrypt.

You will see a similar screen on your terminal:

```
2018-02-22 17:26:31.259:INFO::main: Logging initialized @100ms to
org.eclipse.jetty.util.log.StdErrLog <password>
OBF:1pbi1vn61unn1z7e1vu91ytc1o4u1o5s1yta1vv11z7o1uob1vnw1pcg
MD5:58d613129c5e71de57ee3f44c5ce16bc
CRYPT:NegJERR2H/a1M
```

For more information about running the password utility, see Configuring Secure Password.

3. Connect to the database using the database user credentials that you have configured in `/osa-base/etc/jetty-osa-datasource.xml`.

4. Insert a record into the `osa_users` table using any one of the following commands:

```
insert into osa_users (id,username,pwd) values
(2,'NewUser','OBF:1pbi1vn61unn1z7e1vu91ytc1o4u1o5s1yta1vv11z7o1uob1vnw1pcg'
);
```

or

```
insert into osa_users (id,username,pwd) values
(2,'NewUser','MD5:58d613129c5e71de57ee3f44c5ce16bc');
```

or

```
insert into osa_users (id,username,pwd) values
(2,'NewUser','CRYPT:NegJERR2H/a1M');
```

5. Insert a record into the `osa_user_roles` table using the following command:

```
insert into osa_user_roles (user_id, role_id) values (2,1);
```

> **Important:**
>
> Currently, Oracle GoldenGate Stream Analytics supports only one user role, i.e, the administrator role. So the `role_id` value must always be `1`.

You can now login to Oracle GoldenGate Stream Analytics as `NewUser` using `<password>`. Repeat these steps to create as many users as you require.

3.2.1.2 Changing Password

Change Password Through User Interface

To change a user password:

1. Go to **System Settings**.
2. Click the **User Management** tab.
3. Click **Change Password** next to the required user within the list of available users and then provide a value for the new password and click **Save**.

   Passwords are stored in MD5 hash form.

Change Password Using Code

To change a user password:

1. Obfuscate or encrypt the new password for the user using the utility provided with the application server (Jetty).
2. Update the relevant record in the `osa_users` table. For example:

   ```
   update osa_users set pwd='CRYPT:NesIZC3VkNGN2' where username='NewUser';
   ```

   This command updates the password for the `NewUser`.

Remember to use your updated password the next time you login with `NewUser`.

3.2.1.3 Removing Users

You may want to remove users when you no longer need them.

Before you proceed to delete any user, make a note of the following:

- If a user who owns draft pipelines is deleted, then the pipelines are either migrated to the current user or deleted, based on the selection you make at the time of deletion.
- If you attempt to delete yourself, all your draft pipelines are deleted after you confirm. The current user session is invalidated and you will be signed out of the application immediately.

Delete Users Through User Interface

To delete a user:

1. Go to **System Settings**.
2. Click the **User Management** tab.
3. Click **Delete** next to the required user within the list of available users and then click **OK** within the confirmation dialog.

Delete Users Through Code

To delete a user:

1. Execute the following command from SQLPLUS or SQLDeveloper tools to remove a user:

   ```
   delete from osa_users where id=2;
   ```

   This command deletes the user with the id value as 2, i.e, the second user in the database.
2. Execute the following command to delete the user role corresponding to the user in the above step:

   ```
   delete from osa_user_roles where user_id=2;
   ```

3.2.1.4 Configuring LDAP for User Authentication and Management

Oracle GoldenGate Stream Analytics makes use of the LDAP support for Jetty. The Lightweight Directory Access Protocol (LDAP) is an open source application accepted across various industries. This application protocol is used for obtaining and maintaining distributed directory information services over a network using an Internet Protocol (IP). With this feature, you can use the directory information services for user authentication and management. To use Microsoft directory services, set up a Microsoft Active Directory.

The user authentication and management can be through either internal LDAP or external LDAP.

For internal LDAP use the following command to create an LDAP service with default administrative access:

```
docker run --name LDAP-service --hostname LDAP-service -p 389:389 --detach
osixia/openLDAP:1.2.1
```

3.2.1.4.1 Setting Up LDAP

To use LDAP for user authentication:

1. Update etc/override-web.xml to specify ldap role (EMPLOYEE for oracle ldap) and realm as `osa-realm-ldap`.
   In case you need to switch back to data source from LDAP, you can update `etc/override-web.xml` to specify role (admin) and realm as `osa-realm-ds`. By changing realm in `etc/override-web.xml`, you switch between LDAP and data source. You can keep `ldap-login.conf` configured to retain LDAP configuration and can toggle between LDAP and data source by just changing `override-web.xml` file.
2. Update `/osa-base/etc/LDAP-login.conf` as per LDAP user/group settings. For example:
   For User role:

   ```
   osa-demo-LDAP {
           org.eclipse.jetty.jaas.spi.LDAPLoginModule required
               debug="true"
               contextFactory="com.sun.jndi.LDAP.LDAPCtxFactory"
               hostname=<hostname> <!-- hostname of LDAP -->
               port="389"
               authenticationMethod="simple"
               forceBindingLogin="true"
               userBaseDn="l=emea,dc=oracle,dc=com"
               userRdnAttribute="uid"
   ```

```
            userIdAttribute="mail"
            userPasswordAttribute="userPassword"
            userObjectClass="person"
            roleBaseDn="l=emea,dc=oracle,dc=com"
            roleNameAttribute="opn_access_level"
            roleMemberAttribute="targetdn"
            roleObjectClass="person";
        };
```

For Employee role:

```
osa-demo-LDAP {
        org.eclipse.jetty.jaas.spi.LDAPLoginModule required
            debug="true"
            contextFactory="com.sun.jndi.LDAP.LDAPCtxFactory"
            hostname=<hostname> <!-- hostname of LDAP -->
            port="389"
            authenticationMethod="simple"
            forceBindingLogin="true"
            userBaseDn="l=amer,dc=oracle,dc=com"
            userRdnAttribute="uid"
            userIdAttribute="mail"
            userPasswordAttribute="userPassword"
            userObjectClass="person"
            roleBaseDn="l=amer,dc=oracle,dc=com"
            roleNameAttribute="employeetype"
            roleMemberAttribute="targetdn"
            roleObjectClass="organizationalPerson";
        };
```

Remember to change userBaseDn and RoleBaseDn as per your locality name.

If in America:

```
userBaseDn="l=amer,dc=oracle,dc=com"
    roleBaseDn="l=amer,dc=oracle,dc=com"
```

If in Asia Pacific:

```
userBaseDn="l=apac,dc=oracle,dc=com"
    roleBaseDn="l=apac,dc=oracle,dc=com"
```

If in Europe:

```
userBaseDn="l=emea,dc=oracle,dc=com"
    roleBaseDn="l=emea,dc=oracle,dc=com"
```

3. (Re) start the application.

3.2.1.4.2 Setting Up Microsoft Active Directory

To setup Microsoft Active Directory 2016:

1. Ensure that role name is updated in the web.xml file located at `/osa-base/etc/override-web.xml`:

```
<auth-constraint>
        <role-name>developer</role-name>
    </auth-constraint>
```

2. Update `/osa-base/etc/ldap-login.conf` as per LDAP user/group settings. For example:

```
osa_demo_ldap {
        org.eclipse.jetty.jaas.spi.LdapLoginModule required
            debug="true"
            contextFactory="com.sun.jndi.ldap.LdapCtxFactory"
            hostname=<hostname> <!-- this is the active directory server
hostname -->
            port="389" <!-- this is the active directory server port -->
            bindDn="CN=Administrator,CN=Users,DC=corp,DC=oradev,DC=com"
            bindPassword=<password> <!-- If the active directory server
allows anonymous login, no need to provide bindDn and bindPassword. Else,
set the active directory server admin DN and password -->
            authenticationMethod="simple" <!-- if the active directory
server allows anonymous login then set to 'none' otherwise set it to
'simple'-->
            forceBindingLogin="true"
            userBaseDn="l=amer,dc=oracle,dc=com" <!-- user attributes as
per user setup in active directory server -->
            userRdnAttribute="uid" <!-- user attributes as per user setup
in active directory server -->
            userIdAttribute="mail" <!-- user attributes as per user setup
in active directory server -->
            userPasswordAttribute="userPassword" <!-- user attributes as
per user setup in active directory server -->
            userObjectClass="person" <!-- user attributes as per user
setup in active directory server -->
            roleBaseDn="l=amer,dc=oracle,dc=com" <!-- role (group)
attributes as per user setup in active directory server -->
            roleNameAttribute="opn_access_level" <!-- role (group)
attributes as per user setup in active directory server -->
            roleMemberAttribute="targetdn" <!-- role (group) attributes as
per user setup in active directory server -->
            roleObjectClass="person"; <!-- role (group) attributes as per
user setup in active directory server -->
        };
```

3.2.2 Configuring User Preferences

To set or update user preferences:

1. Click the user name at the top right corner of the screen.
2. Select **Preferences** from the drop-down list.
3. Click **General**, to set the following general preferences:
    - **Start Page**: Select a start page from the drop-down list.

4. Click **Notifications**, to set the following notification preferences:
   - **Show Information Notifications**: Select this option if you want the information notifications to appear in the pipeline. This option is selected by default.
   - **Information Notification duration (in seconds)**: Set the number of seconds for which the notifications appear. The default value is 5.
5. Click **Catalog**, to set the following Catalog page settings:
   - **Default Sorting Column**: Select the column by which you want the columns to be sorted. This value will be used as the default for all columns until you change the value again.
   - **Default Page Size**: Select the value to be used as the default page size. Based on the value selected, the number of records that appear on a page vary. This value will be used as the default for all pages until you change the value again.
6. Click **Pipeline**, to set the following pipeline preferences:
   - Select **Yes**, to display the User Assistance text for the pipelines in the Pipeline Editor.
   - Click **Live Output Stream**, to set the default table size, for the data in the **Live Output Stream** table, of a pipeline.
   - Click **Timestamp**, to set the following timestamp function and format preferences:
     - **Timestamp Function**: Select a value from the drop-down list.
     - **Timestamp Format**: Select a format to display the `timestamp` type fields.
7. Click **Map**, to select a tile layer from the drop-down list.
8. Click **Save**.

4
Manage

4.1 Connections

4.1.1 Create Connections

A connection is a collection of metadata (such as URLs, credentials, etc.,) required to connect to an external system. A connection is the basis for creation of sources (Streams, References, or Geo Fences) and Targets. You can reuse connections to create and access multiple sources, targets, or both, in the same system. For example, different Kafka topics in the same Kafka cluster, or different database tables in the same Oracle database.

GGSA supports the following connection types:

Connection Types	Supported Artifacts
ADW or ATP	• Reference • Target
AWS S3	• Target
Coherence	• Reference • Target
Druid	• Target
Elasticsearch	• Target
GoldenGate	• Stream (Microservices only)
HBase	• Target
HDFS	• Target
Hive	• Target
Ignite Cache	• Reference • Target

Connection Types	Supported Artifacts
JMS	• Stream • Target
Kafka	• Stream • Target
Microsoft Azure Data Lake-Gen2	• Target
MongoDB	• Target
MySQL DB	• Reference
OCI	• Target
ONS	• Target
Oracle AQ	• Stream
Oracle DB	• Reference • Target
OSS	• Stream • Target

4.1.1.1 Creating a Connection to ADW or ATP

To create a connection to Autonomous Data Warehouse (ADW) and Autonomous Transaction Processing (ATP):

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Oracle Database** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Connect using**: Select `Wallet` from the drop-down list.
   - **Service name/SID**: Enter the name of the Service, SID, or Wallet that you want to connect to.
   - **Wallet Archive** > **Upload File**: Upload the archive wallet file stored in your local machine.

     If you do not have the wallet file, you can download it from the OCI Autonomous Database console.

     After you have successfully uploaded the wallet file, all the database services configured under the wallet's tnsnames.ora will be displayed in the service list. Select one of the services based on your requirements.
   - **Username**: Enter the database account user name.

- **Password**: Enter the password for your database account.

> **Note:**
>
> Special characters in the password are treated as wildcard patterns, causing the connection to fail . If your password contains special characters, enclose the password within double quotes (the double quotes with ASCII Value 34).

6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.2 Creating a Connection to AWS S3

Prerequisites:

- An AWS account with S3 access, bucket creation and write data permission. See AWS Account Creation.
- A Client ID and Client Secret to access AWS programmatically. See Programmatic Access.

To create a connection to AWS S3:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **AWS S3** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Client ID**: Enter the Access Key ID obtained from AWS.
   - **Client Secret**: Enter the Secret access key obtained from AWS.
   - **Region**: Region in which the S3 Bucket is already created, or in which you want to create a new bucket. See Regions and Endpoints.
6. Click **Test Connection**, to ensure credentials are correct and to download the third party libraries required to connect to AWS S3.
   The download will take some time to complete, because these jars will be downloaded from maven repository, for the first time. You can track the progress in jetty logs. For the subsequent Test Connection or OSA pipeline deployment, third party jars will be available locally.
7. Click **Save.**

4.1.1.3 Creating a Connection to Coherence

To create a connection to a Coherence:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Coherence** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Server Url(s)**: Enter the server url(s) of the coherence cache server(s) in the following format:
     `Host1:Port1,Host2:Port2`

     The port is the Coherence Extend Proxy Services TCP/IP Server Socket port.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.4 Creating a Connection to Druid

To create a connection to Druid:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Druid** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Zookeepers**: Enter the zookeeper URL.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.5 Creating a Connection to Elasticsearch

To create a connection to ElasticSearch:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Elastic Search** from the submenu.

3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Server Url(s)**:Enter the server url(s) for the ElasticSearch server, in the format `host1:port1, host2:port2`.
   - **Authentication Type**
     - **None**: Select this option to disable security for the ElasticSearch cluster.
     - **Basic**: Select this option to secure the ElasticSearch cluster with Basic Authentication.
       * **User Name**: Set the authentication username.
       * **Password**: Set the authentication password.
     - **SSL**: Select this option to secure and enable ElasticSearch cluster communication over HTTPS.
       * **User Name**: Set the authentication username
       * **Password**: Set the authentication password.
       * **Trust Store**: Upload the truststore file.
       * **Truststore Password**: Set the truststore password.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.6 Creating a Connection to GoldenGate

To create a connection to GoldenGate:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **GoldenGate** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:

- **Service Manager Host**: Enter the name or IP address of the GoldenGate Service Manager.
- **Service Manager Port**: Enter the port on the Service Manager, to connect to GoldenGate.

  Set the port to 443 to create a connection to the Goldengate instance running on an OCI Goldengate stack, because you can access only port 443, by default.
- **GG Username**: Enter the username to authenticate the GoldenGate connection.
- **GG Password**: Enter the password for the GoldenGate connection.
- **Is SSL?**: Select this option if the Goldengate instance uses a SSL based connection
- **Is GG Marketplace?**: Select this option if the GG instance is running on OCI Marketplace Goldengate stack.

6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.7 Creating a Connection to HBase

To create a connection to HBase:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **HBase** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **hbase-site.xml**: Upload the hbase-site. xml file. This file is used to create connection to zookeeper that manages the Hbase cluster. This file must contain zookeeper IP and port.
   - **Authentication Type**:
     - **None**: Select this option if the Hbase server does not have any authentication enabled.
     - **kerberos**: Select this option if the kerberos authentication is enabled for the HBase server.
   - **Kerberos Principal**: Enter the Kerberos principal for the Hbase service.
   - **Kerberos KeyTab**: Enter the kerberos keytab for the Hbase service. The keytab file has the `.keytab` extension.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.8 Creating a Connection to HDFS

To create a connection to HDFS:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **HDFS** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **core-site.xml**: Upload the `core-site.xml` file with `fs.defaultFS`, `fs.default.name`, `hadoop.security.authentication`, and `fs.AbstractFileSystem.hdfs.impl` properties.
   - **hdfs-site.xml** : Set client-specific properties. This is an optional field.
   - **Use Kerberos**: To use a kerberized cluster, select this option to enable Kerberos principal and Kerberos keytab. Provide the following details:
     - **Kerberos KDC**
     - **Kerberos Realm**
     - **Kerberos Principal**: Set HDFS service principal.
     - **Kerberos Key Tab**: Upload the keytab file that has HDFS service principal
6. Click **Test Connection**, to ensure that you have successfully created a connection, and to download the third-party libraries required to connect to HDFS. The libraries are also used by targets to write to HDFS paths.

   > **Note:**
   >
   > Retain the `core-site.xml` and `hdfs-site.xml` file names exactly as they are.

7. Click **Save.**

4.1.1.9 Creating a Connection to Hive

To create a connection to Hive:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Hive** from the submenu.
3. Click **Connection**, and select from the drop-down list.
4. On the **Type Properties** screen, enter the following details:

- **Name**: Enter a unique name for the connection. This is a mandatory field.
- **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
- **Description**
- **Tags**
- **Connection Type**: The selected connection is displayed.

5. Click **Next**.
6. On the **Connection Details** screen, enter the following details:
   - **Core Site XML**: Upload the core-site.xml file to connect to HDFS, where the files to create external hive table are loaded. This is a mandatory field..
   - **Hdfs Site XML**: Upload the hdfs-site.xml file to connect to HDFS, where the files to create external hive table are loaded. This is an optional field.
   - **Use Kerberos**: In case of a kerberized cluster, you can provide the Kerberos principal and keytab by enabling this option.
   - **Kerberos KDC**: Provide the host having the Key Distribution Center(KDC).
   - **Kerberos Realm**: Provide the kerberos realm.
   - **Kerberos Principal**: Set the Kerberos principal for the hive service.
   - **Kerberos KeyTab**: Upload the kerberos keytab file for the hive service.
   - **Hive JDBC URL**: You can connect to hive database using following jdbc url : jdbc:hive2://host:port/<DB_NAME>. The default port is 10000 and default database is "default".
   - **Hive JDBC Username**: Enter the username to connect to hive database. If you are using the default database, this field can be left blank.
   - **Hive JDBC Password**: Password used to connect to hive database. If you are using the default database, this field can be left blank.
7. Click **Test Connection**, to ensure that you have successfully created a connection, and to download the third-party libraries required to connect to hive database to create an external table.
8. Click **Save.**

4.1.1.10 Creating a connection to Ignite Cache

To create a connection to a Ignite Cache:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Ignite Cache** from the submenu.
3. Click **Connection**, and select from the drop-down list.
4. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**

- **Connection Type**: The selected connection is displayed.

5. Click **Next**.
6. On the **Connection Details** screen, enter the following details:
   - **External cache server url(s)**: Enter the ignite server IP address. This is a comma-separated field.
7. Click **Test Connection**, to ensure that you have successfully created a connection.
8. Click **Save.**

4.1.1.11 Creating a Connection to JMS

To create a connection to JMS:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **JNDI** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **JNDI Provider**: Select WebLogic as the JNDI service provider.
   - **Server Url(s)**: Enter the server url(s) for the JNDI connection, in the format `host1:port1, host2:port2`.
   - **Username**: Enter the user name for authenticating the JNDI connection.
   - **Password**: Enter the password for the JNDI connection.
   - **Jndi Other Properties**: This is not a required field for a WebLogic JMS connection.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.12 Creating a Connection to Kafka

To create a connection to Kafka:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Kafka** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**

- **Tags**
- **Connection Type**: The selected connection is displayed.

4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Use Bootstrap**: Check this box to use a bootstrap based connection.
   - **Zookeepers**: Enter the zookeeper URL. Use this option only if you did not select the **Use Bootstrap** box in the previous step.
   - **Kafka bootstrap**: Enter the bootstrap URL.
   - **SSL**: Check this box to connect to an SSL enabled Kafka cluster.
     - **Truststore Location**: Locate and upload the truststore file. This field is applicable only to connect to an SSL enabled Kafka cluster.
     - **Truststore Password**: Enter the truststore password.
   - **SASL**: Check this box if Kafka broker requires authentication.
     - **User Name**: Enter the SASL username for the Kafka broker.

       You can retrieve this information from the OCI console. This field is enabled only if you have checked the SASL option.
     - **Password**: Enter the SASL password, which is an authentication token that you can generate on the **User Details** page, of the OCI console.

       > **Note:**
       >
       > Copy the authentication token when you create it, and save it for future use. You can not retrieve it at a later stage.

   - **MTLS**: Select MLTS to enable 2-way authentication of both the user and the Kafka broker.
     - **Truststore**: Locate and upload the truststore file. This field is applicable only to connect to an SSL enabled Kafka cluster.
     - **Truststore Password**: Enter the truststore password.
     - **Keystore**: Locate and upload the keystore file. This field is applicable only to connect to an SSL enabled Kafka cluster.
     - **Keystore Password**: Enter the keystore password.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.13 Creating a Connection to Microsoft Azure Data Lake-Gen2

To create a connection to Microsoft Azure Data Lake-Gen2:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **HDFS** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.

- **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
- **Description**
- **Tags**
- **Connection Type**: The selected connection is displayed.

4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **core-site.xml**: Upload the `core-site.xml` file with `fs.defaultFS`, `fs.default.name`, `hadoop.security.authentication`, and `fs.AbstractFileSystem.hdfs.impl` properties.

     > **Note:**
     >
     > Download the files `core-site.xml` and `hdfs-site.xml`from the Azure website.

   - **hdfs-site.xml** : Set client-specific properties. This is an optional field.
   - **Use Kerberos**: To use a kerberized cluster, select this option to enable Kerberos principal and Kerberos keytab. Provide the following details:
     - **Kerberos KDC**
     - **Kerberos Realm**
     - **Kerberos Principal**: Set HDFS service principal.
     - **Kerberos Key Tab**: Upload the keytab file that has HDFS service principal
6. Click **Test Connection**, to ensure that you have successfully created a connection, and to download the third-party libraries required to connect to Azure Data lake-Gen2. The libraries are also used by targets to write to HDFS paths.

   > **Note:**
   >
   > Retain the `core-site.xml` and `hdfs-site.xml` file names exactly as they are.

7. Click **Save.**

4.1.1.14 Creating a Connection to MongoDB

To create a connection to MongoDB:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **MongoDB** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**

- **Tags**
- **Connection Type**: The selected connection is displayed.

4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Connection Mode**: Select the connection mode from the drop-down list:
     - **Server Address List:**
       - **Server Address List**: Connection to a list of Replicat set members or mongos. This field accepts a comma separated list of `hostnames:port`. For example, `localhost1:27017, localhost2:27018`.
         - **Authentication Mechanism**: Select the authentication mechanism for the connection, from the drop-down list. This is an optional field. Enter the following details for the mechanism you select:
           - **Username**: Enter the database account user name.
           - **Password**: Enter the password for your database account.
           - **Credentials Source**: Enter the source of the authentication credentials, typically the database that the credentials have been created in.
           - **Write Concern**: Enter the value in JSON format. Accepted keys are `w` and `wtimeout`. For example, `{"w": "value" , "wtimeout": "number"}`
     - **Client URI**:
       - **Client URI**: Set the client URI in the format: `mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][?options]]`
       - **Authentication Mechanism**:
         - **None**: Select this option to disable connection authentication.
         - **SSL Server Certificate Validation**:
           - **Trust Store File**: Upload the truststore file.
           - **Trust Store Password**: Set the truststore password.
         - **SSL Server/Client Certificate Validation**:
           - **Trust Store File**: Upload the truststore file.
           - **Trust Store Password**: Set the truststore password.
           - **Key Store File**: Upload the client certificate, for a two-way SSL communication.
           - **Key Store Password**: Set the keystore password.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.15 Creating a Connection to MySQL Database

To create a connection to a MySQL Database:

1. On the **Catalog** page, click **Create New Item**.

2. Hover the mouse over **Connection** and select **Generic Database** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **database**: Select a MySQL database to connect to.
   - **Jdbc url**: Enter the JDBC connection url to create a database connection. The format for a MySQL JDBC url:

     `jdbc:mysql://<user>:<password>@<host>:<port>/<database>`

     Replace `<user>`, `<password>`, `<host>`, `<port>`, `<database>` with the MySQL database username, password, hostname or IP of MySQL database server; the MySQL database server port, and database name respectively.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.16 Creating a Connection to OCI Object Store

To create an OCI Object Store connection:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **OCI** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **OCI User OCID**: Enter the OCI user ID.
   - **OCI Fingerprint** : Enter the fingerprint of the API public key file that you uploaded to OCI. For example, `oci_api_key_public.pem`.
   - **OCI Key File**: Select the API private key file for signing API calls. For example, oci_api_key.pem.
   - **Key Passphrase**: Enter the Passphrase for the API private key file. The API key can be passphrase-protected.

- **OCI Tenancy OCID**: Enter the OCID of the tenant in which the Object Store bucket is defined.
- **OCI Profile**: Set the OCI profile. Default value is `DEFAULT`.
- **OCI Namespace**: Enter the OCI namespace that spans all compartments within a region.
- **Region**: Enter the region in which tenancy is created. For a list of OCI regions, refer to the *Region Identifier* column in the Regions and Availability Domains documentation.
- **OCI Compartment OCID**: Enter the OCID of the compartment in which the ONS topic or Object Store is defined.

6. Click **Test Connection**, to validate the credentials to connect to OCI, and to ensure that the dependent client libraries are downloaded from maven central repository.
7. Click **Save.**

4.1.1.17 Creating a Connection to ONS

The OCI Notification Service (ONS) connection option is currently available only on the OCI Marketplace GGSA instance.

To create an ONS connection:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **OCI** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **OCI User OCID**: Enter the OCI user ID.
   - **OCI Fingerprint** : Enter the fingerprint of the API public key file that you uploaded to OCI. For example, `oci_api_key_public.pem`.
   - **OCI Key File**: Select the API private key file for signing API calls. For example, oci_api_key.pem.
   - **Key Passphrase**: Enter the Passphrase for the API private key file. The API key can be passphrase-protected.
   - **OCI Tenancy OCID**: Enter the OCID of the tenant in which the ONS topic is defined.
   - **OCI Profile**: Set the OCI profile. Default value is `DEFAULT`.
   - **OCI Namespace**: Enter the OCI namespace that spans all compartments within a region.
   - **OCI Region**: Enter the region in which tenancy is created.
     For a list of OCI regions, refer to the *Region Identifier* column in the Regions and Availability Domains documentation.

- **OCI Compartment OCID**: Enter the OCID of the compartment in which the ONS topic or Object Store is defined.

6. Click **Test Connection**, to validate the credentials to connect to OCI, and to ensure that the dependent client libraries are downloaded from maven central repository.
7. Click **Save.**

> **Note:**
>
> To integrate with OCI Notification service, you have to define an OCI Notification service topic. Once you have defined a topic, note down the following parameters that are required to send Messages from GGSA to OCI Notification:
>
> - **My Message**: The message that target pushed to OCI Notification service topic.
> - **Topic**: The topic created on OCI Notification service. The OSA target can publish message to this topic.
> - **Email, Function, HTTPS, Slack**: The subscriptions to the topic. All users who have subscribed to the topic receive the message.

4.1.1.18 Creating a Connection to Oracle AQ

To create a connection to Oracle Advanced Queue (AQ):

1. On the **Catalog** page, click **Create New Item**
2. Hover the mouse over **Connection** and select **JNDI** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **JNDI Provider**: Select Oracle AQ as the JNDI service provider.
   - **Server Url(s)**: Enter the server url(s) for the JNDI connection, in the format `host1:port1, host2:port2`.
   - **Username**: Enter the user name to authenticate the JNDI connection.
   - **Password**: Enter the password for the JNDI connection.
   - **Jndi Other Properties**: This value should always be:
     - `sid=<sid>`. For example, `sid=XE`

       or
     - `service_name=<service_name>` For example, `service_name=slc.us.oracle.com`.
6. Click **Test Connection**, to ensure that you have successfully created a connection.

7. Click **Save.**

4.1.1.19 Creating a Connection to Oracle Database

To create a connection to Oracle Database:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Oracle Database** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Connect using**: Select the database identifiers from the drop-down list.
   - **Service name/SID**: Enter the name of the Service, SID, or Wallet that you want to connect to.
   - **Host name**: Enter the host name on which the database is running.
   - **Port**: Enter the port on which the database is running. It is usually `1521`.
   - **Username**: Enter the database account user name.
   - **Password**: Enter the password for your database account.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.1.20 Creating a Connection to OSS

To create a connection to OSS:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Kafka** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:

- **Use Bootstrap**: Check this box to use a bootstrap based connection. This is mandatory for OSS connections. Connection to OSS can only be established using the Bootstrap server option.
- **Kafka bootstrap**: Enter the bootstrap URL.
- **SSL**: Do not check this box when connecting to OCI Streaming Service.
- **SASL**: Check this box if Kafka broker requires authentication. This is mandatory for OSS connections.
- **User Name**: Enter the SASL username for the Kafka broker, in the following format:

  ```
  tenancyName/username/stream pool id
  ```

  > **Note:**
  >
  > Enter the tenancyName and userName, not tenancy OCID and user OCID. Similarly, enter the stream pool ID and not the stream pool name. Ensure that the auto create topic is enabled for the stream pool ID.

  You can retrieve this information from the OCI console. This field is enabled only if you have checked the SASL option.
- **Password** — Enter the SASL password, which is an authentication token that you can generate on the **User Details** page, of the OCI console.

  > **Note:**
  >
  > Copy the authentication token when you create it, and save it for future use. You can not retrieve it at a later stage.

6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

4.1.2 Manage Connections

Updating a Connection

To update a connection:

1. Go to the **Catalog** page and click the connection that you want to update.
2. On the **Edit Connection** screen, make the necessary changes.
3. Click **Save**.

Deleting a Connection

To delete a connection:

1. Go to the **Catalog** page and hover the mouse over the connection that you want to delete.
2. Click the delete icon that appears to your right side on the screen.
3. On the **Delete Confirmation** screen, click **Delete**.

4.2 Streams

4.2.1 Create Streams

A Stream is a source of continuous and dynamic data. The data can be from a wide variety of data sources such as IoT sensors, transaction or activity log files, point-of-sale devices, ATM machines, transactional databases, or information from geospatial services or social networks.

4.2.1.1 Creating a File Stream

To create a File stream:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Stream** and select **File** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the stream. This is a mandatory field.
   - **Display Name**: Enter a display name for the stream. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Stream Type**: The selected stream is displayed.
4. Click **Next**.
5. On the **Source Details** screen, enter the following details:
   - **File**: Upload the CSV or JSON sample file to be used.

     > **Note:**
     >
     > Use File stream only for POCs and quick prototyping

   - **Read whole content**: Select this option to read all the records in the file, at once. If you uncheck this option, the engine reads one record at a time.
   - **Number of events per batch**: Enter the number of records that you want to process per batch. The default value is one, but you can specify the number of records to process in each read. You can use this option only when **Read Whole Content** is unchecked.
   - **Loop**: Select this option to process the file in a loop.
   - **Data Format**: Select CSV or JSON as the data format.
6. Click **Next**.
7. On the **Data Format** screen, set the attributes for the selected the data format.
   - For JSON data format:

  - **Allow Missing Column Names**: Select this option to allow an input stream that has a column undefined in the shape.
  - **Array in Multi-lines**: Select this option to allow multi-line data formatting.
- For CSV data format:
  - **CSV Predefined Format**: Select one of the predefined data format from the drop-down list. For more information, see Predefined CSV Data Formats.
  - **First record as header**: Select this option to use the first record as the header row.

8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape** : Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape** : Select this option to infer the fields from a stream or file. You can also update the datatype of the fields.

     Note:

     - To retrieve the entire JSON payload, add a new field with path `$`.
     - To retrieve the content of the array, add a new field with path `$[arrayField]`.

     In both the cases, the value returned is Text.
   - **From File**: Select this option to infer the shape from a JSON schema file, or a JSON or CSV data file. You can also save the auto-detected shape and use it later.
10. Click **Save**.

4.2.1.2 Creating a GoldenGate Stream

Prerequisites:

- A GoldenGate connection.
- GG Change Data

To create a GoldenGate stream:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Stream** and select **GoldenGate** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the stream. This is a mandatory field.
   - **Display Name**: Enter a display name for the stream. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Stream Type**: The selected stream is displayed.

4. Click **Next**.
5. On the **Source Type** page, enter the following details:
   - **Connection**: Select a GG Change Data.
   - **Table name**: Enter a valid table name that includes the `period (.)` delimiter between the catalog, schema, and table names. For example, `test.dbo.table1`
   - **Generate Full Records**: Select this option to stream full data record (value of all fields), irrespective of the database transactional changes to a single column, a subset, or all the columns of a row.
     - **Database Connection**: Select a GoldenGate sourced database connection.
     - **Enable Cache**: Select this option to enable caching for GoldenGate Full Records, to enhance its performance.
6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape** : Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape** : Select this option to manually infer the fields from a stream or file. You can also update the datatype of the fields.

     > **Note:**
     >
     > - To retrieve the entire JSON payload, add a new field with path `$`.
     > - To retrieve the content of the array, add a new field with path `$[arrayField]`.

     In both the cases, the value returned is Text.
   - **From Stream**: Select this option to detect the shape based on the table shape selected in the previous screen.
   - **From File**: Select this option to infer the shape from a JSON file. You can also save the auto-detected shape and use it later.
8. Click **Save**.

> **Note:**
>
> The difference between a Kafka stream and a GoldenGate stream is that the pipeline constructs, like the Query Group Table, understands the GoldenGate syntax and associates it with the relevant GoldenGate fields.

4.2.1.3 Creating a JMS Stream

Prerequisite: A JMS connection.

To create a JMS stream:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Stream** and select **JMS** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the stream. This is a mandatory field.
   - **Display Name**: Enter a display name for the stream. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Stream Type**: The selected stream is displayed.
4. Click **Next**.
5. On the **Source Type** page, enter the following details:
   - **Connection**: Select an existing JNDI connection for the stream
   - **Connection Factory**: Enter a value for the connection factory. A ConnectionFactory encapsulates connection configuration information, and enables JMS applications to create a Connection. The default value is `weblogic.jms.ConnectionFactory`.

     > **Note:**
     >
     > GGSA can read messages from Oracle Advanced Queue. This option is available as a general JMS connector - `oracle.jms.AQjmsInitialContextFactory`.

   - **Jndi name**: Enter the name of the Java interface that reads messages from topics, distributed topics, queues and distributed queues
   - **Client ID**: Enter the unique client ID to be used for a durable subscriber. If you do not provide this value, subscriber ID is used as a clientID to create a durable subscriber.
   - **Message Selector** : Set the message selector to filter messages. Message selectors assign the work of filtering messages to the JMS provider rather than to the application.

     If your messaging application needs to filter the messages it receives, you can use a JMS API message selector. A message selector is a `String` that contains an expression. The syntax of the expression is based on a subset of the SQL92 conditional expression syntax. The message selector in the following example selects any message that has a `NewsType` property that is set to the value `Sports` or `Opinion`:

     ```
     NewsType = 'Sports' OR NewsType = 'Opinion'
     ```

     The `createConsumer` and `createDurableSubscriber` methods allow you to specify a message selector as an argument when you create a message consumer.
   - **Subscription ID**: Enter the unique subscription ID for durable selector. This value is essential for durable subscriber.

> **Note:**
>
> When you specify both clientID and subscriberID, you can have only one running pipeline consuming that stream. If you need multiple subscribers/ pipelines, remove clientID or subscriberName from the stream or create different streams (with different clientID and subscriberName) for multiple pipelines.

- **Data Format**: Select the data format from the drop-down list. The supported formats are: CSV, JSON, AVRO, MapMessage.

  A MapMessage object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined.

6. Click **Next**.
7. On the **Data Format** screen, enter the shape details for the stream, based on the data format you have selected.
   - For JSON:
     - **Allow Missing Column Names**: Select this option to allow an input stream that has a column undefined in the shape.
   - For CSV:
     - **CSV Predefined Format**: Select one of the predefined data format from the drop-down list. For more information, see Predefined CSV Data Formats.
     - **First record as header**: Select this option to use the first record as the header row.
   - For AVRO:
     - **Schema Namespace**: Enter the schema name combined with the namespace, to uniquely identify the schema within the store.
     - **Schema (optional)**: Upload a schema file to infer shape from.
   - If you selected MapMessage as the data format, there are no specific attributes to be set on this screen. The **Data Format** screen is skipped, and you are redirected to the **Shape** screen.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape** : Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also update the datatype of the fields.

> **Note:**
>
> – To retrieve the entire JSON payload, add a new field with path `$`.
>
> – To retrieve the content of the array, add a new field with path `$[arrayField]`.

In both the cases, the value returned is of type Text.

- **From File**: Select this option to infer the shape from a JSON schema file, or a JSON or CSV data file. You can also save the auto-detected shape and use it later.

10. Click **Save**.

JMS Server Clean-Up

GGSA creates a durable subscription with the JMS provider, when you create a JMS stream and select the durable subscription option. When you unpublish or kill a pipeline that is using this stream, the durable subscription still remains on the JMS Server. It is advisable to delete the durable subscription from the JMS Server and clean up the resources, if you do not intend to publish the pipeline anymore.

4.2.1.4 Creating a Kafka Stream

Prerequisite: A Kafka connection.

To create a Kafka stream:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Stream** and select **Kafka** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the stream. This is a mandatory field.
   - **Display Name**: Enter a display name for the stream. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Stream Type**: The selected stream is displayed.
4. Click **Next**.
5. On the **Source Details** screen, enter the following details:
   - **Connection**: Select a Kafka connection for the stream.
   - **Topic name**: Enter a name for the kafka topic that will store the stream.
   - **Data Format**: Select CSV, JSON, or AVRO as the data format for the stream.

     for each format type:
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details for the stream, based on the data format you have selected.
   - For JSON:

  - **Allow Missing Column Names**: Select this option to allow an input stream that has a column undefined in the shape.
- For CSV:
  - **CSV Predefined Format**: Select one of the predefined data formats from the drop-down list. For more information, see Predefined CSV Data Formats.
  - **First record as header**: Select this option to use the first record as the header row.
- For AVRO:
  - **Schema Namespace**: Enter the schema name combined with the namespace, to uniquely identify the schema within the store.
  - **Schema (optional)**: Upload a schema file to infer shape from.

8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape** : Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape** : Select this option to manually infer the fields from a stream or file. You can also update the datatype of the fields.

     > **Note:**
     >
     > - To retrieve the entire JSON payload, add a new field with path `$`.
     > - To retrieve the content of the array, add a new field with path `$[arrayField]`.

     In both the cases, the value returned is of type Text.
   - **From Stream**: Select this option to detect the shape based on the earliest or the latest offset of the kafka topic. The default option is **earliest**. Use **latest** to infer the shape based on latest records in the Kafka topic.
     This option is currently available only for JSON data format.
   - **From File**: Select this option to infer the shape from Kafka, a JSON schema file, or a JSON or CSV data file. You can also save the auto-detected shape and use it later. This option is enabled if you have selected CSV as the data format.
   - **From Schema**: Select this option to infer the shape based on the schema you selected in **Step 6**. This option is enabled if you have selected AVRO as the data format.
10. Click **Save**.

4.2.2 Manage Streams

Updating a Stream

To update a stream:

1. Go to the **Catalog** page and click the stream that you want to update.

2. On the **Edit Stream** screen, click the **Edit** link corresponding to the following sections, and make the necessary changes.
   - Source Details
   - Source Type Parameters
   - Data Type Parameters
   - Source Shape
3. Click **Save**.

Deleting a Stream

To delete a stream:

1. On the **Catalog** page, hover the mouse over the stream that you want to delete.
2. Click the **Delete**icon that appears to your right side on the screen.
3. On the **Delete Confirmation** screen, click **Delete**.

4.2.2.1 Application Timestamp

When defining a Stream, you can mark one of the fields in the payload, as an Application Timestamp. To do this, click the clock icon next to the field. This action advances the time by the application, rather than the system; the window ranges and slides are all controlled by the selected field. The application timestamp is available only to query stages connecting directly to the stream source.

4.2.2.2 Supported Timestamp Formats in an Input Stream

The following timestamp formats, in an input stream, are supported:

- 11/21/2005 11:14:23.111 "MM/dd/yyyy HH:mm:ss.SSS"
- 11/21/2005 11:14:23.11 "MM/dd/yyyy HH:mm:ss.SS"
- 11/21/2005 11:14:23.1 "MM/dd/yyyy HH:mm:ss.S"
- 11/21/2005 11:14:23 "MM/dd/yyyy HH:mm:ss"
- 11/21/2005 11:14 "MM/dd/yyyy HH:mm"
- 11/21/2005 11:14 "MM/dd/yyyy HH"
- 11/21/2005 "MM/dd/yyyy"
- 11-21-2005 11:14:23.111 "MM-dd-yyyy HH:mm:ss.SSS"
- 11-21-2005 11:14:23.11 "MM-dd-yyyy HH:mm:ss.SS"
- 11-21-2005 11:14:23.1 "MM-dd-yyyy HH:mm:ss.S"
- 11-21-2005 11:14:23 "MM-dd-yyyy HH:mm:ss"
- 11-21-2005 11:14 "MM-dd-yyyy HH:mm"
- 11-21-2005 11 "MM-dd-yyyy HH"
- 11-21-2005 "MM-dd-yyyy"
- 15-DEC-01 11.14.14.111 AM"dd-MMM-yy hh.mm.ss.SSS"
- 15-DEC-01 11.14.14.11 "dd-MMM-yy hh.mm.ss.SS"
- 15-DEC-01 11.14.14.1 "dd-MMM-yy hh.mm.ss.S"

- 15-DEC-01 11.14.14 "dd-MMM-yy hh.mm.ss"
- 15-DEC-01 11.14 "dd-MMM-yy hh.mm"
- 15-DEC-01 11 "dd-MMM-yy hh"
- 15-DEC-01 "dd-MMM-yy"
- 15/DEC/01 "dd/MMM/yy"
- 2013-10-5 15:16:0.756 "yyyy-MM-dd HH:mm:ss.SSS"
- 2013-10-5 15.16.0.756 "yyyy-MM-dd HH.mm.ss.SSS"
- 2013-10-5 15:16:0 "yyyy-MM-dd HH:mm:ss"
- 2013-10-5 15.16.0 "yyyy-MM-dd HH.mm.ss"
- 2013-10-5 15:16 "yyyy-MM-dd HH:mm"
- 2013-10-5 15.16 "yyyy-MM-dd HH.mm"
- 2013-10-5 15 "yyyy-MM-dd HH"
- 2012-11-10 "yyyy-MM-dd"
- 11:14:14 "HH:mm:ss"
- "yyyy-MM-dd'T'HH:mm:ss'.'SSS"
- "yyyy-MM-dd'T'HH:mm:ss"
- 1/1/2011 "m/d/yyyy"
- 1-1-2011 "m-d-yyyy"
- 3/23/2019 "m/dd/yyyy"
- 3-23-2019 "m-dd-yyyy"
- 12/4/1982 "mm/d/yyyy"
- 12-4-2019 "mm-d-yyyy"

> **Note:**
>
> The input timestamp is truncated to millisecond precision.

4.2.2.3 Predefined CSV Data Formats

Comma Separated Values (CSV) file is one of the data formats you can select for your input stream. There are variations in the CSV data format due to the different data sources. The following table lists the available predefined CSV data formats:

CSV Predefined Format	Description
`DEFAULT`	Standard comma separated format, as for `RFC4180` but allowing empty lines
`EXCEL`	Excel file format with comma as the value delimiter

CSV Predefined Format	Description
INFORMIX_UNLOAD_CSV	Default Informix CSV UNLOAD format used by the UNLOAD TO file_name operation (escaping is disabled.) This is a comma-delimited format with a LF character as the line separator. Values are not quoted and special characters are escaped with '\'. The default NULL string is "\ \N".
MYSQL	Default MySQL format used by the SELECT INTO OUTFILE and LOAD DATA INFILE operations. This is a tab-delimited format with a LF character as the line separator. Values are not quoted and special characters are escaped with '\'. The default NULL string is "\\N".
POSTGRESQL_CSV	Default PostgreSQL CSV format used by the COPY operation. This is a comma-delimited format with a LF character as the line separator. The default NULL string is "".
POSTGRESQL_TEXT	Default PostgreSQL text format used by the COPY operation. This is a tab-delimited format with a LF character as the line separator. The default NULL string is "\\N".
RFC4180	Comma separated format as defined by RFC4180
TDF	Tab-delimited format

4.3 References

Create References

Manage References

4.3.1 Create References

A reference is a source of static data that provides contextual information about the event data. References are used to enrich data that arrives from a stream.

GGSA supports the following reference types:

- Coherence Reference: This is a reference to an external cache that has data from an external system, and is defined in a coherence cluster.
- Database Reference: This is a reference to a specified table in the database. You can apply caching to this reference to enhance the static data accessibility. Once you load data into the cache, the reference fetches data from the cache only. Any update to the reference table will not be reflected until you set the expiration policy to Never.
- Ignite Cache Reference: This is a reference to an Ignite cache cluster. Ignite reference enriches the stream data with the Ignite cluster reference data, at query stage. Ignite reference has data in key-value pair; key is String type and value is a Json object. GGSA supports only single-value equality join, in query stage to reference.

4.3.1.1 Creating a Coherence Reference

To create a Coherence reference:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Reference** and select **Coherence** from the submenu.
3. On the **Type Properties** screen, enter the following details:

- **Name**: Enter a unique name for the reference. This is a mandatory field.
- **Display Name**: Enter a display name for the reference. If left blank, the **Name** field value is copied.
- **Description**
- **Tags**
- **Reference Type**: The selected reference is displayed.

4. Click **Next**.
5. On the **Source Details** page, provide the following details:
   - **Connection**: Select a coherence connection for the reference

     .
   - **Cache name**: Enter the name of the coherence cache to enable caching. Caching is supported only for single equality join condition.
   - **Data Format**: Select POJO or Map as the data format for the reference.
6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape :
   - If you selected Map as the data format, you have the following options to define the shape:
     Select Existing Shape: Select an existing shape that you want to use for the reference.
   - **Manual Shape**: Select this option if you want to define your own shape.
8. Click **Save**.

For information on data mapping in the two coherence reference types, see:

- Data Mapping in Coherence Reference Map Type
- Data Mapping in Coherence Reference POJO Type

4.3.1.2 Creating a Database Reference

To create a Database reference type:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Reference** and select **Database** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the reference. This is a mandatory field.
   - **Display Name**: Enter a display name for the reference. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Reference Type**: The selected reference is displayed.
4. Provide details for the following fields on the **Source Details** page and click **Next**:
   - **Connection**: Select the connection for the database reference.

- **Enable Caching**: Select this option to enable caching. Ignite cache is the default cache for cache-enabled references. So before deploying a pipeline using cache-enabled references, you have to start the cache cluster from the **System Settings** tab.

  > **Note:**
  >
  > The **Enable Caching** option is not supported in the GGSA marketplace instance.

- **Caching Scheme**: Select the cache type from the drop-down list.

  In a Partitioned Cache the data is partitioned among all the machines of the cluster.

  In a Replicated Cache the data is fully replicated to every member of the cluster. Use Replicated Cache when the number of cache entries are relatively low and do not need to be updated often.

- **Expiry Delay**: The duration delay from last update that the entries will be kept by the cache before being marked as expired. Any attempt to read an expired entry will result in a reloading of the entry from the configured cache store. This field is enabled only when caching is enabled.

5. Provide details for the following fields on the **Shape** page and click **Save**:
   - **Shape Name**: Select a shape that you want to use for the reference

When the datatype of the table data is not supported, the table columns do not have auto generated datatype. Only the following datatypes are supported:

- `numeric`
- `interval day to second`
- `text`
- `interval year to month`
- `timestamp` (without timezone)
- `date time` (without timezone)

> **Note:**
>
> The `date` column cannot be mapped to `timestamp`. This is a limitation in the current release.

4.3.1.3 Creating an Ignite Reference

To create an Ignite Cache reference:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Reference** and select **Ignite Cache** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the reference. This is a mandatory field.
   - **Display Name**: Enter a display name for the reference. If left blank, the **Name** field value is copied.

- **Description**
- **Tags**
- **Reference Type**: The selected reference is displayed.

4. Click **Next**.
5. On the **Source Details** page, provide the following details:
   - **Connection**: Select an ignite cache connection for the reference.
   - **Cache name**: Enter the name of the ignite cache to enable caching. Caching is supported only for single equality join condition.
   - **Data Format**: Select the data format from the drop-down list.
6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape**: Select this option to detect the shape automatically from the input data stream.
     - **From Stream**: Select this option to infer shape from a stream.
     - **From File**: Select this option to infer the shape from a JSON schema file, or a JSON or CSV data file. You can also save the auto-detected shape and use it later.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually edit the shape. You can also update the datatype of the fields.
8. Click **Save**.

4.3.2 Manage References

Updating a Reference

To update a Reference:

1. Go to the **Catalog** page and click the reference that you want to update.
2. On the **Reference** screen, click the **Edit** link corresponding to the following sections, and make the necessary changes.
   - Source Details
   - Source Type Parameters
   - Source Shape
3. Click **Save**.

Deleting a Reference

To delete a reference:

1. Go to the **Catalog** page and hover the mouse over the reference that you want to delete.
2. Click the delete icon that appears to your right side on the screen.
3. On the **Delete Confirmation** screen, click **Delete**.

4.3.2.1 Coherence Reference

4.3.2.1.1 Configuring Extend Proxy on the Coherence Server

To enable communication between GoldenGate Stream Analytics and the Coherence server, you must configure extend proxy on the Coherence server or the cluster. GGSA can connect to the Coherence server only by extending its proxy service. To configure extend proxy on the Coherence server, define the following configuration in the server's `cache-config.xml`, as in the format below:

```
<caching-schemes>
      .............

      <proxy-scheme>
         <service-name>ExtendTcpProxyService</service-name>
         <acceptor-config>
            <tcp-acceptor>
               <local-address>
                  <address>{ADDRESS}</address>
                  <port>{PORT}</port>
               </local-address>
            </tcp-acceptor>
         </acceptor-config>
         <autostart>true</autostart>
      </proxy-scheme>

</caching-schemes>
```

4.3.2.1.2 Limitations of Coherence as Reference

The following are the limitations of a Coherence cache as a reference:

- You cannot test the connection
- You need to specify the cache name manually
- You can choose only equal operator to establish a correlation with coherence reference
- You must use manual shape
- You can choose only one look-up condition under sources. This condition will be used for key look-up.

4.3.2.1.3 Loading Number Type Data on Coherence Cache

When you load big decimal number type data on the coherence cache, ensure that you include precision and scale. Only when you specify these values, the join functionality works.
Below is a sample in which a number type data is defined, with the scale and precision values specified:

```
NamedCache cache  = CacheFactory.getCache("externalcachetimestamp");

        java.math.BigDecimal big10 = new java.math.BigDecimal("10",new
MathContext(58)).setScale(56, RoundingMode.HALF_UP);

        Map<String,Object> order1 = new HashMap<String, Object>();
```

```
order1.put("strValue", "Test");
order1.put("intervalValue", "+000000002 03:04:11.330000000");
        order1.put("orderTag", big10);

        cache.put(big10,order1);
```

4.3.2.1.4 Data Mapping in Coherence Reference Map Type

The `Map` type coherence reference maps with the cache data in the key-value format. Key is object type and value is `Map<String,Object>`. `Map<String,Object>` is a map of attribute names and values. The attributes list should match with external event type. GGSA currently supports only external schema for key and value. Only single value equality join is supported.

4.3.2.1.5 Data Mapping in Coherence Reference POJO Type

The `POJO` type coherence reference maps with the cache date in the format `Map<String, Object>`.

> **Note:**
>
> When you upload the POJO in a jar, you must ensure that the fully-qualified class name of POJO matches exactly in cache and the custom POJO jar.
>
> For example, if you have loaded `com.company.CustomPOJO` objects in cache, custom JAR should have `CustomPOJO` class inside the `com.company` package.
>
> If there any mismatch in the class name, then querying the cache for certain type of objects, does not return a result and you will not see any data in the live output table.

4.3.2.1.6 Datatypes Supported in Correlation Conditions

POJO Coherence Reference supports only some data types as join keys, in correlation conditions.

Data types that are supported:

- `Java.lang.String (Interval and Interval YM)`
- `java.math.BigDecimal (Number)`
- `Java.lang.String (Text)`
- `int (Integer)`
- `Java.lang.Integer (Integer)`
- `Java.lang.Long (BigInteger)`
- `long (BigInteger)`
- `java.sql.Timestamp (Timestamp)`
- `double (Double)`
- `Java.lang.Double (Double)`
- `float (Float)`

- Java.lang.Float (Float)
- boolean (Boolean)
- Java.lang.Boolean (Boolean)
- Java.math.BigInteger

Data types that are not supported:

- char[]
- char
- BigInteger
- oracle.spatial.geometry.JGeometry(SDO Geometry)
- java.sql.Date (Timestamp)

4.3.2.1.7 Sample POJO Cache Loading in Coherence

The following is a sample POJO cache loading in coherence:

```
NamedCache<Integer, Object> cache = CacheFactory.getCache

    ("externalcachepojo");

    private void preseedCoherencePOJOReferenceData(NamedCache<Integer,

    Object> cache) {

    OrderPOJO order1 = new OrderPOJO(1,"HP Deskjet v2");
    OrderPOJO order2 = new OrderPOJO(2,"Oracle Database 12");
    OrderPOJO order3 = new OrderPOJO(3,"Apple iPhone6s");
    OrderPOJO order4 = new OrderPOJO(4,"Logitech Mouse");
    cache.put(1, order1);
    cache.put(2, order2);
    cache.put(3, order3);
    cache.put(4, order4);
    }
```

For this example coherence reference should be created with cache name as `externalcachepojo` and can join with stream with `orderid`.

4.3.2.1.8 Sample POJO Class

```
public class OrderPOJO implements Serializable{
private String orderId ;
Private String orderDesc;

public String setOrderId(String str1) {
         this.orderId=str1;
}
public String setOrderDesc(String str2) {
        this.orderDesc=str2;
}

public String getOrderId() {
```

```
return orderId;
}
public String getOrderDesc() {
return orderDesc;
}
public boolean equals(Object object) {
if (this == object) return true;
if (object == null || getClass() != object.getClass()) return false;
if (!super.equals(object)) return false;
OrderPOJO that = (OrderPOJO) object;
return java.util.Objects.equals(orderId, this.orderId) &&
java.util.Objects.equals(orderDesc, this.orderDesc);
}
public int hashCode() {
return java.util.Objects.hash(super.hashCode(), orderId, orderDesc);
}
}
```

> **Note:**
>
> Ensure that the POJO class does not have a GGSA coherence target as a constructor, because it can instantiate the POJO class using default constructor, and then access the setXXX and getXXX, and isXXX methods.

4.4 Targets

Create Targets

Manage Targets

4.4.1 Create Targets

A target is an external system to which the stream processing results are ouput. It is an interface with a downstream system.

The supported target types are:

- AWS S3
- Azure DataLake Gen-2
- Coherence Target
- Database Target
- Elasticsearch Target
- HBase Target
- HDFS Target
- Hive Target
- Ignite Cache Target
- JMS Target
- Kafka Target

- MongoDB Target
- NFS Target
- Notification Target
- Object Storage Target
- OSS Target
- REST Target

4.4.1.1 Creating an AWS S3 Target

To create an AWS S3 target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **AWS S3** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select an AWS connection from the drop-down list.
   - **File Name**: Enter the name of the file used to save data to the AWS S3 bucket.
   - **AWS S3 Path**: Enter a name for folder to be created in the AWS S3 bucket. A new folder is created if there is no existing folder.
   - **Bucket**: Enter the name of the bucket to be created. A new bucket is created if there is no existing bucket in the region.
   - **Roll Interval**: Enter the roll-over interval to write a new file. The interval can be in 1000ms, 10s, 1m, or 1.5h format.
   - **File Roll Max Size**: Enter the roll-over file size to create a new file. The size can be in 1000, 10k, 1m, or 1g format.
   - **NFS Path**: Enter the local file or NFS path where the files are written first and then uploaded to the AWS S3 bucket.
   - **Storage Format**: Select a storage format from the drop-down list.
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the storage format you have selected.
   - For FILE:
     - **File Format**: Select a file format from the drop-down list.
       * **JSON Delimiter**: Enter the JSON delimiter if you have selected the JSON file format. This is an optional field.

        - **Avro Codec**: Select a compression codec from the drop-down list. This option is enabled if you have selected the file format as AVRO or AVRO Object Container Format.
    - For PARQUET:
        - **PARQUET Compression**: Select a compression codec from the drop-down list.
    - For ORC:
        - **ORC Compression**: Select a compression codec from the drop-down list.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
    - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
    - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
        - **Shape Name**: Enter a name for the shape.
        - **Clear Fields**: Click to delete all the fields in the shape.
        - **Field Name**: Add the necessary fields.
        - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

4.4.1.2 Creating an Azure DataLake Gen-2 Target

To create Azure DataLake Gen-2 target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Azure DataLake Gen-2** from the submenu.
3. On the **Type Properties** screen, enter the following details:
    - **Name**: Enter a unique name for the target. This is a mandatory field.
    - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
    - **Description**
    - **Tags**
    - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
    - **Connection**: Select an HDFS connection from the drop-down list.
    - **HDFS File**: Enter a file name. The file name is appended with current timestamp and the extension, based on the type of storage format.
    - **HDFS Path**: Enter the HDFS location. Provide full access to this location to enable users other than the folder owner, to write to this path.
    - **File Roll Interval**: Enter the roll-over interval to write a new file. The interval can be in 1000ms, 10s, 1m, or 1.5h format.
    - **File Roll Max Size**: Enter the roll-over file size to create a new file. The size can be in 1000, 10k, 1m, or 1g format.

- **NFS Path**: Enter the local file or NFS path where the files are written first and then uploaded to HDFS.
- **Storage Format**: Select a storage format from the drop-down list.

6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the storage format you have selected.
   - For FILE:
     - **File Format**: Select a file format from the drop-down list.
       * **JSON Delimiter**: Enter the JSON delimiter if you have selected the JSON file format.
       * **Avro Codec**: Select a compression codec from the drop-down list. This option is enabled if you have selected the file format as AVRO or AVRO Object Container Format.
   - For PARQUET:
     - **PARQUET Compression**: Select a compression codec from the drop-down list.
   - For ORC:
     - **ORC Compression**: Select a compression codec from the drop-down list.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - **Shape Name**: Enter a name for the shape.
     - **Clear Fields**: Click to delete all the fields in the shape.
     - **Field Name**: Add the necessary fields.
     - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

4.4.1.3 Creating a Coherence Target

To create an Coherence target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Coherence** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.

5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select a coherence connection.
   - **Cache Name**: Enter a name for the coherence cache.
   - **Data Format**: Select a data format from the drop-down list.
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details for the stream, based on the data format you have selected.
   - For JSON:
     - **Create nested json object**: Select this option to create a nested JSON object for the target.
8. Click **Next**.
9. On the **Shape** screen, enter the following details:
   - For JSON:
     - **Shape Name**: Enter a name for the shape.
     - **Clear Fields**: Click to clear all the fields from the existing shape.
     - **Key**: Select key fields, based on which data is partitioned. For example, records containing the same values for the selected key fields will all be stored in the same Kafka partition.
       You can select multiple fields as key. Key selection is not mandatory.
     - **Field Name**: Add the necessary fields.
     - **Field Path**: Enter the field path.

       > **Note:**
       > * To retrieve the entire JSON payload, add a new field with path `$`.
       > * To retrieve the content of the array, add a new field with path `$[arrayField]`.

       In both the cases, the value returned is of type Text.
     - **Field Type**: Select the field data type from the drop-down list.
   - For POJO:
     - **Jar Name**: Select a jar name from the custom POJO jars, from the drop-down list.
     - **Class Name**: Select the class name from the POJO classes in the chosen jars, from the drop-down list.
10. Click **Save**.

4.4.1.4 Creating a Database Target

Prerequisite: A Database connection.

To create a Database target:

1. On the **Catalog** page, click **Create New Item**.

2. Hover the mouse over **Target** and select **Database** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select a database connection from the drop-down list.
6. Click **Next**.
7. On the **Shape** screen, enter the following details:
   - **Table Name**: Select a database table from the drop-down list.
8. Click **Save**.

4.4.1.5 Creating an Elasticsearch Target

To create an Elastic Search target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Elastic Search** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select an Elastic Search connection from the drop-down list.
   - **Index Name**: An Elasticsearch index is a collection of documents with similar characteristics. You can create an index name only in lowercase.
     Example format: `index.name`, your index in elastic search will be`index_name`.
6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape:
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually define a shape. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - **Shape Name**: Enter a name for the shape.

- **Clear Fields**: Click to delete all the fields in the shape.
- **Key**: Select one or more fields as key, which will be used as the ID field.
  Example:
  `{"_index":"json_data","_type":"_doc","_id":"2","_score":1.0,"_source":{"address":"Mumbai","serial":"2","clientName":"Joe"}}]}}`

> **Note:**
>
> * Any update to the value will result in new entry rather than updating previous value.
> * If a key has a null value, ElasticSearch will autogenerate the key. In the example above, ID is 2, because `serial` is selected as the key field. If record has null in serial field: `{"address":"Mumbai","serial":null,"clientName":"Joe"}`, then ID will be autogenerated by Elasticsearch.
> * Index is json_data which is provided in previous step, ID will be value of each record.

- **Field Name**: Add the necessary fields.
- **Field Type**: Select the field data type from the drop-down list.

8. Click **Save**.

4.4.1.6 Creating an HBase Target

To create HBase target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **HBase** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select an HBase connection from the drop-down list.
   - **Column Family**: Provide a column family to group the columns in the HBase table. GGSA and HBase Handler support only a single column family.
   - **Table Name**: Select an already created table in HBase, or provide a table name for GGSA to create a table, with default HBase table properties.
6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape:

- **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
- **Manual Shape**: Select this option to manually define a shape. You can also add to, or remove fields from, an existing shape. Enter the following details:
  - **Shape Name**: Enter a name for the shape.
  - **Clear Fields**: Click to delete all the fields in the shape.
  - **Key**: Selecting atleast one field in the HBase table as a primary key. A primary key is mandatory.
  - **Field Name**: Add the necessary fields.
  - **Field Type**: Select the field data type from the drop-down list.

8. Click **Save**.

4.4.1.7 Creating HDFS Target

To create HDFS target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **HDFS** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select an HDFS connection from the drop-down list.
   - **HDFS File**: Enter a file name. The file name is appended with current timestamp and the extension, based on the type of storage format.
   - **HDFS Path**: Enter the HDFS location. Provide full access to this location to enable users other than the folder owner, to write to this path.
   - **File Roll Interval**: Enter the roll-over interval to write a new file. The interval can be in 1000ms, 10s, 1m, or 1.5h format.
   - **File Roll Max Size**: Enter the roll-over file size to create a new file. The size can be in 1000, 10k, 1m, or 1g format.
   - **NFS Path**: Enter the local file or NFS path where the files are written first and then uploaded to HDFS.
   - **Storage Format**: Select a storage format from the drop-down list.
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the storage format you have selected.
   - For FILE:
     - **File Format**: Select a file format from the drop-down list.

        * **JSON Delimiter**: Enter the JSON delimiter if you have selected the JSON file format.
        * **Avro Codec**: Select a compression codec from the drop-down list. This option is enabled if you have selected the file format as AVRO or AVRO Object Container Format.
    - For PARQUET:
      - **PARQUET Compression**: Select a compression codec from the drop-down list.
    - For ORC:
      - **ORC Compression**: Select a compression codec from the drop-down list.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
    - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
    - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
      - **Shape Name**: Enter a name for the shape.
      - **Clear Fields**: Click to delete all the fields in the shape.
      - **Field Name**: Add the necessary fields.
      - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

4.4.1.8 Creating a Hive Target

To create Hive target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Hive** from the submenu.
3. On the **Type Properties** screen, enter the following details:
    - **Name**: Enter a unique name for the target. This is a mandatory field.
    - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
    - **Description**
    - **Tags**
    - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
    - **Connection**: Select a Hive connection from the drop-down list.
    - **Table Name**: Enter a table name for the external hive table to be created. The table is created in the database mentioned in the JDBC url.
    - **HDFS Path**: Enter the file path to write the Avro_OCF files. Data from these files are loaded into the external tables.
    - **Schema File Path**: Enter the HDFS path to write the Avro_OCF schema file. This schema file is used to derive the schema of the external hive table. Ensure that the Avro_ocf data file path and the schema file path are different.

- **File Roll Interval**: Enter the roll-over interval to write a new file. The interval can be in 10ms, 10s, 10m, 1hr formats.
- **File Roll Max Size**: Enter the roll-over file size to create a new file. The size can be in 1000, 10k, 10m, 1g formats.

6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape:
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - **Shape Name**: Enter a name for the shape.
     - **Key**: Select key fields, based on which the data is partitioned.
     - **Clear Fields**: Click to delete all the fields in the shape.
     - **Field Name**: Add the necessary fields.
     - **Field Type**: Select the field data type from the drop-down list.
8. Click **Save**.

4.4.1.9 Creating an Ignite Cache Target

To create an Ignite Cache target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select**Ignite Cache** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select an Ignite connection. Select an embedded connection if you have the Internal cache cluster is started.
   - **Cache Name**: Enter a unique cache name for the ignite cluster. This will be verified when the cache name is validated.
   - **Expiry Delay**: Select the cache expiry period from the drop-down list.
   - **Caching Scheme**: Select a scheme from the drop-down list. This field is not applicable for an embedded cluster.
     - **Replicated** : If you select this scheme, all the data is replicated to every node in the cluster.
     - **Partitioned**: If you select this scheme, the entire data set is divided equally into partitions.

- **Backup**: Enter the number of backup nodes. This field is not applicable for an embedded cluster.
- **Update Cache Entry**: Select this option to update a particular key value with new data. This option is enabled by default.
- **Data Format**: Select a data format from the drop-down list.

6. Click **Next**.
7. On the **Data Format** screen, enter the shape details for the stream, based on the data format you have selected.
   - For JSON:
     - **Create nested json object**: Select this option to create a nested JSON object for the target.
8. Click **Next**.
9. On the **Shape** screen, enter the following details:
   - For JSON:
     - **Shape Name**: Enter a name for the shape.
     - **Clear Fields**: Click to clear all the fields from the existing shape.
     - **Key**: Select a key from the input data to store record.
       You can select multiple fields as key. Key selection is mandatory.
     - **Field Name**: Add the necessary fields.
     - **Field Path**: Enter the field path.
     - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

> **Note:**
>
> You cannot edit an Ignite target once created. This restriction avoids cache data corruption because only one target from the GGSA platform is allowed to write to only one cache in the ignite server.

4.4.1.10 Creating a JMS Target

Prerequisite: A JMS connection.

To create a JMS target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **JMS** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**

- **Target Type**: The selected target is displayed.

4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select a JNDI connection.
   - **JNDI name**: Enter a name for the JNDI topic.
   - **Data Format**: Select a data format from the drop-down list.
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the data format you have selected.
   - For JSON:
     - **Create nested json object**: Select this option to create a nested JSON object for the target.
   - For CSV:
     - **CSV Predefined Format**: Select one of the predefined data formats from the drop-down list. For more information, see Predefined CSV Data Formats.
     - **First record as header**: Select this option to use the first record as the header row.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape**: Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - For JSON:
       * **Shape Name**: Enter a name for the shape.
       * **Clear Fields**: Click to remove all the fields from the shape.
       * **Key**: Select key fields, based on which data is partitioned. For example, records containing the same values for the selected key fields will all be stored in the same Kafka partition.
         You can select multiple fields as key. Key selection is not mandatory.
       * **Field Name**: Add the necessary fields.
       * **Field Path**: Enter the field path.

         > **Note:**
         >
         > * To retrieve the entire JSON payload, add a new field with path `$`.
         > * To retrieve the content of the array, add a new field with path `$[arrayField]`.

         In both the cases, the value returned is of type Text.

        - **Field Type**: Select the field data type from the drop-down list.
    - For CSV, AVRO, and MapMessage:
        - **Shape Name**: Enter a name for the shape.
        - **Clear Fields**: Click to delete all the fields from the shape.
        - **Field Name**: Add the necessary fields.
        - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

4.4.1.11 Creating a Kafka Target

Prerequisite: A Kafka connection.

To create a Kafka target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Kafka** from the submenu.
3. On the **Type Properties** screen, enter the following details:
    - **Name**: Enter a unique name for the target. This is a mandatory field.
    - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
    - **Description**
    - **Tags**
    - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
    - **Connection**: Select a Kafka connection.
    - **Topic name**: Enter a name for the Kafka topic.
    - **Data Format**: Select a data format from the drop-down list.
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the data format you have selected.
    - For JSON:
        - **Create nested json object**: Select this option to create a nested JSON object for the target.
    - For CSV:
        - **CSV Predefined Format**: Select one of the predefined data formats from the drop-down list. For more information, see Predefined CSV Data Formats.
        - **First record as header**: Select this option to use the first record as the header row.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
    - **Infer Shape**: Select this option to detect the shape automatically from the input data stream.

- **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
- **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
  - For JSON:
    - **Shape Name**: Enter a name for the shape.
    - **Clear Fields**: Click to remove all the fields from the shape.
    - **Key**: Select key fields, based on which data is partitioned. For example, records containing the same values for the selected key fields will all be stored in the same Kafka partition.
      You can select multiple fields as key. Key selection is not mandatory.
    - **Field Name**: Add the necessary fields.
    - **Field Path**: Enter the field path.

      > **Note:**
      >
      > - To retrieve the entire JSON payload, add a new field with path `$`.
      > - To retrieve the content of the array, add a new field with path `$[arrayField]`.

      In both the cases, the value returned is of type Text.
    - **Field Type**: Select the field data type from the drop-down list.
  - For CSV and AVRO:
    - **Shape Name**: Enter a name for the shape.
    - **Clear Fields**: Click to delete all the fields in the shape.
    - **Field Name**: Add the necessary fields.
    - **Field Type**: Select the field data type from the drop-down list.

10. Click **Save**.

4.4.1.12 Creating a MongoDB Target

To create an MongoDB target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **MongoDB** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.

4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select a MongoDB connection from the drop-down list.
   - **Database**: Enter the name of the database to be used for the target.
   - **Collection**: Enter the name of the collection to insert documents.
6. Click **Next**.
7. On the **Shape** screen, select one of the methods to define the shape:
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually define a shape. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - **Shape Name**: Enter a name for the shape.
     - **Clear Fields**: Click to delete all the fields in the shape.
     - **Key**: Select none, or one or more fields as key. This key will be the ID field.
     - **Field Name**: Add the necessary fields.
     - **Field Type**: Select the field data type from the drop-down list.
8. Click **Save**.

4.4.1.13 Creating a Network File System (NFS) Target

To create an NFS target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **NFS** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **File Name**: Enter the name of the file to be stored on the local file system or NFS. The file name is prefixed with timestamp and extension, when finally stored.
   - **NFS Path**: Enter the shared file path which is accessible from the Spark cluster nodes.
   - **File Roll Interval**: Enter the roll-over interval to write a new file. The interval can be in 1000ms, 10s, 1m, or 1.5h format.
   - **File Roll Max Size**: Enter the roll-over file size to create a new file. The size can be in 1000, 10k, 1m, or 1g format.
   - **Storage Format**: Select a storage format from the drop-down list.
6. Click **Next**.

7. On the **Storage Format** screen, enter the shape details, based on the storage format you have selected.
   - For FILE:
     - **File Format**: Select a file format from the drop-down list.
     - **Avro Codec**: Select a compression codec from the drop-down list. This option is enabled if you have selected the file format as AVRO or AVRO Object Container Format.
   - For PARQUET:
     - **PARQUET Compression**: Select a compression codec from the drop-down list.
   - For ORC:
     - **ORC Compression**: Select a compression codec from the drop-down list.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - **Shape Name**: Enter a name for the shape.
     - **Clear Fields**: Click to delete all the fields in the shape.
     - **Field Name**: Add the necessary fields.
     - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

> **Note:**
>
> In case of JSON, AVRO, AVRO OCF schema files would be written under `NFS Path/SCHEMA` folder.

4.4.1.14 Creating a Notification Target

Prerequisite: An OCI connection.

To create a Notification target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Notification** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.

5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select an OCI connection from the drop-down list.
   - **Topic**: Enter the OCID of the topic.
   - **Data Format**: Select a data format. Currently, OSA supports only JSON data format.
6. Click **Next**.
7. On the **Shape** screen, you do not have the option to define a shape. An already populated shape is displayed. Enter the following details:
   - **Header**: Enter the message header.
   - **Body**: Enter the message body.
8. Click **Save**.

> **Note:**
>
> ONS connection type is no longer supported in GGSA. Recreate older Notification type targets in the pipeline, using an OCI connection.

4.4.1.15 Creating an OCI Object Store Target

To create an OCI Object Store target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Object Storage** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**:
   - **Object Storage File Name**: Enter the name of the file written to the Object Storage bucket.
   - **Object Storage File Path**: Enter the name of the folder to be created in the Object Storage bucket. A new folder is created if there is no existing folder.
   - **Object Storage Bucket**: Enter the OCI Object Storage bucket.
   - **File Roll Interval**: Enter the roll-over interval to write a new file. The interval can be in 1000ms, 10s, 1m, or 1.5h format.
   - **File Roll Max Size**: Enter the roll-over file size to create a new file. The size can be in 1000, 10k, 1m, or 1g format.

- **NFS Path**: Enter the local file or NFS path where the files are written first and then uploaded to the Object Storage.
- **Storage Format**: Select a storage format from the drop-down list.

6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the storage format you have selected.
   - For FILE:
     - **File Format**: Select a file format from the drop-down list.
       * **JSON Delimiter**: Enter the JSON delimiter if you have selected the JSON file format.
       * **Avro Codec**: Select a compression codec from the drop-down list. This option is enabled if you have selected the file format as AVRO or AVRO Object Container Format.
   - For PARQUET:
     - **PARQUET Compression**: Select a compression codec from the drop-down list.
   - For ORC:
     - **ORC Compression**: Select a compression codec from the drop-down list.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - **Shape Name**: Enter a name for the shape.
     - **Clear Fields**: Click to delete all the fields in the shape.
     - **Field Name**: Add the necessary fields.
     - **Field Type**: Select the field data type from the drop-down list.
10. Click **Save**.

4.4.1.16 Creating an OSS Target

Prerequisite: A Kafka connection.

To create a Kafka target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **Kafka** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.

4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **Connection**: Select a Kafka connection.
   - **Topic name**: Enter a name for the kafka topic.
   - **Data Format**: Select a data format from the drop-down list.
6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the data format you have selected.
   - For JSON:
     - **Create nested json object**: Select this option to create a nested JSON object for the target.
   - For CSV:
     - **CSV Predefined Format**: Select one of the predefined data formats from the drop-down list. For more information, see Predefined CSV Data Formats.
     - **First record as header**: Select this option to use the first record as the header row.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape**: Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - For JSON:
       * **Shape Name**: Enter a name for the shape.
       * **Clear Fields**: Click to clear all the fields from the existing shape.
       * **Key**: Select key fields, based on which data is partitioned. For example, records containing the same values for the selected key fields will all be stored in the same Kafka partition.
         You can select multiple fields as key. Key selection is not mandatory.
       * **Field Name**: Add the necessary fields.
       * **Field Path**: Enter the field path.

         > **Note:**
         >
         > * To retrieve the entire JSON payload, add a new field with path `$`.
         > * To retrieve the content of the array, add a new field with path `$[arrayField]`.

         In both the cases, the value returned is of type Text.
       * **Field Type**: Select the field data type from the drop-down list.

- For CSV and AVRO:
  - **Shape Name**: Enter a name for the shape.
  - **Clear Fields**: Click to clear all the fields from the existing shape.
  - **Field Name**: Add the necessary fields.
  - **Field Type**: Select the field data type from the drop-down list.

10. Click **Save**.

4.4.1.17 Creating a REST Target

To create a REST target:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Target** and select **REST** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the target. This is a mandatory field.
   - **Display Name**: Enter a display name for the target. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Target Type**: The selected target is displayed.
4. Click **Next**.
5. On the **Target Details** screen, enter the following details:
   - **URL**: Enter the REST service URL.
   - **Use SSL**: Select this option to enable SSL and basic authentication.
   - **Trust Store File** > **Upload File**: Click to upload the Truststore file.
   - **Trust Store Password**: Enter the truststore password.

     If you do not have the Truststore file and password, click **Trust password** to connect the REST end point.

     > **Note:**
     >
     > The Trust Store File and Trust Store Password options allow the use of untrusted certificates for REST connections, resulting in an insecure connection.

   - **Trust Anyway**: Select this option to supersede the TrustStoreFile selection.
   - **Custom HTTP headers**: Set the custom headers for HTTP in the format `key=value[,key2=value2, ....]`, without quotes. If the end point requires authentication, you can pass it as a custom header field.

     An example custom header would be `Authorization=Basic XXXXXXXX`, where `XXXXXXXX` is a base64-encoded string of `username:password`.

- **Batch processing**: Select this option to process batch events as a single request. Enable this option for high throughput pipelines. For example,

```
Eg: [{"address":
          { "street" : xxxxxxx }
},{"address":
          { "street" : xxxxxxa }]
```

- **HTTP Method**: Select this option for the REST target to send requests to REST end-point, using Http POST and PUT methods. Default is set to POST.
- **Data Format**: Select a data format from the drop-down list.

6. Click **Next**.
7. On the **Data Format** screen, enter the shape details, based on the data format you have selected.
   - For JSON:
     - **Create nested json object**: Select this option to create a nested JSON object for the target. For example, if the target shape is defined as

```
field:attribute_street, field_path:address/street.
}
```

       then output json is

```
{"address":
          { "street" : xxxxxxx }
```

   - For CSV:
     - **CSV Predefined Format**: Select one of the predefined data formats from the drop-down list. For more information, see Predefined CSV Data Formats.
     - **First record as header**: Select this option to use the first record as the header row.
8. Click **Next**.
9. On the **Shape** screen, select one of the methods to define the shape:
   - **Infer Shape**: Select this option to detect the shape automatically from the input data stream.
   - **Select Existing Shape**: Select one of the existing shapes from the drop-down list.
   - **Manual Shape**: Select this option to manually infer the fields from a stream or file. You can also add to, or remove fields from, an existing shape. Enter the following details:
     - For JSON:
       - **Shape Name**: Enter a name for the shape.
       - **Clear Fields**: Click to remove all the fields from the shape.
       - **Key**: Select key fields, based on which data is partitioned. For example, records containing the same values for the selected key fields will all be stored in the same Kafka partition.
         You can select multiple fields as key. Key selection is not mandatory.
       - **Field Name**: Add the necessary fields.

* **Field Path**: Enter the field path.

> **Note:**
>
> * To retrieve the entire JSON payload, add a new field with path `$`.
> * To retrieve the content of the array, add a new field with path `$[arrayField]`.

In both the cases, the value returned is of type Text.

* **Field Type**: Select the field data type from the drop-down list.

– For CSV:

* **Shape Name**: Enter a name for the shape.
* **Clear Fields**: Click to delete all the fields in the shape.
* **Field Name**: Add the necessary fields.
* **Field Type**: Select the field data type from the drop-down list.

10. Click **Save**.

4.4.2 Manage Targets

Coherence Target

4.4.2.1 Coherence Target

4.4.2.1.1 Datatypes supported in the POJO class

The following data types are supported in the POJO class:

- java.lang.String
- java.lang.Integer
- java.lang.Long
- java.lang.Float
- java.lang.Double
- java.lang.Boolean
- java.math.BigDecimal
- java.math.BigInteger

4.4.2.1.2 Sample POJO Class

```
public class OrderPOJO implements Serializable{
private String orderId ;
Private String orderDesc;

public String setOrderId(String str1) {
         this.orderId=str1;
```

```
}
public String setOrderDesc(String str2) {
        this.orderDesc=str2;
}

public String getOrderId() {
return orderId;
}
public String getOrderDesc() {
return orderDesc;
}
public boolean equals(Object object) {
if (this == object) return true;
if (object == null || getClass() != object.getClass()) return false;
if (!super.equals(object)) return false;
OrderPOJO that = (OrderPOJO) object;
return java.util.Objects.equals(orderId, this.orderId) &&
java.util.Objects.equals(orderDesc, this.orderDesc);
}
public int hashCode() {
return java.util.Objects.hash(super.hashCode(), orderId, orderDesc);
}
}
```

> **Note:**
>
> Ensure that the POJO class does not have a GGSA coherence target as a constructor, because it can instantiate the POJO class using default constructor, and then access the setXXX and getXXX, and isXXX methods.

4.4.2.1.3 Sample Code Snippet to declare a Method which returns Boolean

If a method in the POJO class returns a Boolean value, prefix the method name with `is` instead of `get`, while defining the POJO class.

```
Class Abc {

private boolean var1;

public setVar1(boolean aa){
this.var1 = aa;
}
public boolean isVar1(){
return var1;
}
}
```

4.5 Pipelines

Create a Pipeline

Manage Pipelines

4.5.1 Create a Pipeline

To create a pipeline:

1. On the **Catalog** page, click **Create New Item**, and select **Pipeline** from the drop-down list.
2. On the **Type Properties** screen, enter the following details:
   - **Name**
   - **Description**
   - **Tags**
   - **Stream**: Select a stream from the drop-down list.
3. Click **Save**.

4.5.2 Manage Pipelines

4.5.2.1 Using the Pipeline Editor

The canvas on which you edit a pipeline and add different stages to the pipeline is called *Pipeline Editor*.

In the *Pipeline Editor*, you can:

- Adjust the pipeline pane, editor pane, and the live output table pane using the resizing arrows
- See the relationship and dependencies between various stages of the pipeline
- Add any type of stage to any of the existing stages in the pipeline.
  To add a stage:
  1. Right-click the stage after which you want to add the new stage.
  2. Click **Add a Stage**, and select the stage type to add.
  3. Provide the details for the new stage.
  4. Click **Save**.
- Expand or collapse a pipeline.
  - To expand a pipeline, click the
- Switch the layout of the pipeline to vertical or horizontal.
- Zoom to fit a pipeline

4.5.2.2 Publishing a Pipeline

You must publish a pipeline to make the pipeline available for all the users of Oracle Stream Analytics and send data to targets.

A published pipeline will continue to run on your Spark cluster after you exit the Pipeline Editor, unlike the draft pipelines which are undeployed to release resources.

To publish a pipeline:

1. Open a draft pipeline in the **Pipeline Editor**.
2. Click **Publish**.

The Pipeline Settings dialog box opens.

3. Update any required settings. See Configuring Pipeline Preferences.

> **Note:**
>
> Make sure to allot more memory to executors in the scenarios where you have large windows.

4. Click **Publish** to publish the pipeline.

   A confirmation message appears when the pipeline is published.

You can also publish a pipeline from the Catalog using the **Publish** option in the **Actions** menu.

4.5.2.3 Unpublishing a Pipeline

Unpublishing a pipeline from the Catalog page

1. Go to the **Catalog** page and hover the mouse over the pipeline that you want to unpublish.
2. Click the **Unpublish** icon that appears to your right side on the screen.
3. On the **Warning** screen, click **OK**.

Unpublishing a pipeline from the *Pipeline Editor*

1. Click the **Unpublish** button at the top right corner of the pipeline editor.
2. On the **Warning** screen, click **OK**.

4.5.2.4 Exporting and Importing a Pipeline and Its Dependent Artifacts

The export and import features let you migrate your pipeline and its contents between Oracle Stream Analytics systems (such as development and production). You also have the option to migrate only select artifacts. You can import a pipeline developed with the latest version of Oracle Stream Analytics. On re-import, the existing metadata is overwritten with the newly imported metadata if the pipeline is not published. You can delete the imported artifacts by right-clicking them and selecting **Delete**.

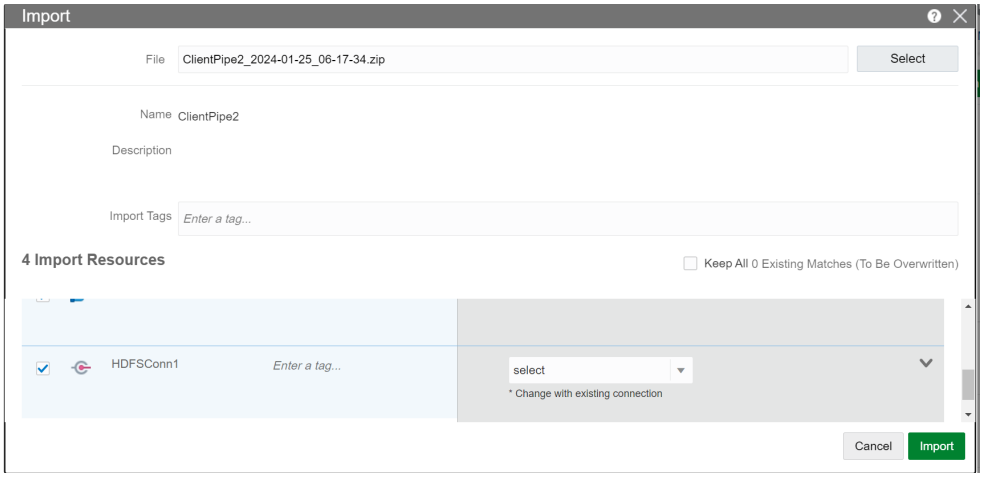
To export a pipeline:

1. On the **Catalog** page, hover the mouse over, or select the pipeline that you want to export to another GGSA instance.
2. Click the **Export** option that appears to your right side on the screen.
3. The selected pipeline and its dependent artifacts are exported as a JSON zip file, to your computer's default `Downloads` folder.

To import a pipeline:

1. Go to the GGSA instance to which you want to import the exported metadata.
2. On the **Catalog** page, click **Import**.
3. In the **Import** dialog box, click **Select**, to locate and select the exported zip file on your computer.

4. On the **Import Resources** tab, you can select an existing connection from the catalog, or use the imported connection.



5. Click **Import**.

The imported pipeline and its dependent artifacts are available on the **Catalog** page.

Note:

- Each pipeline should have a unique name. If you are importing an updated version of a pipeline, you can retain the same name. If you are importing a new pipeline and if a pipeline with the same name already exists in the catalog, change the name of the pipeline that you are importing.
- If you have already exported a pipeline with the same name, update the pipeline name as below:
  1. Create a directory `exportUpdate`.
  2. Copy the exported zip, say `exportNameUpdateExample.zip`, to the folder `exportUpdate`.
  3. Unzip the file `exportNameUpdateExample.zip`.
  4. Open the json file in edit mode.
  5. Search for pipeline/ artifact name in the json file. For example, if `Nano pipeline` was the name given to the pipeline, update it to `Nano pipeline updated`.
  6. Update the json file in `exportNameUpdateExample.zip`.
  7. Import this zip.
  8. The pipeline is automatically assigned a name, using the display name.
  9. The draft pipeline and publish pipeline topic are created as below:
     a. `sx_Nanopipelineupdated_Nano_Stream_draft`
     b. `sx_Nanopipelineupdated_Nano_Stream_public`

4.5.2.5 Working with Live Output Table

The streaming data in the pipeline appears in a live output table. Select any stage in the pipeline to see its output.

Hide/Unhide Columns

In the live output table, right-click a column and click **Hide** to hide that column from the output. This option only hides the columns from the UI and does not remove them from the output. To unhide the hidden columns, click **Columns** and then click the eye icon to make the columns visible in the output.

Select/Unselect the Columns

Click the **Columns** link at the top of the output table to view all the columns available. Use the arrow icons to either select or unselect individual columns or all columns. Only the columns that you select appear in the output table and in the actual output when the pipeline is published.

Pause/Restart the Table

Click **Pause/Resume** to pause or resume the streaming data in the output table.

Perform Operations on Column Headers

Right-click on any column header to perform the following operations:

- **Hide**: Hides the column from the output table. Click the Columns link and unhide the hidden columns.
- **Remove from output**: Removes the column from the output table. Click the Columns link and select the columns to be included in the output table.
- **Rename**: Renames the column to the specified name.
- **Function**: Captures the column in Expression Builder using which you can perform various operations through the in-built functions.

Add a Timestamp

Include timestamp in the live output table by clicking the clock icon in the output table.

Reorder the Columns

Click and drag the column headers to right or left in the output table to reorder the columns.

4.5.2.6 Using the Topology Viewer

Topology is a graphical representation and illustration of the connected entities and the dependencies between the artifacts.

The topology viewer helps you in identifying the dependencies that a selected entity has on other entities. Understanding the dependencies helps you in being cautious while deleting or undeploying an entity. Oracle Stream Analytics supports two contexts for the topology — *Immediate Family* and *Extended Family*.

You can launch the Topology viewer in any of the following ways:

- Select the **Show topology** icon next to the Pipeline to launch the **Topology Viewer** for the selected entity.

- Click the **Show Topology** icon in the Pipeline Editor.

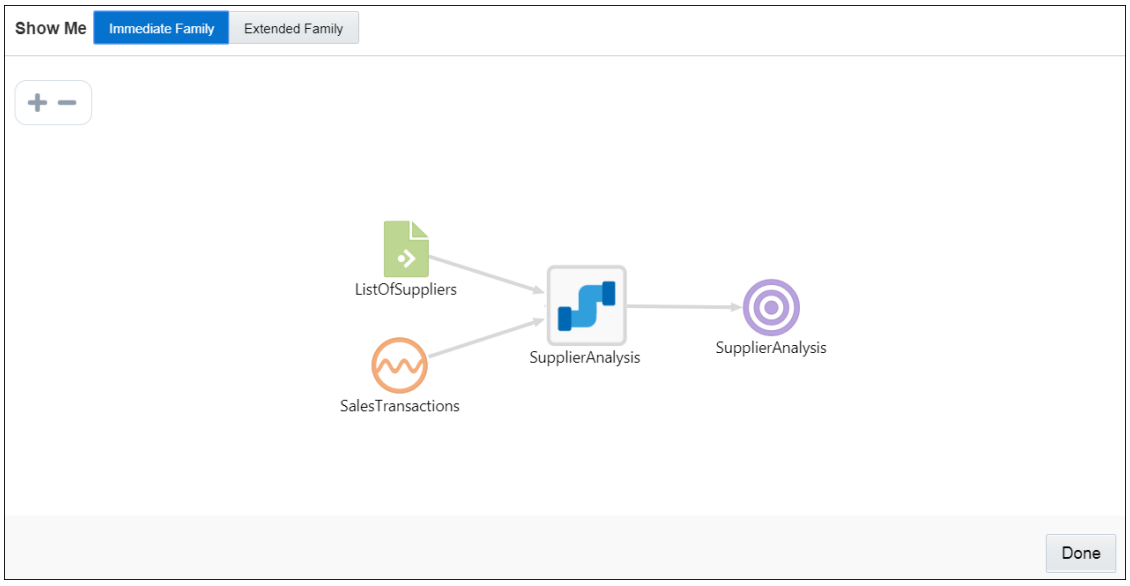
Click the **Show Topology** icon at the top-right corner of the editor to open the topology viewer. By default, the topology of the entity from which you launch the Topology Viewer is displayed. The context of this topology is **Immediate Family**, which indicates that only the immediate dependencies and connections between the entity and other entities are shown. You can switch the context of the topology to display the full topology of the entity from which you have launched the Topology Viewer. The topology in an **Extended Family** context displays all the dependencies and connections in the topology in a hierarchical manner.

> **Note:**
>
> The entity for which the topology is shown has a grey box surrounding it in the Topology Viewer.

Immediate Family

Immediate Family context displays the dependencies between the selected entity and its child or parent.

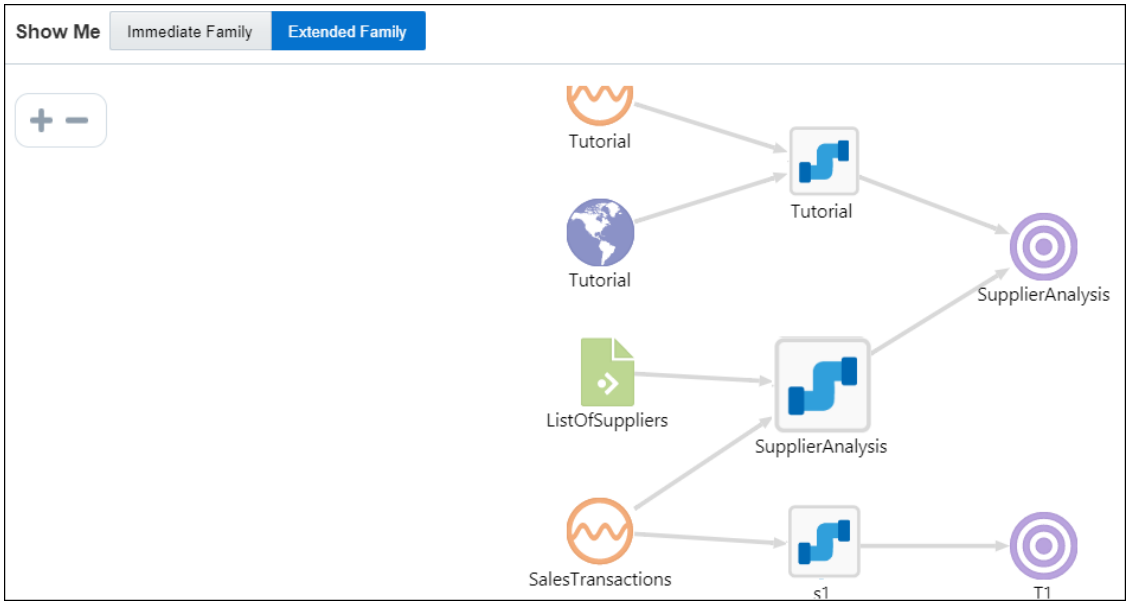
The following figure illustrates how a topology looks in the **Immediate Family**.



Extended Family

Extended Family context displays the dependencies between the entities in a full context, that is if an entity has a child entity and a parent entity, and the parent entity has other dependencies, all the dependencies are shown in the Full context.

The following figure illustrates how a topology looks in the **Extended Family**.



4.6 GoldenGate Change Stream

Getting a GoldenGate Change Stream into a Kafka Topic

Manage GG Change Data Stream

4.6.1 Getting a GoldenGate Change Stream into a Kafka Topic

To create a GG Change Data:

1. On the **Catalog** page, click **Create New Item** and select **GG Change Data** from the drop-down list.
2. On the **Type Properties** screen, enter the following details:
   - **Name**
   - **Description**
   - **Tags**
   - **GG Type**: Select **Change Data** from the drop-down list.
3. Click **Next**.
4. On the **GG Deployment Details** page, enter the following details:
   - **Connection**: Select a GG connection form the drop-down list.
   - **Deployments**: Select a deployment from the drop-down list.
   - **Deployment Username**: Enter the GoldenGate username of the deployment.
   - **Deployment Password**: Enter the GoldenGate password of the deployment.

> **Note:**
>
> The GoldenGate username and password of the deployment should be of the user with access to create a new distribution path from the Goldengate instance.

5. Click **Next**.
6. On the **GG Change Data Details** page, enter the following details:
   - **GG Extracts**: Select a GG stream from the drop-down list.
   - **Target Trail**: Enter a two character name for the Goldengate trail file.
   - **Kafka Connection**: Select a Kafka connection from the drop-down list.
   - **GG Change Data name**: Enter a name for the goldengate stream (maximum 8 characters). This name will be used for the replicat process that puts the change data from trail file to Kafka topics.
7. Click **Save**.

> **Note:**
>
> The following template parameter files for the replicat process are located at `osa-base/etc/`:
>
> - `kafka.props.template`
> - `replicat.prm.template`
> - `custom_kafka_producer.properties.template`
>
> You can modify these template files to customize the replicat process before proceeding to the next step.

4.6.2 Manage GG Change Data Stream

4.6.2.1 Starting a GoldenGate Change Stream

To start a GG Change Data stream:

1. Go to the **Catalog** page and hover the mouse over the **GG Change Data** stream that you want to start.
2. Click the **Start GG Change Data** icon that appears to your right side on the screen.

3. On the warning dialog box, click **OK**.

> **Note:**
>
> When you start a GG Change Data replicat process, it creates kafka topics, and starts pushing changed data to the new topics. For example, if there are 10 tables in the extract process that you chose while creating the GG Change Data, 10 new topics will be created.
>
> The names of the topics created are in the following format:
>
> `GGChangeDataName_fullyQualifiedTableName`
>
> You can use these topics to create a new stream (with Goldengate as stream type), and in pipelines, similar to using a Kafka stream.

4.6.2.2 Stopping a GG Change Data Stream

To stop a GG Change Data stream:

1. Go to the **Catalog** page and hover the mouse over the **GG Change Data** stream that you want to stop.
2. Click the **Stop GG Change Data** icon that appears to your right side on the screen..

3. On the warning dialog box, click **OK**.

4.6.2.3 Purging the GoldenGate Trail Files

The trail files are not needed once the replicat has finished processing them. You can purge the trail files to save disk space.

The default settings are as follows: `PURGEOLDEXTRACTS /location-of-trail-files MINKEEPHOURS 1, FREQUENCYMINUTES 10`.

The trail files, after being processed completely by the replicat process, and after one hour of inactivity, will be purged. The files will be checked for purging every 10 minutes.

You can modify the above rule, in an OCI GGSA VM, following the steps below:

1. Stop the manager process by running the command `sudo systemctl stop ggbd-mgr`.
2. Modify the rules in the file `/u01/app/ggbd/OGG_BigData_Linux_x64_19.1.0.0.0/dirprm/mgr.prm`.
3. Start the manager process by running the command `sudo systemctl start ggbd-mgr`.

For more information on rules about purging the trail files, see PURGEOLDEXTRACTS for Manager.

4.6.2.4 Streaming GoldenGate Full Records

The GoldenGate Extract process captures either full data records or transactional data changes, depending on the configuration parameters. To minimize the overhead or performance impact on the transactional database, GGSA users configure the Extract to capture only the transactional changes. This also helps to reduce payload size needs to transfer over the network, thus increasing the performance and security. But a few customers also need the unchanged columns (full data records), making them available to the processes that require up-to-date data feed, or to replicate this data to various big data targets for analysis.

To enable streaming of full data record (value of all fields), GGSA provides the **Generate Full Records** option while creating a GoldenGate stream. Enable this option to stream all the records, irrespective of the database transactional changes made to a single row, a subset or all the columns of a row.

> **Note:**
>
> Full Records option is not supported in the GGSA marketplace instance.

4.7 Embedded Ignite Cache

GGSA implements an Embedded Ignite Cache. Ignite can be used as a target or a reference for any pipeline, to persist events. In case of GoldenGate, the pipeline persists incoming events containing full record, and also updates the cache with modified events, making available the latest modified records.

> **Note:**
>
> Ignite Caching is not supported in the GGSA marketplace instance.

4.7.1 Starting a Cache Cluster

To start an Embedded Cache Cluster:

1. Open **System Settings** and click the **Cache Cluster** tab.
2. Update the **Persistence Store Path** to a preferred accessible location. This path will be the persistence storage for cached data. Default is set as `/tmp/ignite/persistence`.

   > **Note:**
   >
   > NFS mounted path is the preferred persistence store path, to enable cache rehydration, on restart of a cluster or a node.

3. Click **Start Cluster**.

> **Note:**
>
> When you start a cache cluster for the first time, an embedded Ignite Connection is created with connection type as **Ignite Cache** and Connection Name as **Embedded**. This connection is not editable connection, and connection details are not shown, but you can select this connection while creating an Ignite Target and an Ignite Reference.

4.7.2 Stopping a Cache Cluster

To stop a running Embedded Cache Cluster:

1. Open **System Settings** and click the **Cache Cluster** tab.
   The **Cluster Status** shows previous valid status, by default. The status is updated with the current status from server. You will see a green icon, indicating a running server status.

   Persistence Store Path is not editable, if the cluster is in running state.
2. Click **Stop Cluster**.

On a successful stop action, the cluster status changes to **Stopped**, and the persistence store path becomes editable.

4.7.3 Restarting a Cache Cluster

To restart a running Embedded Cache Cluster:

1. Open **System Settings** and click the **Cache Cluster** tab.
   The **Cluster Status** shows previous valid status, by default. The status is updated with the current status from server. You will see a red icon, indicating a killed server status.
2. Click **Restart Cluster**.

On a successful cluster restart, the status changes to **Running**, and the persistence store path is not editable.

4.7.4 Monitoring Cache in the Cache Cluster

You can monitor the caches in both the internal (embedded) and external clusters, using the **Monitor** option on the GGSA application's homepage.

To monitor a cache:

1. Click **Monitor** on the homepage.
2. On the **Cache Management** screen, select a cache connection from the drop-down list.

   The **Cache Management** page is populated with the following details for the selected cache connection:

   - All the Cache connections created using ignite, which includes both internal and external caches, are listed. Internal caches created from the **System Settings** tab are marked `Embedded`.
   - By default, all the caches created using the first connection in the list are displayed. If you select another connection from drop-down list, the corresponding caches will be retrieved and listed.
   - The **Created Target** column lists all the targets created using a cache.

- The **Referred By** column lists the references created using a cache.
- Click the **Reset Cache** icon to clear all the cache entries.
- Click the **Reload Cache** icon to reload the cache size.
- The caches are displayed in the ascending order of their names, by default. Use the **Sort By** option, to sort the caches by name or size.

4.8 Ignite Cluster on OCI GGSA

4.8.1 Starting an Ignite Cluster

To start an Ignite Cluster:

1. Open **System Settings**, select **Manage Clusters**, and expand the **Ignite Cluster** option to set the following parameters:
2. **Number of Cluster instances**: Specify the number of cluster instances that you require . The default value is 2. You can add up to 5 cluster instances.
3. **Memory Limit**: Enter the memory to allocate to the cluster instance.
4. **CPU Limit**: Enter the CPU limit to allocate to the cluster instance.
5. Click **Start Cluster**.
6.

The **Cluster Status** changes to **In Progress**. You will see a status message to refresh the page. Close the **System Settings** page and reopen it. The **Cluster Status** changes to **Running**.

4.8.2 Scaling an Ignite Cluster

To start an Ignite Cluster:

1. Open **System Settings**, select **Manage Clusters**, and from the **Ignite Cluster** drop-down, set the following parameters:
2. **Number of Cluster instances**: Update this value to increase or decrease the number of cluster instances. The default value is 2. You can add up to 5 cluster instances.
3. Click **Scale Cluster**.

The **Cluster Status** changes to **In Progress**. You will see a status message to refresh the page. Close the **System Settings** page and reopen it. The **Cluster Status** changes to **Running**. The cluster

4.8.3 Deleting Storage

To delete the storage allocated to the cluster instance:

1. Open **System Settings**, select **Manage Clusters**, and expand the **Ignite Cluster** option.

2. Select **Delete Storage**.

> **Note:**
>
> - The **Delete Storage** option is available only while the Ignite Cluster is running.
> - If you leave the box unselected, the cached values are retained for the next restart.

4.8.4 Stopping an Ignite Cluster

To stop an Ignite Cluster:

1. Open **System Settings**, select **Manage Clusters**, and expand the **Ignite Cluster** option.
2. Click **Stop Cluster**.

The **Cluster Status** changes to **Stopped**.

4.9 GGBD Cluster on OCI GGSA

OCI GoldenGate Stream Analytics embeds a GoldenGate Big Data environment to receive a change stream from GoldenGate extracts.

Starting a GGBD Cluster

Stopping a GGBD Cluster

4.9.1 Starting a GGBD Cluster

To start a GGBD Cluster:

1. Open **System Settings**, select **Manage Clusters**, and expand the **GGBD Cluster** option to set the following parameters:
2. **Memory Limit**: Enter the maximum memory to allocate to the cluster instance.
3. **CPU Limit**: Enter the maximum CPU limit to allocate to the cluster instance.
4. Click **Start Cluster**.

The **Cluster Status** changes to **In Progress**. You will see a status message to refresh the page. Close the **System Settings** page and reopen it. The **Cluster Status** changes to **Running**.

4.9.2 Stopping a GGBD Cluster

To stop an GGBD Cluster:

1. Open **System Settings**, select **Manage Clusters**, and expand the **GGBD Cluster** option.
2. Click **Stop Cluster**.

The **Cluster Status** changes to **Stopped**.

5
Transform

5.2 Correlating Streams and References

A *correlation* is used to enrich the incoming event in the data stream with static data in a database table or with data from other streams.

For example, if the event in the data stream only includes `SensorId` and `Sensor Temperature`, the event could be enriched with data from a table to obtain `SensorMake`, `SensorLocation`, `SensorThreshold`, and many more.

Correlating an event with other sources requires the join condition to be based on a common key. In the above example, the `SensorId` from the stream cand be used to correlate with `SensorKey` in the database table. The following query illustrates the above data enrichment scenario producing sensor details for all sensors whose temperature exceeds their pre-defined threshold.

```
Select T.SensorId, T.Temperature, D.SensorName, D.SensorLocation
From TemperatureStream[Now] T, SensorDetailsTable D
Where T.SensorId = D.SensorKey And T.Temperature > D.SensorThreshold
```

Queries like above and more complex queries can be automatically generated by configuring sources and filter sections of the query stage.

5.2.1 Joining Mutiple Streams

You can correlate a stream with another stream.

Stream-to-stream Correlation

- A Stream is an unbounded sequence of events. To correlate a Stream with another Stream, first convert both streams to a Relation or a bounded sequence of events, by applying window functions.
- After applying window functions on both streams, define a correlation condition that evaluates to true or false.

The output from stream-to-stream correlation is a subset of the Cartesian product of tuples from both windows, where the correlation condition is true.

5.2.2 Joining a Stream with a Reference or an External Source

You can join a stream with external data in a Database or a Coherence Cache.

Stream-to-Database Table Correlation

- Convert the Stream to a bounded sequence of events, by applying a window function.
- After applying the window function on the stream, define a correlation condition that evaluates to true or false.

The output from Stream-to-database correlation is the Cartesian product of tuples from window and the database table, where the correlation condition is true.

Stream-to-Cache Correlation

- Convert the Stream to a bounded sequence of events, by applying a window function.
- After applying the window function on the stream, define a correlation condition that evaluates to true of false.

The output from Stream-to-Cache is the Cartesian product of tuples from window and the cache, where the correlation condition is true. Currently, OSA supports only Coherence cache.

5.3 Applying Window Functions to a Stream

Apply window functions, to specify time and event based windows, to process your stream.

To apply a **Window** function:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the query stage to apply the window function.
3. Click the **Sources** tab.
4. Click the clock icon, and select the required window type from the **Window Type** drop-down list.

5.3.1 Applying a Time Window with Slide

- Range value: Integer
- Range unit: nanoseconds, microseconds, milliseconds, seconds, minutes, hours
- Slide value: Integer
- Slide unit: nanoseconds, microseconds, milliseconds, seconds, minutes, hours
- Applicable on: Query Stage

The CQL example is as follows:

```
[range 5 MINUTES slide 30 SECONDS]
```

In the above example, the data is retained for 5 minutes but the query is evaluated every 30 seconds.

> **Note:**
>
> There will be an output only if the current results of the query is different from the previous results. This avoids sending duplicates to downstream applications.

If you set the slide to same as range, it will create a tumbling window instead of a sliding window. For example,

```
[Range 5 MINUTES slide 5 MINUTES]
```

will only retain 5 minutes of data and the query will only execute every 5 minutes.

> **Note:**
>
> Use the tumbling window to batch output results before sending to downstream systems. For example, you may want to create a file on object store only after you have accumulated at least 10000 events. This would avoid the small-file problem in Big-Data systems. Similarly, you may want to avoid multiple writes to a database system and instead perform a single write, after sufficient events have been accumulated.

5.3.2 Applying a Time Window without Slide

- Range value: Integer
- Range unit: nanoseconds, microseconds, milliseconds, seconds, minutes, hours
- Applicable on: Query Stage, Query Group Stream Stage and Query Group Table Stage

The CQL example is as follows:

```
[range 1 minutes ]
```

In the example above, the default slide value which is same as spark streaming batch interval is used.

> **Note:**
>
> If you do not specify a slide value, it will take the default slide, which is same as the Spark batch interval.

5.3.3 Applying a Row Window with Slide

- Rows value: Integer
- Applicable on: Query Stage, Query Group Stream Stage and Query Group Table Stage

The CQL example is as follows:

```
[rows 10 slide 1]
```

Maximum window size is 10 events, but Slide of 1 implies the query is executed on the arrival of every new event.

5.3.4 Applying a Row Window without Slide

- Rows value: Integer
- Applicable on: Query Stage, Query Group Stream Stage and Query Group Table Stage

The CQL example is as follows:

```
[rows 10]
```

Last 10 events is used to evaluate the query. Default slide value is used.

5.3.5 Applying a window with current year, month, day, or hour

CurrentYear

- Applicable on: Query Stage, Detect Duplicates Pattern, Eliminate Duplicate Pattern

The CQL example is as follows:

```
[ CurrentYear ]
```

Data is retained until end of the current year. Default slide value is used.

CurrentMonth

- Applicable on: Query Stage, Detect Duplicates Pattern, Eliminate Duplicate Pattern

The CQL example is as follows:

```
[ CurrentMonth ]
```

Data is retained until the end of the current month. Default slide value is used.

CurrentDay

- Applicable on: Query Stage, Detect Duplicates Pattern, Eliminate Duplicate Pattern

The CQL example is as follows:

```
[ CurrentDay ]
```

CurrentHour

- Supported types of shape fields: timestamp, int, bigint
- Applicable on: Query Stage, Detect Duplicates Pattern, Eliminate Duplicate Pattern

The CQL example is as follows:

```
[ CurrentHour ]
```

Data is retained until the end of the current hour. Default slide value is used.

5.3.6 Applying your own Window using Field from Payload

- Interval value: interval
- Supported types of shape fields: timestamp
- Applicable on: Query and Query Group Stream Stage and Query Group Table Stage

The CQL example is as follows:

```
[range "DS_INTERVAL" on c1]
```

Here the range is based on a field value in the payload.

Use this window type to aggregate data using a timestamp column from payload. For example,

```
[range INTERVAL "2 0:0:0.0" DAY TO SECOND on EventCaptureTime]
```

will only retain events from the last 2 days, based on the timestamp value in **EventCaptureTime** field.

5.3.7 Applying a Row window with Partition without Range

- Shape fields of Partition by: MultiSelect
- Rows value: Integer
- Applicable on: Query Stage

The CQL example is as follows:

```
[partition by F1, F2 rows 10]
```

Last 10 events for each partition value. For example [partition by stockSymbol rows 10] will use last 10 quotes for ORCL, last 10 quotes for AMZN, etc.

Query is evaluated on the arrival of new events and not on time ticks.

Default slide value is used.

5.3.8 Applying a Row Window with Partition with Range without Slide

- Shape fields of Partition by: MultiSelect
- Rows value: Integer
- Range value: Integer
- Range unit: nanoseconds, microseconds, milliseconds, seconds, minutes, hours
- Applicable on: Query Stage

The CQL example is as follows:

```
[partition by F1, F2 rows 10 range 15 seconds]
```

Events may be evicted from the window even when it is not full with all 10 rows, but 15 seconds have elapsed since the event arrived.

5.3.9 Applying a Row Window with Partition with Slide and Range

- Shape fields of Partition by: MultiSelect
- Rows value: Integer
- Range value: Integer
- Range unit: nanoseconds, microseconds, milliseconds, seconds, minutes, hours
- Slide Value: Integer
- Slide unit: nanoseconds, microseconds, milliseconds, seconds, minutes, hours
- Applicable on: Query Stage

The CQL example is as follows:

```
[partition by F1, F2 rows 10 range 15 seconds slide 1 second]
```

5.1 Adding Stages to a Pipeline

5.1.1 Adding a Query Stage

You can include simple or complex queries on the data stream without any coding to obtain refined results in the output.

1. Open a pipeline in the **Pipeline Editor**.
2. Right-click the stage after which you want to add a query stage, click **Add a Stage**, and then select **Query**.
3. Enter a **Name** and **Description** for the Query Stage.
4. Click **Save**.

5.1.2 Adding a Filter to a Query Stage

You can add filters in a pipeline to obtain more accurate streaming data.

To add a filter:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required query stage.
3. Navigate to the **Filters** tab.
4. Click **Add a Filter**.
5. Select the required column and a suitable operator and value.

   You can also calculate fields within filters.

> **Note:**
>
> IN operator is available as an operator in the drop-down list. This operator is not supported for `Interval`, `Interval YM`, `Timestamp`, and `SDO Geometry` datatypes. You can use the IN filter to refer to a column in a database table. When you change the database column values at runtime, the pipeline picks up the latest values from the DB column, without republishing the pipeline.

6. Click **Add a Condition** to add and apply a condition to the filter.
7. Click **Add a Group** to add nested conditions.
8. Repeat these steps for as many filters, conditions, or groups as you want to add.

   You can create blocks without adding condition expression, which you can add at any later stage.

   Link the blocks using AND/ OR

   Define complex conditions.

   Example:

```
IF (
    ((op_type == 'I') AND (after_SHHOLD is not null))
    OR  (
            ((before_SHHOLD is null) OR (before_SHHOLD = "  "))
            AND (
                after_SHHOLD is not null
            )
    )
)
```

5.1.3 Adding a Summary to a Query Stage

To add a summary:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required query stage and click the **Summaries** tab.
3. Click **Add a Summary**.
4. Select the suitable function and the required column.
5. Repeat the above steps to add as many summaries you want.

5.1.4 Adding a Summary with Group By

To add a group by:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required query stage and click the **Summaries** tab.
3. Click **Add a Group By**.
4. Click **Add a Field** and select the column on which you want to group by.

When you create a group by, the live output table shows the group by column alone by default. Turn ON **Retain All Columns** to display all columns in the output table.

You can add multiple group by's.

5.1.5 Adding a Query Group Stage

A query group is a combination of summaries (aggregation functions), group-bys, filters and a range window. Different query groups process your input in parallel and the results are combined in the query group stage output. You can also define input filters that process the incoming stream before the query group logic is applied, and result filters that are applied on the combined output of all query groups together.

A query group stage of the stream type applies processing logic to a stream. It is in essence similar to several parallel query stages grouped together for the sake of simplicity.

A query group stage of the table type can be added to a stream containing transactional semantic. For example, change data capture stream produced by the Oracle GoldenGate BigData plugin. The stage of this type will recreate the original database table in memory using the transactional semantics contained in the stream. You can then apply query groups to this table in memory, to run real-time analytics on your transactional data, without affecting the performance of your database.

5.1.5.1 Adding Query Group: Stream

You can apply aggregate functions with different groupbys and window ranges to your streaming data.

To add a query group stage of type stream:

1. Open a pipeline in the **Pipeline Editor**.
2. Right-click the stage after which you want to add a query group stage, click **Add a Stage**, select **Query Group**, and then **Stream**.

   You can add a query stage group only at the end of the pipeline.
3. Enter a name and a description for the query group stage of the type stream and click **Save**.

   The query group stage of the type stream appears in the pipeline.
4. On the **Input Filters** tab, click **Add a Filter**. See Adding a Filter to a Query Stage.

   These filters process data before it enters the query group stage. Hence, you can only see fields of the original incoming shape.
5. On the **Groups** tab, click **Add a Group**. A group can consist one or many of summaries, filters, and group bys.

   See Adding a Summary to a Query Stage and Adding a Summary with GroupBy.
6. Repeat the previous step to add as many groups as you want.
7. On the **Result Filters** tab, click **Add a Filter** to filter the results.

   These filters process data before it exits the query group stage. Hence, you can see a combined set of fields in the outgoing shape.
8. On the **Visualizations** tab, click **Add a Visualization** and add the required type of visualization. See Adding Chart Visualizations.

5.1.5.2 Adding Query Group: Table

You can apply aggregate functions with different groupbys and window ranges to a database table data recreated in memory.

To add a query group stage of the type table:

1. Open a pipeline in the **Pipeline Editor**.
2. Right-click the stage after which you want to add a query group stage, click **Add a Stage**, select **Query Group**, and then **Table**.
3. Enter a name and a description for the Query Group Table and click **Next**.
4. On the **Transactions Settings** screen, select a column in the **Transaction Field** drop-down list.

   The transaction column is a column from the output of the previous stage that carries the transaction semantics (insert/update/delete). Make sure that you use the values that correspond to your change data capture dataset. The default values work for Oracle GoldenGate change data capture dataset.
5. On the **Field Mappings** screen, select the columns that carry the before and after transaction values from the original database table. For example, in case of Oracle GoldenGate, the before and after values have `before_` and `after_` as prefixes, respectively. Specify a column as primary key in the table.
6. Click **Save** to create a query group stage of the type table.

   You can see the table configuration that you have specified while creating the table stage in the **Table Configuration** tab.
7. On the **Input Filters** tab, click **Add a Filter**. See Adding a Filter to a Query Stage.
8. On the **Groups** tab, click **Add a Group**. A group can consist one or many of summaries, filters, and groupbys.

   See Adding a Summary to a Query Stage and Adding a Summary with GroupBy.
9. Repeat the previous step to add as many groups as you want.
10. On the **Result Filters** tab, click **Add a Filter** to filter the results.
11. On the **Visualizations** tab, click **Add a Visualization** and add the required type of visualization. See Adding Chart Visualizations.

5.1.6 Adding a Rule Stage

Using a rule stage, you can add the IF-THEN logic to your pipeline. A rule is a set of conditions and actions applied to a stream. There is no specific sequence to add rules.

To add a rule stage:

1. Open a pipeline in the **Pipeline Editor**.
2. Right-click the stage after which you want to add a rule stage, click **Add a Stage**, and then select **Rule**.
3. Enter a **Name** and **Description** for the rule stage.
4. Click **Add a Rule**.
5. Enter **Rule Name** and **Description** for the rule and click **Done** to save the rule.

6. Select a suitable condition in the **IF** statement, **THEN** statement, and click **Add Action** to add actions within the business rules.

   Actions can also be expressions. For example, `SET Revenue TO =-Revenue`, will convert the current value of Revenue to a negative number.

   Expressions must always start with a '=' sign. For a constant text value, just type in the text. For example, `SET CustomerType TO GOLD`.

The rules are applied to the incoming events one by one and actions are triggered if the conditions are met.

5.1.7 Adding a Pattern Stage

A pattern is a template of an Oracle GoldenGate Stream Analytics application, with a business logic built into it. You can create pattern stages within the pipeline. Patterns are not stand-alone artifacts, they need to be embedded within a pipeline.

For detailed information about the various type of patterns, see Transforming and Analyzing Data using Patterns.

To add a pattern stage:

1. Open a pipeline in the **Pipeline Editor**.
2. Right-click the stage after which you want to add a pattern stage, click **Add a Stage**, and then select **Pattern**.
3. Choose the required pattern from the list of available patterns.
4. Enter a **Name** and **Description** for the pattern stage.

   The selected pattern stage is added to the pipeline.
5. Click **Parameters** and provide the required values for the parameters.
6. Click **Visualizations** and add the required visualizations to the pattern stage.

5.1.8 Adding a Scoring Stage

To add a scoring stage:

1. Open the required pipeline in Pipeline Editor.
2. Right-click the stage after which you want to add a scoring stage, click **Add a Stage**, and then select **Scoring**.
3. Enter a meaningful name and suitable description for the scoring stage and click **Save**.
4. In the stage editor, enter the following details:
   a. **Model name**: Select the predictive model that you want to use in the scoring stage
   b. **Model Version**: Select the version of the predictive model
   c. **Mapping**: Select the corresponding model fields that appropriately map to the stage fields

You can add multiple scoring stages based on your use case.

5.1.9 Adding a Target Stage

To add a target stage:

1. Open the required pipeline in Pipeline Editor.
2. Right-click the stage after which you want to add a target stage, click **Add a Stage**, and then select **Target**.
3. Enter a name and suitable description for the target.
4. Click **Save**.

For more information on creating different target types, see #unique_200.

5.1.10 Adding a Custom CQL Stage

To add a custom stage:

1. Open the required pipeline in Pipeline Editor.
2. Right-click the stage after which you want to add a custom stage. Click **Add a Stage**, and **Custom**, and then select **Custom CQL**.
3. Enter a name and suitable description for the custom stage and click **Save**.
4. Type your custom CQL query in the right pane of the pipeline editor.

5.4 Applying Functions to Create a New Column

You can perform calculations on the data streaming in the pipeline, and also add new fields into the stream using in-built functions of the Expression Builder.

To launch the Expression Builder, click **fx** in the **Live Output** table.

> **Note:**
>
> Currently, you can use expressions only within a query stage.

Adding a Constant Value Column

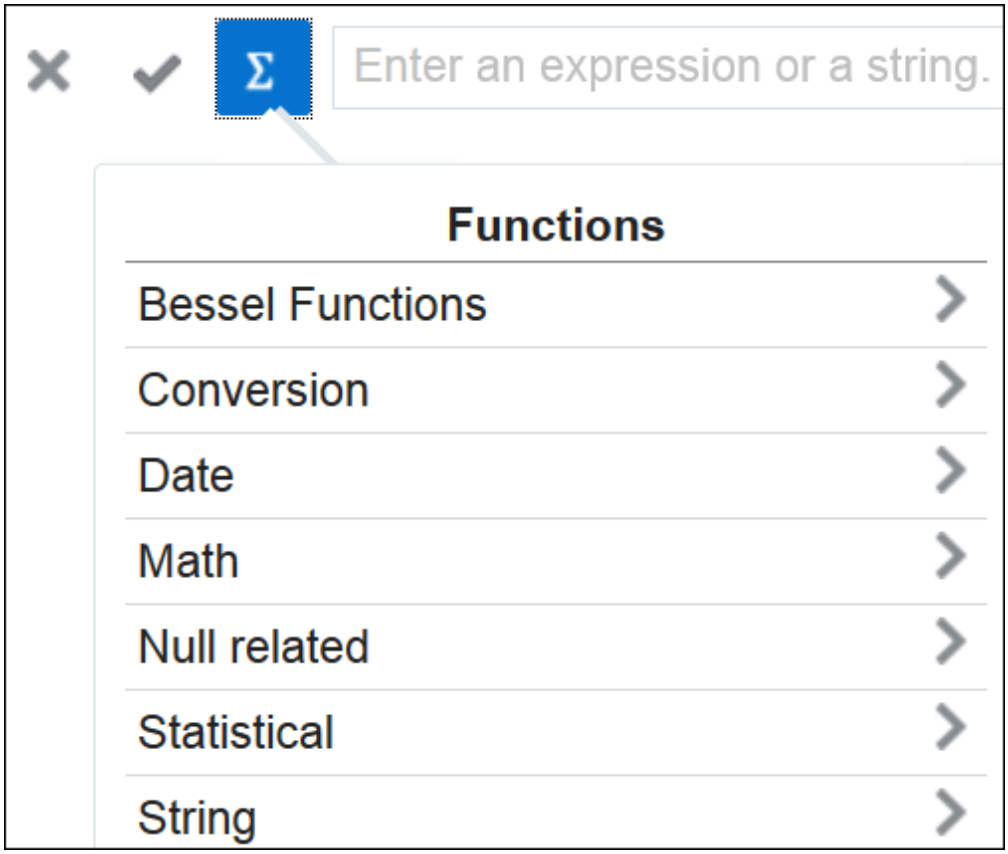
A constant value is a simple string or number. No calculation is performed on a constant value. Enter a constant value directly in the expression builder to add it to the live output table.

Using Functions

You can select a CQL Function from the list of available functions and select the input parameters. Make sure to begin the expression with "=". Click **Apply** to apply the function to the streaming data.

Example expression using functions:

```
=float((CanceledOrdersFloat/NewOrdersFloat) * 100.0)
```



You can see custom functions in the list of available functions when you add/import a custom jar in your pipeline.

For a list of supported functions, see #unique_202 .

5.4.1 Using Bessel Functions

The mathematical cylinder functions for integers are known as Bessel functions.

The following Bessel functions are supported in this release:

Function Name	Description
`BesselI0(x)`	Returns the modified Bessel function of order 0 of the double argument as a double
`BesselI0_exp(x)`	Returns the exponentially scaled modified Bessel function of order 0 of the double argument as a double
`BesselI1(x)`	Returns the modified Bessel function of order 1 of the double argument as a double
`BesselI1_exp(x)`	Returns the exponentially scaled modified Bessel function of order 1 of the double argument as a double
`BesselJ(x,x)`	Returns the Bessel function of the first kind of order n of the argument as a double
`BesselK(x,x)`	Returns the modified Bessel function of the third kind of order n of the argument as a double
`BesselK0_exp(x)`	Returns the exponentially scaled modified Bessel function of the third kind of order 0 of the double argument as a double

Function Name	Description
`BesselK1_exp(x)`	Returns the exponentially scaled modified Bessel function of the third kind of order 1 of the double argument as a double
`BesselY(x)`	Returns the Bessel function of the second kind of order n of the double argument as a double

5.4.1.1 BesselI0

Returns the modified Bessel function of order 0 of the input argument.

The input arguments can be one of the following data types: double, float.

Returned value type will be double.

Function	Result
`besselI0(65)`	8.403039845625433E26
`besselI0(3125.2)`	1.07389541368045088E17

5.4.1.2 BesselI0_exp

Returns the exponentially scaled modified Bessel function of order 0 of the double argument as a double.

The input argument can be one of the following data type: double, integer, float. Returned value type will be double.

Function	Result
`besselI0_exp(1451.44)`	8.113723742037748E23

5.4.1.3 BesselI1(value1)

Function returns the modified Bessel function of order 1 of the double argument.

The input arguments can be one of the following data types: double, integer, float.

The returned value type will be double.

Function	Result
`besselI1(432.98)`	2.1043808863643512E186
`besselI1(31)`	2.055972795294565E12

5.4.1.4 BesselI1_exp(value1)

Function returns the exponentially scaled modified Bessel function of order 1, of the input argument.

The input arguments can be one of the following types: double, integer, float.

Returned value type will be double.

Function	Result
`besselI1_exp(99)`	0.03994284829937756

5.4.1.5 BesselK0_exp(value1)

Function returns the exponentially scaled modified Bessel function of the third kind of order 0.

Input value can be one of the following types: double, integer, float.

Returned value type will be double.

Function	Result
`Besselk0_exp(3.6)`	0.6404559726736455

5.4.1.6 BesselIK1_exp(value1)

Function returns the exponentially scaled modified Bessel function of the third kind of order 1.

Input value can be one of the following types: double, integer, float.

Returned value type will be double.

Function	Result
`BesselIK1_exp(72)`	0.14847048263652857
`BesselIK1_exp(3.6)`	0.7244606719817783

5.4.1.7 BesselY(value1, value2)

Function returns the Bessel function of the second kind of order n of the input argument.

Value 1 can be of the following types: integer.

Value 2 can be of the following types: double, integer, float.

Returned value type will be double.

Function	Result
`BesselY(30,2.2)`	-1.6816755062290252E29

5.4.1.8 BesselJ(value1, value2)

Function returns the Bessel function of the first kind of order n of the argument.

The input arguments can be one of the following data types:

- Value1 can be one of the following types: integer.
- Value 2 can be one of the following types: double, integer, float.

Returned value type will be double.

Function	Result
`besselJ(4,3.3)`	0.1742753869717833

5.4.1.9 BesselK(value1,value2)

Function returns the modified Bessel function of the third kind of order n of the input argument.

Value1 can be one of the following types: integer.

Value 2 can be one of the following types: double, integer, float.

Returned value type will be double.

Function	Result
`BesselK(30,2)`	4.271125754887687E30

5.4.2 Using Conversion Functions

The conversion functions help in converting values from one data type to other.

The following conversion functions are supported in this release:

Function Name	Description
`bigdecimal(value1)`	Converts the given value to bigdecimal
`boolean(value1)`	Converts the given value to logical
`date(value1,value2)`	Converts the given value to datetime
`double(value1)`	Converts the given value to double
`float(value1)`	Converts the given value to float
`int(value1)`	Converts the given value to integer
`long(value1)`	Converts the given value to long
`string(value1,value2)`	Converts the given value to string

5.4.2.1 bigdecimal(value1)

Converts the input argument value to big decimal. The input argument can be one of the following data types: big integer, number, double, integer, text, float. Returned value type will be number.

Function	Result
`bigdecimal(60)`	6E+1
`bigdecimal(32)`	32

5.4.2.2 boolean(value1)

Converts the input argument value to logical. The input argument can be one of the following data type: big integer or integer. Returned value type will be Boolean.

Examples

Function	Result
`boolean(5)`	TRUE
`boolean(0)`	FALSE
`boolean(NULL)`	TRUE
`boolean()`	TRUE
`boolean(-5)`	TRUE

5.4.2.3 double(value1)

Converts the input argument value to double. The input argument can be one of the following data types: integer, big integer, double, text or float. Returned value type will be double.

Examples

Function	Result
`double(3.1406)`	3.1405999660491943
`double(1234.56)`	1234.56005859375

5.4.2.4 float(value1)

Converts the input argument value to float. The input argument can be one of the following data types: integer, big integer, double, text or float. Returned value will be a single-precision floating-point number.

Examples

Function	Result
`float(1.67898989395)`	1.6789899
`float(1.796709289)`	1.7967093
`float(12.60508090750)`	12.605081

5.4.2.5 int(value1)

Converts the input argument value to integer. The input argument can be one of the following types: integer, text. Returned value type will be integer.

Function	Result
`int(50/3)`	16

5.4.2.6 long()

Converts the input argument value to long. The input argument can be one of the following types: big integer, integer, text, float, timestamp. Returned value type will be big integer.

Function	Result
`long(5039505078907524)`	5039505078907524
`long(22)`	22

5.4.2.7 string(value1, value2)

Conversion to string.

Value 1 can be one of the following types: intervalym, big integer, number, boolean, double, interval, integer, float, timestamp. Required.

Value 2 is output date format. It's required argument for value1 of type timestamp. Value can be one of the following types: text. Optional.

Returned value will be of type text.

Function	Result
`string(transaction_time,"hh-mm-ss")` , where transaction_time is 12/19/2016 12:23:04	12-23-04
`string(transaction_time,"M-DD-YY")` , where transaction_time is 12/19/2016 12:23:04	12-19-16

5.4.3 Using Date Functions

The following date functions are supported in this release:

Function Name	Description
`day(date)`	Returns day of the date
`eventtimestamp()`	Returns event timestamp from stream
`hour(date)`	Returns hour of the date
`minute(date)`	Returns minute of the date
`month(date)`	Returns month of the date
`nanosecond(date)`	Returns nanosecond of the date
`second(date)`	Returns second of the date
`systimestamp()`	Returns the system's timestamp on which the application is running
`timeformat(value1,value2)`	Returns the provided timestamp in required time format
`year(date)`	Returns year of the date

5.4.3.1 Acceptable Formats for Timestamp Values

This sections lists the acceptable formats for timestamp values in Oracle Stream Analytics.

Format	Example Values
MM/dd/yyyy HH:mm:ss.SSSS	3/21/2018 11:14:23.1111
MM/dd/yyyy HH:mm:ss.SSS	3/21/2018 11:14:23.111
MM/dd/yyyy HH:mm:ss.SS	3/21/2018 11:14:23.11
MM/dd/yyyy HH:mm:ss.S	3/21/2018 11:14:23.1
MM/dd/yyyy HH:mm:ss	3/21/2018 11:14:23
MM/dd/yyyy HH:mm	3/21/2018 11:14
MM/dd/yyyy HH	3/21/2018 11
MM/dd/yyyy	3/21/2018
MM-dd-yyyy HH:mm:ss.SSSS	11-21-2018 11:14:23.1111
MM-dd-yyyy HH:mm:ss.SSS	11-21-2018 11:14:23.111
MM-dd-yyyy HH:mm:ss.SS	11-21-2018 11:14:23.11
MM-dd-yyyy HH:mm:ss.S	11-21-2018 11:14:23.1
MM-dd-yyyy HH:mm:ss	11-21-2018 11:14:23
MM-dd-yyyy HH:mm	11-21-2018 11:14
MM-dd-yyyy HH	11-21-2018 11
MM-dd-yyyy	11-21-2018
dd-MMM-yy hh.mm.ss.SSSSSS a	11-Jan-18 11.14.23.111111 AM
dd-MMM-yy hh.mm.ss.SSSS	11-Jan-18 11.14.23.1111
dd-MMM-yy hh.mm.ss.SSS	11-Jan-18 11.14.23.111
dd-MMM-yy hh.mm.ss.SS	11-Jan-18 11.14.23.11
dd-MMM-yy hh.mm.ss.S	11-Jan-18 11.14.23.1
dd-MMM-yy hh.mm.ss	11-Jan-18 11.14.23
dd-MMM-yy hh.mm	11-Jan-18 11.14
dd-MMM-yy hh	11-Jan-18 11
dd-MMM-yy	11-Jan-18
dd/MMM/yy	15/MAR/18
yyyy-MM-dd HH:mm:ss.SSSSSS	2018-03-5 15:16:0.756000 +5:30, 2018-03-5 15:16:0.756000
yyyy-MM-dd HH.mm:.ss.SSSSSS	2018-03-5 15.16.0.756000 +5:30, 2018-03-5 15.16.0.756000
yyyy-MM-dd HH:mm:ss	2018-03-5 15:16:0; 2018-03-5 15:16:0 +5:30
yyyy-MM-dd HH.mm.ss	2018-03-5 15.16.0; 2018-03-5 15.16.0 +5:30
yyyy-MM-dd HH:mm	2018-03-5 15:16; 2018-03-5 15:16 +5:30
yyyy-MM-dd HH.mm	2018-03-5 15.16; 2018-03-5 15.16 +5:30
yyyy-MM-dd HH	2018-03-5 15
yyyy-MM-dd	2018-03-5
HH:mm:ss	11:14:14 PST
yyyy-MM-dd'T'HH:mm:ss'.'SSS	2018-03-04T12:08:56.235
yyyy-MM-dd'T'HH:mm:ss'.'SSSZ	2018-03-04T12:08:56.235-0700

Format	Example Values
`yyyy-MM-dd'T'HH:mm:ss'.'SSSz`	2018-03-04T12:08:56.235 PDT
`yyyy-MM-dd'T'HH:mm:ss`	2018-03-04T12:08:56
`yyyy-MM-dd'T'HH:mm:ssZ`	2018-03-04T12:08:56-0700
`yyyy-MM-dd'T'HH:mm:ssz`	2018-03-04T12:08:56 PDT

5.4.3.2 Day(date)

`day(date)` function takes as an argument any one of the following data types: time interval or timestamp. The returned value represents the day in the timestamp represented by this date object. Returns a big integer indicating the day represented by this date.

Examples

Function	Result
`day(transaction-time)`, where transaction_time is 12/19/2016 12:22:48	19

5.4.3.3 eventtimestamp(value1)

Event timestamp from stream.

Returned value will be of type timestamp.

Function	Result
`eventtimestamp()`	4/4/2019 16:40:57

,

5.4.3.4 hour(date)

`hour(date)` function takes as an argument any one of the following data types: time interval or timestamp. The returned value represents the hour in the time represented by this date object. Returns a big integer indicating the hour of the time represented by this date.

Examples

Function	Result
`hour(12/06/17 09:15:22 AM)`	09
`hour(2015:07:21 12:45:35 PM)`	12

5.4.3.5 minute(date)

`minute(date)` function takes as an argument any one of the following data types: time interval or timestamp. The returned value represents the minutes in the time represented by this date object. Returns a big integer indicating the minutes of the time represented by this date.

Examples

Function	Result
`minute(12/06/17 09:15:22 AM)`	15
`minute(2015:07:21 12:45:35 PM)`	45

5.4.3.6 month(date)

`month(date)` function takes as an argument any one of the following data types: time interval or timestamp. The returned value represents the month of the year that contains or begins with the instant in time represented by this date object. Returns a big integer indicating the month of the year represented by this date.

Examples

Function	Result
`month(12/06/17 09:15:22 AM)`	12
`month(2017:09:23 11:20:25 AM)`	9

5.4.3.7 nanosecond(value1)

Extracts and returns the current fractional part of second from date.

Value 1 can be one of the following types: timestamp.

Returned value will be of type big integer.

Function	Result
`nanosecond(transaction_time)`, 12/19/2016 12:22:57	719978080

5.4.3.8 systemtimestamp(value1)

Returns the current system time.

Returned value will be of type timestamp.

Function	Result
`systemtimestamp()`	4/4/2019 17:06:14
,	

5.4.3.9 timeformat(value1, value2)

Event Formatted time.

Value 1 can be one of the following types: timestamp.

Value 2 can be one of the following types: text.

Returned value will be of type text.

Function	Result
`timeformat(transaction_time, "M-dd-yy"),` where calc is 12/19/2016 12:22:46	12-19-16
`timeformat(transaction_time, "DAY"),` where transaction_time is 12/19/2016 12:22:44	Monday

5.4.3.10 Year(date)

`year(date)` function takes as an argument any one of the following data types: time interval or time stamp. The returned value represents the year of the instant in time represented by this date object. Returns a big integer indicating the year represented by this date.

Examples

Function	Result
`year(12/06/17 09:15:22 AM)`	17
`year(2015:07:21 12:45:35 PM)`	2015

5.4.4 Using Geometry Functions

The Geometry functions allow you to convert the given values into a geometrical shape.

The following interval functions are supported in this release:

Function Name	Description
`CreatePoint(lat,long,SRID)`	Returns a 2–dimensional point type geometry from the given latitude and longitude. The default SRID is 8307. The return value is of the datatype `sdo geometry`.
`distance(lat1,long1,lat2,long2,SRID)`	Returns distance between the first set of latitude, longitude and the second set of latitude, longitude values. The default SRID is 8307. **Note:** Only SRID 8307 is supported in the current release. The return value is of the datatype `double`.

5.4.4.1 CreatePoint(value1, value2, value3)

createPoint(lat,long,SRID) - Function Returns a 2d point type geometry, default SRID is 8307.

Value 1: Latitude - Value can be one of the following types: number, double, float. Required.

Value 2: Longitude - Value can be one of the following types: number, double, float.

Value 3: SRID - Value can be one of the following types: integer.

Returned value type will be sdo geometry.

Function	Result
`CreatePoint(78995333342435,-122.4005650002481937,8307)`	point

5.4.4.2 distance(lat1, long1, lat2, long2,SRID)

Function Returns distance between lat1/long1 and lat2/long2, default SRID is 8307.

Value 1: Latitude1 - Value can be one of the following types: number, double, float.

Value 2: Longitude1 - Value can be one of the following types: number, double, float.

Value 3: Latitude2 - Value can be one of the following types: number, double, float.

Value 4: Longitude2 - Value can be one of the following types: number, double, float.

Value 5: SRID - Value can be one of the following types: integer.

Returned value will be of type double.

Function	Result
`distance(37.78371333337545, -122.4052500001069, 37.78371333337545, 37.78371333337545, 8307)`	1.1394718018250743E7

5.4.5 Using Interval Functions

The Interval functions help you in calculating time interval from given values.

The following interval functions are supported in this release:

Function Name	Description
`dsintervaltonum (c1 INTERVAL DAY TO SECOND, c2 char)`	Converts the given value to a numeric value. User must provide unit for the output numeric value as second argument to this function. Allowed values for the unit is: DAY,HOUR,MINUTE,SECOND
`numtodsinterval(n,interval_unit)`	Converts the given value to an `INTERVAL DAY TO SECOND` literal. The value of the `interval_unit` specifies the unit of `n` and must resolve to one of the string values: *DAY*, *HOUR*, *MINUTE*, or *SECOND*. The return value is of the datatype `interval`.
`numtoyminterval(n,interval_unit)`	Converts the given value to an `INTERVAL YEAR TO MONTH`literal. The value of the `interval_unit`specifies the unit of `n`and must resolve to one of the following string values: *YEAR*, *MONTH*.

Function Name	Description
`to_dsinterval(string)`	Converts a string in format `DD HH:MM:SS` into a `INTERVAL DAY TO SECOND` data type. The `DD` indicates the number of days between 0 to 99. The `HH:MM:SS` indicates the number of hours, minutes and seconds in the interval from 0:0:0 to 23:59:59.999999. The seconds part can accept upto six decimal places. The return value is of the datatype `interval`.
`to_yminterval(string)`	Converts a string in format `YY-MM` into a `INTERVAL YEAR TO MONTH` data type. The `YY`part indicates the number of years between 0 to 99. The `MM`part indicates the number of months between 0-11. The return value is of the datatype `interval`.
`ymintervaltonum(c1 INTERVAL YEAR TO MONTH, c2 char)`	Converts the given value to a numeric value. User must provide unit for the output numeric value as second argument to this function. Allowed values for the unit is: YEAR,MONTH

5.4.5.1 dsintervaltonum(value1, value 2)

DSINTERVALTONUM(c1 INTERVAL DAY TO SECOND, c2 char) - Function will convert interval value(c1) into a numeric value. User must provide unit for the output numeric value as second argument to this function. Allowed values for the unit is: DAY,HOUR,MINUTE,SECOND

Input value 1 can be one of the following types: interval.

Input value 2 can be one of the following types: text.

Returned value will be of type double.

Function	Result
`dsintervaltonum(calc_7, "MINUTE")` , 	301.0
`dsintervaltonum(calc_7, "DAY")`	5.016666666666667

5.4.5.2 numtodsinterval(value1, value2)

Function converts n to an INTERVAL DAY TO SECOND literal.The value for interval_unit specifies the unit of n and must resolve to one of the following string values:DAY,HOUR,MINUTE,SECOND.

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: text.

Returned value will be of type interval.

Function	Result
`numtodsinterval(34, "MONTH")`	2 yy 10 mm
`numtodsinterval(26.5, "HOUR")`	1 dd 2 hr 30 mm 0 sec

Function	Result
`numtodsinterval(1230, "MINUTE")`	00 dd 20 hr 30 mm 0 sec
`numtodsinterval(1, "DAY")`	1 dd 0 hr 0 mm 0 sec

5.4.5.3 numtoyminterval(value1, value 2)

NUMTOYMINTERVAL(n,interval_unit) - Function converts n to an INTERVAL YEAR TO MONTH literal. The value for interval_unit specifies the unit of n and must resolve to one of the following string values:YEAR, MONTH.

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: text.

Returned value will be of type intervalym.

Function	Result
`numtoyminterval(10.5, "YEAR")`	10 yy 6 mm
`numtoyminterval(34, "MONTH")`	2 yy 10 mm

5.4.5.4 to_dsinterval(value1)

Function converts a string in format 'DD HH:MM:SS' into a INTERVAL DAY TO SECOND data type. The DD part indicates the number of days between 0 to 99. The HH:MM:SS part indicates the number of hours, minutes and seconds in the interval from 0:0:0 to 23:59:59.999999. The second part can accept upto 6 decimal places.

Input value can be one of the following types: text.

Returned value will be of type interval.

Function	Result
`to_dsinterval("02 23:34:12")`	2 dd 23hr 34mm 12 sec

5.4.5.5 to_yminterval(value1)

Function converts a string in format 'YY-MM' into a INTERVAL YEAR TO MONTH data type.The YY part indicates the number of years between 0 to 99. The MM part indicates the number of months between 0-11.

Value can be one of the following types: text.

Returned value type will be intervalym.

Function	Result
`to_yminterval("94-3")`	94 yy 3 mm

5.4.5.6 ymintervaltonum(value1, value2)

Function converts interval value(c1) into a numeric value. You must provide the unit for the output numeric value as the second argument to this function. Allowed values for the unit are: YEAR,MONTH.

Value 1 can be one of the following types: intervalym.

Value 2 can be one of the following types: text.

Returned value type will be double.

Function	Result
`ymintervaltonum(94yy 5mm,"MONTH')`	1133.0

5.4.6 Using Math Functions

The math functions allow you to perform various mathematical operations and calculations ranging from simple to complex.

The following math functions are supported in this release:

Function Name	Description
`IEEEremainder(value1,value2)`	Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard
`abs(value1)`	Returns the absolute value of a number
`acos(value1)`	Returns arc cosine of a value
`asin(value1)`	Returns arc sine of a value
`atan(value1)`	Returns arc tangent of a value
`atan2(arg1,arg2)`	Returns polar angle of a point (`arg2`, `arg1`)
`binomial(base,power)`	Returns binomial coefficient of the base raised to the specified power
`bitMaskWithBitsSetFromTo(x)`	BitMask with BitsSet (From, To)
`cbrt(value1)`	Returns cubic root of the specified value
`ceil(value1)`	Rounds to ceiling
`copySign(value1,value2)`	Returns the first floating-point argument with the sign of the second floating-point argument
`cos(value1)`	Returns cosine of a value
`cosh(value1)`	Returns cosine hyperbolic of a value
`exp(x)`	Returns exponent of a value
`expm1(x)`	More precise equivalent of `exp(x)`; Returns 1 when x is around zero
`factorial(value1)`	Returns factorial of a natural number
`floor(value1)`	Rounds to floor
`getExponent(value1)`	Returns the unbiased exponent used in the representation of a double

Function Name	Description
`getSeedAtRowColumn(value1,value2)`	Returns a deterministic seed as an integer from a (seemingly gigantic) matrix of predefined seeds
`hash(value1)`	Returns an integer hashcode for the specified double value
`hypot(value1,value2)`	Returns square root of sum of squares of the two arguments
`leastSignificantBit(value1)`	Returns the least significant 64 bits of this UUID's 128 bit value
`log(value1,value2)`	Calculates the log value of the given argument to the given base, where `value 1` is the value and `value 2` is the base
`log1(value1)`	Returns the natural logarithm of a number
`log10(value1)`	Calculates the log value of the given argument to base 10
`log2(value1)`	Calculates the log value of the given argument to base 2
`logFactorial(value1)`	Returns the natural logarithm (base e) of the factorial of its integer argument as a double
`longFactorial(value1)`	Returns the factorial of its integer argument (in the range k >= 0 && k < 21) as a long
`maximum(value1,value2)`	Returns the maximum of 2 arguments
`minimum(value1,value2)`	Returns the minimum of 2 arguments
`mod(value1,value2)`	Returns modulo of a number
`mosttSignificantBit(value1)`	Returns the most significant 64 bits of this UUID's 128 bit value
`nextAfter(value1,value2)`	Returns the floating-point number adjacent to the first argument in the direction of the second argument
`nextDown(value1)`	Returns the floating-point value adjacent to the input argument in the direction of negative infinity
`nextUp(value1)`	Returns the floating-point value adjacent to the input argument in the direction of positive infinity
`Pow(m,n)`	Returns m raised to the nth power
`rint(value1)`	Returns the double value that is closest in value to the argument and is equal to a mathematical integer
`round(value1)`	Rounds to the nearest integral value
`Scalb(d,scaleFactor)`	Returns d × 2scaleFactor rounded as if performed by a single correctly rounded floating-point multiply to a member of the double value set
`signum(value1)`	Returns signum of an argument as a double value
`sin(value1)`	Returns sine of a value
`sinh(value1)`	Returns sine hyperbolic of a value
`sqrt(value1)`	Returns square root of a value
`stirlingCorrection(value1)`	Returns the correction term of the Stirling approximation of the natural logarithm (base e) of the factorial of the integer argument as a double
`tan(value1)`	Returns tangent of a value
`tanh(value1)`	Returns tangent hyperbolic of a value
`toDegrees(value1)`	Converts the argument value to degrees

Function Name	Description
`toRadians(value1)`	Returns the measurement of the angle in radians
`ulp(value1)`	Returns the size of an ulp of the argument

5.4.6.1 IEEEremainder(value1, value1)

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard : IEEEREMAINDER

Value 1 can be one of the following types: double.

Value 2 can be one of the following types: double

Returned value type will be double.

Function	Result
`IEEEremainder(8809,8808)`	-1.0

5.4.6.2 abs(value1)

Returns the Absolute value of the input argument.

Input value can be one of the following types: number, big integer, double, integer, float.

Returned value type will be the same as the input argument type.

Function	Result
`abs(1234.560789)`	1234.56078
`abs(0.67)`	0.6700000166893005

5.4.6.3 acos(value1)

Returns the Arc cosine of a value.

Value can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`acos(0.5)`	1.0471975511965979

5.4.6.4 asin(value1)

Computes the arc sine of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value type will be double.

Function	Result
`asin(0.5)`	0.5235987755982989

5.4.6.5 atan(value1)

Returns the arc tangent of the input value.

Input value can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`atan(34)`	1.5413930385908916

5.4.6.6 atan2

Returns the polar angle of a point (value2, value1).

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`atan2(8681.44, 8682.44)`	0.7853405725825559

5.4.6.7 binomial(base, power)

Returns the Binomial coefficient of the input base and power values.

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: big integer, integer.

Returned value will be of type double.

Function	Result
`binomial(8609.4, 38)`	5.955734227594785E104

5.4.6.8 bitMaskWithBitsSetFromTo(value1, value2)

Value 1 can be one of the following types: integer.

Value 2 can be one of the following types: integer.

Returned value will be of type double.

Function	Result
`bitMaskWithBitsSetFromTo(23, 23)`	8388608.0

5.4.6.9 cbrt()

Returns the cubic root of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value type will be double.

Function	Result
`cbrt(27)`	3

5.4.6.10 ceil()

Round to ceiling.

The input arguments can be one of the following data types: double, float.

Returned value type will be float.

Function	Result
`ceil(65)`	65.0

5.4.6.11 copySign()

Function returns the first floating-point argument with the sign of the second floating-point argument.

Value1 can be one of the following types: double, float.

Returned value type will be double, float.

Function	Result
`copySign(3.0, -4.0))`	-3.0

5.4.6.12 cos(value1)

Returns the cosine of a value

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value type will be double.

Function	Result
`cos(7740.8)`	0.9964325256163951

5.4.6.13 cosh(value1)

Returns the Cosine hyperbolic of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Function	Result
`cosh(0.5)`	1.1276259652063807

5.4.6.14 exp(value1, value2)

Returns the exponent of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`exp(10)`	22026.465794806718

5.4.6.15 expm1(value1)

Returns the more precise equivalent of Exp(x)-1 when x is around zero.

Value can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`expm1(0.7)`	1.0137526834646737

5.4.6.16 factorial(value1)

Returns the Factorial of a natural.

Value 1 can be one of the following types: integer.

Returned value type will be double.

Function	Result
`factorial(6)`	720.0

5.4.6.17 floor(value1)

Value can be one of the following types: big integer, double, integer, float.

Returned value will be of type float.

Function	Result
`floor(0.567)`	0.0

5.4.6.18 GetExponent(value1)

Function returns the unbiased exponent used in the representation of a double.

Value 1 can be one of the following types: double, float.

Returned value will be of type integer.

Function	Result
`getExponent(10.0)`	3.0

5.4.6.19 getSeedAtRowColumn(value1, value2)

Returns a deterministic seed as an integer from a (seemingly gigantic) matrix of predefined seeds : GETSEEDATROWCOLUMN

Value 1 can be one of the following types: integer.

Value 2 can be one of the following types: integer.

Returned value will be of type integer.

Function	Result
`getSeedAtRowColumn(48, 2)`	443210610

5.4.6.20 hash(value1)

Function returns an integer hashcode for the specified value.

Value can be one of the following types: big integer, double, integer, float.

Returned value will be of type integer.

Function	Result
`hash(8.1)`	1.33589862E9

5.4.6.21 hypot(value1, value2)

Square root of sum of squares of the two arguments.

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: big integer, double, integer, float.

Returned value type will be double.

Function	Result
`hypot(2,4)`	4.47213595499958

5.4.6.22 LeastSignificantBit(value1)

Method is used to return the least significant 64 bits of this UUID's 128 bit value.

Value 1 can be one of the following types: integer.

Returned value will be same as the input argument.

Function	Result
`LeastSignificantBit(2)`	1.0

5.4.6.23 log(value1, value2)

Logarithm(base, arg)

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`log(20,3)`	0.3667257913420846

5.4.6.24 log1(value1)

Function returns the natural logarithm of a number.

Value 1 can be one of the following types: double, integer, float.

Returned value will be of type double.

Function	Result
`log1(20)`	2.995732273553991

5.4.6.25 log10(value1)

Logarithm(10, arg)

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`log10(20)`	1.301029995663981

5.4.6.26 log2(value1)

Logarithm(2, arg)

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`log2(20)`	4.321928094887362

5.4.6.27 logFactorial(value1)

Function returns the natural logarithm (base e) of the factorial of its integer argument as a double.

Value 1 can be one of the following types: double, integer, float.

Returned value will be of type double.

Function	Result
`logFactorial(20)`	42.335616460753485

5.4.6.28 long()

Converts the input argument value to long. The input argument can be one of the following types: big integer, integer, text, float, timestamp. Returned value type will be big integer.

Function	Result
`long(5039505078907524)`	5039505078907524
`long(22)`	22

5.4.6.29 longFactorial(value1)

Function returns the natural logarithm (base e) of the factorial of its integer argument as a double.

Value 1 can be one of the following types: double, integer, float.

Returned value will be of type double.

Function	Result
`longFactorial(10)`	15.104412573075516

5.4.6.30 minimum(value1, value2)

Returns the minimum of two arguments. The first argument is a value to compare with the second argument's value and can be any one of the following data type: big integer, double, interval, integer, float. The second argument is a value to compare with the first argument's value and can be any one of the following data type: big integer, double, interval, integer, float.

Examples

Function	Result
`minimum(16324, 16321)`	16321
`minimum(3.16, 3.10)`	3.10

> **Note:**
>
> If the user provides two different data types as arguments, then Stream Analytics does implicit conversion to convert one argument to the other argument's type.

5.4.6.31 mod(value1, value2)

Functions returns Modulo of a number

Value 1 can be one of the following types: big integer, double, integer, float.

Value 2 can be one of the following types: big integer, double, integer, float.

Returned value will be of the same type as the first argument.

Function	Result
`mod(10,3)`	1.0

5.4.6.32 mostSignificantBit(value1)

Function returns the most significant 64 bits of this UUID's 128 bit value .

Value 1 can be one of the following types: integer.

Returned value will be of the same type as the first argument.

Function	Result
`mostSignificantBit(10)`	3.0

5.4.6.33 nextAfter(value1, value2)

Function returns the floating-point number adjacent to the first argument in the direction of the second argument.

Value 1 can be one of the following types: double, float.

Value 2 can be one of the following types: double, float.

Returned value will be the same type as the first argument.

Function	Result
`nextAfter()`	

5.4.6.34 nextDown(value1, value2)

Function returns the floating-point number adjacent to the first argument in the direction of the second argument.

Value 1 can be one of the following types: double, float.

Value 2 can be one of the following types: double, float.

Returned value will be the same type as the first argument.

Function	Result
`nextDown()`	

5.4.6.35 nextUp(value1)

Function returns the floating-point number adjacent to the first argument in the direction of the second argument.

Value 1 can be one of the following types: double, float.

Returned value will be the same type as the first argument.

Function	Result
`nextUp()`	

5.4.6.36 pow(value1, value2)

Power function returns m raised to the nth power.

Value 1 can be one of the following types: double, integer, float.

Value 2 can be one of the following types: double, integer, float.

Returned value will be of type double.

Function	Result
`pow(12,2)`	144

5.4.6.37 rint(value1)

Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

Value can be one of the following types: double.

Returned value will be of type double.

Function	Result
`rint()`	

5.4.6.38 round(value1)

Rounds the argument value to the nearest integer value. The input argument can be of the following data types: big integer, double, integer, float.

Examples

Function	Result
`round(7.16)`	7
`round(38.941)`	39

Function	Result
`round(3.5)`	4

5.4.6.39 scalb(

Function Return d × 2scaleFactor rounded as if performed by a single correctly rounded floating-point multiply to a member of the double value set.

Value 1 can be one of the following types: double, float.

Value 2 can be one of the following types: integer.

Returned value will be the same type as the first argument.

Function	Result
`scalb(10.0,2)`	40.0

5.4.6.40 signum(value1)

Signum of an argument as a double value.

Value 1 can be one of the following types: number, big integer, double, integer, float.

Returned value will be of type integer.

Function	Result
`signum(10)`	1.0

5.4.6.41 sin(value1)

Returns the sine of the input value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`sin(7740.8)`	0.08419864005868474

5.4.6.42 sinh(value1)

Returns the Sine hyperbolic of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`sinh(0.5)`	0.5210953054937474

5.4.6.43 sqrt(value1)

Returns the Square root of the input value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of the type double.

Function	Result
`sqrt(7434.73)`	86.22488040003303

5.4.6.44 stirlingCorrection(value1)

Returns the correction term of the Stirling approximation of the natural logarithm (base e) of the factorial of the integer argument as a double: STIRLINGCORRECTION

Value 1 can be one of the following types: integer.

Returned value will be of the type double.

Function	Result
`stirlingCorrection(70)`	0.0011904680924708464

5.4.6.45 tan(value1)

Returns the Tangent of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`tan(60)`	0.320040389379563

5.4.6.46 tanh(value1)

Returns the Tangent hyperbolic of a value.

Value 1 can be one of the following types: big integer, double, integer, float.

Returned value will be of type double.

Function	Result
`tanh(1)`	0.7615941559557649

5.4.6.47 toDegrees(value1)

Converts the argument value to degrees. The input argument is an angle in radians and can be of type double. The returned value will be the measurement of the angle in degrees and is of type double.

Examples

Function	Result
`toDegrees(3.14)`	180.0
`toDegrees(0.785)`	45.0

5.4.6.48 toRadians(value1)

Converts the argument value to radians. The input argument is an angle in degrees and can be of type double. The returned value will be the measurement of the angle in radians and is of type double.

Examples

Function	Result
`toRadians(180.0)`	3.14
`toRadians(45.0)`	0.785

5.4.6.49 ulp(value1)

Returns the returns the size of an ulp of the argument: ULP.

The input arguments can be one of the following data types: double, float.

Returned value type will be the same as the input value type.

Function	Result
`ulp(1451.54)`	2.2737367544323206E-13

5.4.7 Using Null-related Functions

The following null-related functions are supported in this release:

Function Name	Description
`nvl(value1,value2)`	Replaces null with a value of the same type

5.4.7.1 nvl(value1, value2)

`nvl` lets you replace null (returned as a blank) with a value of the same type as the first argument. For example, in a list of employees and commission, you can substitute *Not Applicable* if the employee receives no commission using the `nvl(value1,value2)` function as `nvl(Not Applicable,Commission)`.

Example

Function	Result
`nvl(Not Applicable,Commission)`	Not Applicable

5.4.8 Using Statistical Functions

Statistical functions help you in calculating the statistics of different values.

The following statistical functions are supported in this release:

Function Name	Description
`beta1(value1,value2,value3)`	Returns the area from zero to `value3` under the beta density function
`betaComplemented(value1,value2,value3)`	Returns the area under the right hand tail (from `value3` to infinity) of the beta density function
`binomial2(value1,value2,value3)`	Returns the sum of the terms 0 through `value1` of the Binomial probability density. All arguments must be positive.
`binomialComplemented(value1,value2,value3)`	Returns the sum of the terms `value1+1` through `value2` of the binomial probability density. All arguments must be positive.
`chiSquare(value1,value2)`	Returns the area under the left hand tail (from 0 to `value2`) of the chi square probability density function with `value1` degrees of freedom. The arguments must both be positive.
`chiSquareComplemented(value1,value2)`	Returns the area under the right hand tail (from `value2` to infinity) of the chi square probability density function with `value1` degrees of freedom. The arguments must both be positive.
`errorFunction(value1)`	Returns the error function of the normal distribution
`errorFunctionComplemented(value1)`	Returns the complementary error function of the normal distribution
`gamma(value1,value2,value3)`	Returns the gamma function of the arguments
`gammaComplemented(value1,value2,value3)`	Returns the integral from `value3` to infinity of the gamma probability density function
`incompleteBeta(value1,value2,value3)`	Returns the incomplete beta function evaluated from zero to `value3`
`incompleteGamma(value1,value2)`	Returns the incomplete gamma function
`incompleteGammaComplement(value1,value2)`	Returns the complemented incomplete gamma function
`logGamma(value1)`	Returns the natural logarithm of the gamma function
`negativeBinomial(value1,value2,value3)`	Returns the sum of the terms 0 through `value1` of the negative binomial distribution. All arguments must be positive.
`negativeBinomialComplemented(value1,value2,value3)`	Returns the sum of the terms `value1+1` to infinity of the negative binomial distribution. All arguments must be positive.
`normal(value1,value2,value3)`	Returns the area under the normal (Gaussian) probability density function, integrated from minus infinity to `value1` (assumes mean is zero, variance is one)

Function Name	Description
`normalInverse(value1)`	Returns the value for which the area under the normal (Gaussian) probability density function is equal to the argument `value1` (assumes mean is zero, variance is one)
`poisson(value1,value2)`	Returns the sum of the first `value1` terms of the Poisson distribution. The arguments must both be positive.
`poissonComplemented(value1, value2)`	Returns the sum of the terms `value1+1` to infinity of the poisson distribution
`studentT(value1,value2)`	Returns the integral from minus infinity to `value2` of the Student-t distribution with `value1` > 0 degrees of freedom
`studentTInverse(value1,value2)`	Returns the value, for which the area under the Student-t probability density function is equal to `1-value1/2`. The function uses the studentT function to determine the return value iteratively.

5.4.8.1 beta1(value1, value2, value3)

Returns the area from zero to value3 under the beta density function.

The input arguments can be one of the following data types: double, float. Returned value is of type double.

Values 1 and 2 must be greater than 0.0. Value 3 must be greater than 0 and less than 1.

Function	Result
`beta1(0.1, 1.1, 0.2)`	0.8620112116492348
`beta1(316.13, 316.13, 0.2)`	1.40801423421089E-63

5.4.8.2 betacomplemented(value1, value2, value3)

Returns the area under the right hand tail (value 3 to infinity) of the beta density function.

The input arguments can be one of the following data types: double, float. Returned value is of type double.

Values 1 and 2 must be greater than 0.0. Value 3 must be greater than 0 and less than 1.

Function	Result
`betacomplemented(0.1, 1.1, 0.2)`	0.017407170120127144

5.4.8.3 binomial2(value1, value2, value3)

Returns the sum of the terms 0 through value1 of the Binomial probability density.

The input arguments can be one of the following data types:

- Value 1 - The end term. Data Type: Integer.
- Value 2 - The number of trials. Data type: Integer.
- Value 3 - The probability of success. Value must be between 0.0 and 1.0. Data type: Double, float.

Function	Result
`beta1(2, 2, 0.5)`	1.0

5.4.8.4 binomialcomplemented(value1, value2, value3)

Returns the sum of the terms value1 + 1 through value 2, of the Binomial probability density.

The input arguments can be one of the following data types:

- Value 1 - The end term. Data Type: Integer.
- Value 2 - The number of trials. Data type: Integer.
- Value 3 - The probability of success. Value must be between 0.0 and 1.0. Data type: Double, float.

The returned value is of the type double.

Function	Result
`binomialcomplemented(2, 3, 0.5)`	0.125

5.4.8.5 chiSquare(value1, value2)

Returns the area under the left hand tail (from 0 to value2) of the Chi square probability density function with value1 degrees of freedom. The arguments must both be positive.

Value 1: The degrees of freedom. Value can be one of the following types: double, float.

Value 2: The integration end point. Value can be one of the following types: double, float.

Returned value type will be double.

Function	Result
chiSquare(3.0, 5.0)	0.8282028557032665

5.4.8.6 chiSquareComplemented(value1, value2)

Returns the area under the right hand tail (from value2 to infinity) of the Chi square probability density function with value1 degrees of freedom. The arguments must both be positive.

Value 1 is the degrees of freedom. Value can be one of the following types: double, float.

Value 2 is the Chi-square variable. Value can be one of the following types: double, float.

Returned value type will be double.

Function	Result
`chiSquareComplemented(value1, value2)`	0.1717971442967335

5.4.8.7 errorFunction(value1)

Returns the error function of the normal distribution.

Value 1 can be one of the following types: double, float.

Returned value type will be double.

Function	Result
`errorFunction(5.0)`	0.9999999999984626

5.4.8.8 errorFunctionComplemented(value1)

Returns the complementary Error function of the normal distribution.

Value 1 can be one of the following types: double, float.

Returned value type will be double.

Function	Result
errorFunctionComplemented(5.0)	1.5374597944280347E-12

5.4.8.9 gamma(value1, value2, value3)

Returns the gamma function of the input arguments.

Value1: The paramater a (alpha) of the gamma distribution. Value can be one of the following types: double, float. Required.

Value 2: The paramater b (beta, lambda) of the gamma distribution. Value can be one of the following types: double, float. Optional.

Value 3: The integration end point. Value can be one of the following types: double, float. Optional.

Returned value type will be double.

Function	Result
`gamma(1.0,2.0,5.0)`	0.04042768199451279

5.4.8.10 gammacomplemented(value1, value2, value3)

Returns the integral from value3 to infinity of the gamma probability density function.

Value1: The paramater a (alpha) of the gamma distribution. Value can be one of the following types: double, float.

Value 2: The paramater b (beta, lambda) of the gamma distribution. Value can be one of the following types: double, float. R

value3The integration end point. Value can be one of the following types: double, float.

Returned value type will be double.

Function	Result
`gammacomplemented(1.0, 2.0, 5.0)`	0.04042768199451279

5.4.8.11 incompleteBeta(value1, value2, value3)

Returns the Incomplete Beta Function evaluated from zero to value3. Where values must be in range (Value1 && Value2 > 0.0) and (Value3 > 0 && Value3 < 1).

Value1: The alpha parameter of the beta distribution. Value can be one of the following types: double, float. Required.

Value2: The beta parameter of the beta distribution. Value can be one of the following types: double, float. Required.

Value3: The integration end point. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`incompleteBeta(1.0,2.0,0.5)`	0.75

5.4.8.12 incompleteGamma(value1, value2)

Returns the Incomplete Gamma function.

Value1: The parameter of the gamma distribution. Value can be one of the following types: double, float. Required.

Value2: The integration end point. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`incompleteGamma(1.0,2.0)`	0.8646647167633873

5.4.8.13 incompleteGammaComplement(value1, value2)

Returns the Complemented Incomplete Gamma function.

Value1: The parameter of the gamma distribution. Value can be one of the following types: double, float. Required.

Value2: The integration start point. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
incompleteGammaComplement(1.0, 2.0)	0.1353352832366127

5.4.8.14 logGamma(value1)

Returns the natural logarithm of the gamma function

Value can be one of the following types: double, float.

Returned value will be of type double.

Function	Result
logGamma(7795.6)	62059.66356433673

5.4.8.15 negativeBinomial(value1, value2, value3)

Returns the sum of the terms 0 through value1 of the Negative Binomial Distribution. All arguments must be positive.

Value1: The end term. Value can be one of the following types: integer. Required.

Value2: The number of trials. Value can be one of the following types: integer. Required.

Value3: The probability of success [must be in (0.0,1.0)]. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`negativeBinomial(1,2,0.5)`	0.5

5.4.8.16 negativeBinomialComplemented(value1, value2, value3)

Returns the sum of the terms value1+1 to infinity of the Negative Binomial distribution. All arguments must be positive.

Value1: The end term. Value can be one of the following types: integer. Required.

Value2: The number of trials. Value can be one of the following types: integer. Required.

Value3: The probability of success [must be in (0.0,1.0)]. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`negativeBinomialComplemented(1.0, 2.0, 0.5)`	0.5

5.4.8.17 normal(value1, value2, value3)

Returns the area under the Normal (Gaussian) probability density function, integrated from minus infinity to value1 (assumes mean is zero, variance is one).

Value1: The mean of the normal distribution. Value can be one of the following types: double, float. Required.

Value2: The variance of the normal distribution. Value can be one of the following types: double, float. Optional.

Value3: The integration limit. Value can be one of the following types: double, float. Optional.

Returned value type will be double.

Function	Result
`normal(5.0,3.0,0.5)`	0.004687384229717484

5.4.8.18 normalInverse(value1)

Returns the value for which the area under the Normal (Gaussian) probability density function is equal to the argument value1 (assumes mean is zero, variance is one).

Input value should be between 0 and 1. The value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`normalInverse(0.5)`	0.0

5.4.8.19 poisson(value1, value2)

Returns the sum of the first value1 terms of the Poisson distribution. Both the arguments must be positive.

Value1: The number of terms. Value can be one of the following types: integer. Required.

Value2: The mean of the poisson distribution. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`poisson(61,123.75)`	3.0509714140892473E-10

5.4.8.20 poissonComplemented(value1, value2)

Returns the sum of the terms value1+1 to Infinity of the Poisson distribution.

Value1: The start term. Value can be one of the following types: integer. Required.

Value2: The mean of the poisson distribution. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`poissonComplemented(5,3.0)`	0.08391794203130347

5.4.8.21 studentT(value1, value2)

Returns the integral from minus infinity to 'value2' of the Student-t distribution with value1 > 0 degrees of freedom.

Value1: The degrees of freedom. Value can be one of the following types: double, float. Required.

Value2: The integration end point. Value can be one of the following types: double, float. Required.

Returned value type will be double.

Function	Result
`studentT(2.0,5.0)`	0.9811252243246882

5.4.8.22 studentTInverse(value1, value2)

Returns the value, for which the area under the Student-t probability density function is equal to 1-value1/2. The function uses the studentT function to determine the return value iteratively.

Value1: The probability. Value can be one of the following types: double, float. Required.

Value2: The size of data set. Value can be one of the following types: integer. Required.

Returned value type will be double.

Function	Result
`studentTInverse(0.5, 10)`	0.6998121397488263

5.4.9 Using String Functions

The following String functions are supported in this release:

Function Name	Description
`coalesce(value1,...)`	Returns the first non-null expression in the list. If all expressions evaluate to null, then the COALESCE function will return null
`concat(value1,...)`	Returns concatenation of values converted to strings
`indexof(string,match)`	Returns first index of `\'match\'` in `\'string\'`or 1 if not found
`initcap(value1)`	Returns a specified text expression, with the first letter of each word in uppercase and all other letters in lowercase
`length(value1)`	Returns the length of the specified string
`like(value1,value2)`	Returns a matching pattern
`lower(value1)`	Converts the given string to lower case
`lpad(value1,value2,value3)`	Pads the left side of a string with a specific set of characters (when `string1` is not null)
`ltrim(value1,value2)`	Removes all specified characters from the left hand side of a string
`replace(string,match,replacement)`	Replaces all `\'match\'` with `\'replacement\'` in `\'string\'`
`rpad(value1,value2,value3)`	Pads the right side of a string with a specific set of characters (when `string1` is not null)
`rtrim(value1,value2)`	Removes all specified characters from the right hand side of a string
`substr(string,from)`	Returns substring of a 'string' when indices are between 'from' (inclusive) and up to the end of the string

Function Name	Description
`substring(string,from,to)`	Returns substring of a \'string\' when indices are between \'from\' (inclusive) and \'to\' (exclusive)
`translate(value1,value2,value3)`	Replaces a sequence of characters in a string with another set of characters. However, it replaces a single character at a time.
`upper(value1)`	Converts given string to uppercase

5.4.9.1 coalesce(value1,...)

`coalesce` returns the first non-null expression in the list of expressions. You must specify at least two expressions. If all expressions evaluate to null then the `coalesce` function will return null.

For example:

In `coalesce(expr1,expr2)`:

- If `expr1` is not null then the function returns `expr1`.
- If `expr1` is null then the function returns `expr2`.
- If `expr1` and `expr2` are null then the function returns null.

In `coalesce(expr1,expr2,.....,exprn)`

- If `expr1` is not null then the function returns `expr1`.
- If `expr1` is null then the function returns `expr2`.
- If `expr1` and `expr2` are null then the function returns the next non-null expression.

5.4.9.2 Concat(value1,...)

Concat(value1,...) - Concatenation of values converted to strings

Value1: A part of string to concatenate with others. Value can be one of the following types: big integer, number, double, text, integer, float, timestamp. Required.

Vararg1: A part of string to concatenate with others. Value can be one of the following types: big integer, number, double, text, integer, float, timestamp. Optional.

Returned value will be of type text.

Function	Result
`Concat(client_name, card_number)`	Declan BENNETT0142354466948788

5.4.9.3 indexof(value1, value2)

Returns first index of 'match' in 'string' or -1 if not found

Value1: First argument. Value can be one of the following types: text. Required.

Value2: Second argument. Value can be one of the following types: text. Required.

Returned value type will be integer.

Function	Result
`indexof(client_name,"c")`, where client name is Alphonse Gabriel Capone	17
`indexof(client_name,"c")`, where client name is Braden Gray	-1

5.4.9.4 initcap(value1)

Function returns a specified text expression, with the first letter of each word in uppercase and all other letters in lowercase : INITCAP

Value1: A text expression. Value can be one of the following types: text.

Returned value will be of type text.

Function	Result
`initcap(client_name)`, where client name is Owen TAYLOR	Owen Taylor

5.4.9.5 length(value1)

Returns the length in characters of the string passed as an input argument. The input argument is of the data type text. The returned value is an integer representing the total length of the string.

If `value1` is null, then `length(value1)` returns null.

If `value1` is an empty string, then `length(value1)` returns null.

Examples

Function	Result
`length(“one”)`	3
`length()`	ERROR: Function has invalid parameters.
`length(“john”)`	4
`length(“ “)`	NULL
`length(null)`	NULL
`length(“firstname.lastname@example.com”)`	30

5.4.9.6 like(string, pattern)

Function returns 'true' or 'false' based on the string matching the supplied pattern. .

Value 1 can be one of the following types: text.

Value 2 can be one of the following types: text.

Returned value type will be boolean.

Function	Result
`like(client_name, "ADAMS")`, where client name is Cameron Adams	True
`like(client_name, "ADAMS")`, where client name is Levi Gray	False

5.4.9.7 lower(value1)

Converts a string to all lower-case characters. The input argument is of the data type text. The returned value is the lowercase of the specified string.

Examples

Function	Result
`lower("PRODUCT")`	product
`lower("ABCdef")`	abcdef
`lower("abc")`	abc

5.4.9.8 lpad(value1, value2, value3)

LPad(text-exp , length [,pad-exp]) - Function pads the left-side of a string with a specific set of characters (when string1 is not null). : LPAD

Value1: text-exp - A text expression that you want to pad. Value can be one of the following types: text.

Value 2: length - The total length of the return value as it is displayed on your screen. Value can be one of the following types: integer.

Value 3: pad-exp - A text expression that specifies the padding characters. The default value of pad-exp is a single blank. Value can be one of the following types: text.

Returned value type will be text.

Function	Result
`lpad("David",10,"e")`	eeeeeDavid

5.4.9.9 ltrim(value1, value2)

The ltrim() function removes all specified characters from the left-hand side of a string : LTRIM.

Value 1 can be one of the following types: text.

Value 2 can be one of the following types: text.

Returned value type will be text.

Function	Result
`ltrim(client_name, "A")`, where client_name is Alphonse Gabriel CAPONE	lphonse Gabriel CAPONE

5.4.9.10 replace(string, match, replacement)

Replaces all `match` characters in a string with `replacement` characters. The first input argument is the `string` and is of the data type text. The second argument is the `match` and is of the data type text. The third argument is `replacement` and is of data type text. The returned value is a text in which the third string argument (`replacement`) replaces the second string argument (`match`).

If `match` is not found in the string, then the original string will be returned.

Examples

Function	Result
`replace("aabbccdd","cc","ff")`	aabbffdd
`replace("aabbcccdd","cc","ff")`	aabbffcdd
`replace("aabbddee","cc","ff")`	aabbddee

5.4.9.11 rpad(value1, value2, value3)

RPad(text-exp , length [,pad-exp]) - Function pads the right-side of a string with a specific set of characters (when string1 is not null). : RPAD

Value 1: text-exp - A text expression that you want to pad. Value can be one of the following types: text.

Value 2: length - The total length of the return value as it is displayed on your screen. Value can be one of the following types: integer.

Value 3: pad-exp - A text expression that specifies the padding characters. The default value of pad-exp is a single blank. Value can be one of the following types: text.

Returned value will be of type text.

Function	Result
`rpad("Levi Cruz", 25, "a")`	Levi Cruzaaaaaaaaaaaaaaaa

5.4.9.12 rtrim(value1, value2)

The rtrim() function removes all specified characters from the right-hand side of a string : RTRIM.

Value 1 can be one of the following types: text.

Value 2 can be one of the following types: text.

Returned value type will be text.

Function	Result
`rtrim(client_name, "S")`, where client_name is Cooper DAVIS	Cooper DAVI

5.4.9.13 substr()

Substr(string, from) - Substring of a 'string' when indices are between 'from' (inclusive) and up to the end of the string.

Value 1 can be one of the following types: text.

Value 2 can be one of the following types: integer.

Returned value type will be text.

Function	Result
`substr(client_name, 4)`,where client_name is Logan THOMPSON	n THOMPSON

5.4.9.14 substring(string, from, to)

Returns a substring of a string when indices are between `from` (inclusive) and `to` (exclusive). The first input argument is the `string` and is of the data type text. The second argument is the start index and is an integer. The third argument is the finish index and is an integer. The returned value is a substring and is of type text.

Examples

Function	Result
`substring("abcdefgh",3,7)`	cdef
`substring("abcdefgh",1,6)`	abcde

5.4.9.15 translate(expression, from_string, to_string)

Function replaces a sequence of characters in a string with another set of characters. However, it replaces a single character at a time.

Vallue1: exp - A text expression in which you want to replace characters. Value can be one of the following types: text.

Value2: from_string - A text expression that is the characters you want to replace. Value can be one of the following types: text.

Value3: to_string - A text expression that is the characters that you want to use for replacement in the order of from_string. When you include fewer characters in this argument than are in from_string, the function removes the extra characters in from_string from the return value. Value can be one of the following types: text.

Returned value type will be text.

Function	Result
`translate(client_name, "JONES", "Mark"), where the value for client_name is Cooper JONES.`	Cooper Mark

5.4.9.16 upper(value1)

Converts a string to all upper-case characters. The input argument is of the data type text. The returned value is the uppercase of the specified string.

Examples

Function	Result
`upper("name")`	NAME
`upper("abcdEFGH")`	ABCDEFGH
`upper("ABCD")`	ABCD

5.5 Adding Custom Functions and Custom Stages

Custom functions are user-defined functions that are custom implementations to an application's built-in functions.

5.5.1 Creating a Custom Jar

A custom jar is a user-supplied Jar archive containing Java classes for custom stage types or custom functions that will be used within a pipeline.

To create a Custom Jar:

1. On the **Catalog** page, click **Create New Item**, and select **Custom Jar** from the drop-down list.
1. On the **Type Properties** screen, enter the following details:
   - **Name**
   - **Description**
   - **Tags**
   - **Custom Jar Type**: Select Custom Jar, from the drop-down list.
2. Click **Next**.
3. On the **Custom Jar Details** page, click **Upload file**, select the jar file that you want to import into the application.
4. Click **Save**.

Your custom Java/Scala class must implement the BatchEventProcessor interface as defined in the Javadoc.

5.5.2 Adding Custom Functions

The custom functions get installed, when you add a custom jar file.

The custom functions will be available in the Expression Builder after they get installed. The custom functions will be listed under the *Custom* category.

5.5.3 Implementing Custom Functions

For a custom function, apply the `@OsaFunction` annotation to a method in any class, including a class implementing a custom stage type. For more information, see the Javadoc and the Sample.

> **Note:**
>
> Functions with same name within same package/class/method in same/different jar are not supported.

5.5.3.1 Sample: Encrypt a Column

This sample class defines a custom function that takes one textual field and produces an MD5 hash for it.

```
package com.oracle.osacs;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import com.oracle.cep.api.annotations.OsaFunction;


public class CustomMD5Function {

        @OsaFunction(name = "md5", description = "Create an md5 hex from a
string")
        public static String md5(String message) {
               String result = null;

               try {
                       MessageDigest md = MessageDigest.getInstance("MD5");
                   md.update(message.getBytes());
                   byte[] digest = md.digest();
                   result =
javax.xml.bind.DatatypeConverter.printHexBinary(digest);
               } catch (NoSuchAlgorithmException e) {
                       e.printStackTrace();
               }

             return result;
        }

}
```

5.5.4 Adding a Custom Stage

To add a custom stage:

1. Open the required pipeline in Pipeline Editor.

2. Right-click the stage after which you want to add a custom stage. Click **Add a Stage**, and **Custom**, and then select **Custom Stage from Custom Jars**.
3. Enter a name and suitable description for the custom stage and click **Save**.
4. In the stage editor, enter the following details:
    a. **Custom Stage Type**: Select the custom stage that was previously installed though a custom jar
    b. **Input Mapping**: Select the corresponding column from the previous stage for every input parameter

You can add multiple custom stages based on your use case.

5.5.4.1 Sample: Encrypt a Column

This sample class defines a custom stage that takes one textual field and produces an MD5 hash for it.

```
package com.oracle.osacs;

import com.oracle.cep.api.event.*;
import com.oracle.cep.api.annotations.OsaStage;
import com.oracle.cep.api.stage.EventProcessor;
import com.oracle.cep.api.stage.ProcessorContext;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;

@SuppressWarnings("serial")
@OsaStage(name = "md5", description = "Create an md5 hex from a string",
inputSpec = "input, message:string", outputSpec = "output, message:string,
md5:string")
public class CustomMD5Stage implements EventProcessor {

    EventFactory eventFactory;
    EventSpec outputSpec;

    @Override
    public void init(ProcessorContext ctx, Map<String, String> config) {
        eventFactory = ctx.getEventFactory();
        OsaStage meta = CustomMD5Stage.class.getAnnotation(OsaStage.class);
        String spec = meta.outputSpec();
        outputSpec = TupleEventSpec.fromAnnotation(spec);
    }

    @Override
    public void close() {
    }

    @Override
    public Event processEvent(Event event) {
        Attr attr = event.getAttr("message");
        Map<String, Object> values = new HashMap<String, Object>();
        if (!attr.isNull()) {
            String val = (String) attr.getObjectValue();
```

```
            String md5 = null;
            try {
                MessageDigest md = MessageDigest.getInstance("MD5");
                md.update(val.getBytes());
                byte[] digest = md.digest();
                md5 = javax.xml.bind.DatatypeConverter.printHexBinary(digest);
            } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            values.put("message", val);
            values.put("md5", md5);
        } else {
            values.put("message", "empty");
            values.put("md5", "empty");
        }
        Event outputEvent = eventFactory.createEvent(outputSpec, values,
event.getTime());
        return outputEvent;
    }
}
```

5.5.4.2 Sample: Invoke a REST Service

```
package com.oracle.osacs;

import com.oracle.cep.api.event.*;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.oracle.cep.api.annotations.OsaStage;
import com.oracle.cep.api.stage.EventProcessor;
import com.oracle.cep.api.stage.ProcessorContext;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.Random;

import org.apache.http.HttpHost;
import org.apache.http.HttpResponse;
import org.apache.http.StatusLine;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpRequestBase;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.util.EntityUtils;

class BookResult {
    String isbn;
    String title;
    String publishedDate;
    String publisher;
}
```

```
@SuppressWarnings("serial")
@OsaStage(name = "RestBooks", description = "Provide info for a given book",
inputSpec = "input, isbn:string", outputSpec = "output, isbn:string,
title:string, publishedDate:string, publisher:string")
public class CustomStageRest implements EventProcessor {

    EventFactory eventFactory;
    EventSpec outputSpec;

    static Properties props = new Properties();

    static {
        try {
            props.load(CustomStageRest.class.getResourceAsStream("/
CustomStageRest.properties"));
        } catch (IOException ioex) {
            ioex.printStackTrace();
        }
    }

    @Override
    public void init(ProcessorContext ctx, Map<String, String> config) {
        eventFactory = ctx.getEventFactory();
        OsaStage meta = CustomStageRest.class.getAnnotation(OsaStage.class);
        String spec = meta.outputSpec();
        outputSpec = TupleEventSpec.fromAnnotation(spec);
    }

    @Override
    public void close() {
    }

    @Override
    public Event processEvent(Event event) {
        Attr isbnAttr = event.getAttr("isbn");

        Map<String, Object> values = new HashMap<String, Object>();
        if (!isbnAttr.isNull()) {
            String isbn = (String) isbnAttr.getObjectValue();

            BookResult result = getBook(isbn);

            values.put("isbn", isbn);
            values.put("title", result.title);
            values.put("publishedDate", result.publishedDate);
            values.put("publisher", result.publisher);

        } else {
            values.put("isbn", "");
            values.put("title", "");
            values.put("publishedDate", "");
            values.put("publisher", "");
        }
        Event outputEvent = eventFactory.createEvent(outputSpec, values,
event.getTime());
```

```
        return outputEvent;
    }


    /**
     * Calls the Google Books REST API to get book information based on the
ISBN ID
     * @param isbn
     * @return BookResult book information
     */
    public BookResult getBook(String isbn) {
        HttpRequestBase request;
        BookResult result = null;

        String uri = "https://www.googleapis.com/books/v1/volumes?q=isbn:" +
isbn;

        request = new HttpGet(uri);

        CloseableHttpClient client = HttpClientBuilder.create().build();

        String proxyHost = props.getProperty("proxyHost");
        String proxyPort = props.getProperty("proxyPort");
        if (proxyHost != null && proxyPort != null) {
            int proxyPortInt = Integer.parseInt(proxyPort);
            HttpHost proxy = new HttpHost(proxyHost, proxyPortInt);
            RequestConfig config =
RequestConfig.custom().setProxy(proxy).build();
            request.setConfig(config);
        }

        try {
            HttpResponse response = client.execute(request);
            String resultJson = EntityUtils.toString(response.getEntity());
            StatusLine sl = response.getStatusLine();
            int code = sl.getStatusCode();
            if (code < 200 || code >= 300) {
                System.err.println("" + code + " : " + sl.getReasonPhrase());
            }

            ObjectMapper mapper = new ObjectMapper();
            JsonNode root = mapper.readValue(resultJson, JsonNode.class);
            JsonNode bookArray = root.path("items");

            if (bookArray.size() > 0) {
                result = new BookResult();
                JsonNode book = bookArray.path(0).path("volumeInfo"); // We
only consider the first book for this ISBN
                result.isbn = isbn;
                result.title = book.path("title").asText();
                result.publishedDate = book.path("publishedDate").asText();
                result.publisher = book.path("publisher").asText();
                return result;
            } else {
                return null; // No book found
            }
```

```
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
        }
}
```

> **Note:**
>
> Following third-party jars are required for compilation of REST sample,
>
> - httpclient-4.5.6.jar
> - httpcore-4.4.10.jar
> - jackson-databind-2.9.10.jar
>
> The above jars are required only at compile time and need not be packaged along with custom jar. These libraries and their dependencies are already packaged with OSA distribution.

5.5.4.3 Sample: Invoke a SOAP Service

```
    import java.net.MalformedURLException;
    import java.net.URL;
    import java.util.ArrayList;
    import java.util.HashMap;
    import java.util.Iterator;
    import java.util.List;
    import java.util.Map;
    import javax.xml.namespace.QName;
    import javax.xml.ws.Service;
    import com.oracle.cep.api.annotations.OsaStage;
    import com.oracle.cep.api.event.Attr;
    import com.oracle.cep.api.event.Event;
    import com.oracle.cep.api.event.EventFactory;
    import com.oracle.cep.api.event.EventSpec;
    import com.oracle.cep.api.event.TupleEventSpec;
    import com.oracle.cep.api.stage.BatchEventProcessor;
    import com.oracle.cep.api.stage.ProcessorContext;

    @SuppressWarnings("serial")
    @OsaStage(name = "CustomSoapBatchCall", description = "Call a Hello World
Soap WS", inputSpec = "input, message:string", outputSpec = "output,
message:string, result:string")
    public class CustomSoapBatchCall implements BatchEventProcessor {

        EventFactory eventFactory;
        EventSpec outputSpec;
        URL url;
        QName qname;
        Service service;
        HelloWorldServer server;
```

```
        @Override
        public void init(ProcessorContext ctx, Map<String, String> config) {
            eventFactory = ctx.getEventFactory();
            OsaStage meta =
CustomSoapBatchCall.class.getAnnotation(OsaStage.class);
            String spec = meta.outputSpec();
            outputSpec = TupleEventSpec.fromAnnotation(spec);
        try {
            url = new URL("http://hostname:9879/hw?wsdl");
        } catch (MalformedURLException e) {
    e.printStackTrace();
    }
    qname = new QName("http://ws.osa.oracle.com/",
    "HelloWorldServerImplService");
    service = Service.create(url, qname);
    server = (HelloWorldServer) service.getPort(HelloWorldServer.class);
    }

    @Override
    public void close() {
    }

    @Override
    public Iterator<Event> processEvents(Iterator<Event> iterator) {
        List<String> reqs = new ArrayList<String>();
        while(iterator.hasNext()){


reqs.add((String)iterator.next().getAttr("message").getObjectValue());

        }

        String[] ress = server.sayHelloBatch(reqs.toArray(new
String[reqs.size()]));

        return new Iterator<Event>() {
            int i = 0;
            @Override
            public boolean hasNext() {
                return i++ < ress.length;

            }

            @Override
            public Event next() {
                Map<String, Object> values = new HashMap<String, Object>();
                values.put("message",reqs.get(i-1));
                values.put("result",ress[i-1]);
                return
eventFactory.createEvent(outputSpec,values,System.currentTimeMillis());

            }

        };

    }
```

```
    @Override
    public Event processEvent(Event event) {
        Attr attr = event.getAttr("message");
        Map<String, Object> values = new HashMap<String, Object>();
        if (!attr.isNull()) {
        String val = (String) attr.getObjectValue();
        String result = callSoap(val);
        values.put("message", val);
        values.put("result", result);
    } else {
    values.put("message", "empty");
    values.put("result", "empty");
    }
    Event outputEvent = eventFactory.createEvent(outputSpec,
values,event.getTime());
    return outputEvent;
    }

    public String callSoap(String myName) {
    return server.sayHello(myName);
    }

    }
```

5.5.5 Limitations

The limitations and restrictions of the custom stages and custom functions are listed in this section.

Custom stage type and custom functions must:

- only be used for stateless transformations. Access to state from previous calls to stage type or function methods cannot be guaranteed and might change based on optimizations.
- not use any blocking invocations.
- not start a new thread.
- not use any thread synchronization primitives, including the wait() method, which could potentially introduce deadlocks.
- have/be in a fully-qualified class name.

When you use the custom stages or custom functions, be careful about the heap space usage.

> **Note:**
>
> The resulting jar must include all the required dependencies and third-party classes and the size of the jar file must be less than 160 MB.

5.5.6 Mapping of Data Types

The following table lists the data types that can be used by custom stage types and custom functions.

Oracle Stream Analytics Data Type	Java Data Type	Comment
BOOLEAN	boolean	
INT	int	
BIGINT	long	
FLOAT	float	
DOUBLE	double	
STRING	String	
BIGDECIMAL	BigDecimal	
TIMESTAMP	long (in nanoseconds)	Can only be used in Custom Stage Types
INTERVAL	long	Can only be used in Custom Stage Types

5.6 Writing CQL Queries

You can add a Custom CQL stage for pipelines, to write a CQL query to perform any *Select* operation on the data of the previous stage. Currently, creating certain business rules are not supported by the existing pipeline stages or patterns stages. You can use the CQL query as part of the Custom CQL stage, for such requirements.

To learn more about the CQL, see CQL Reference Guide.

5.6.1 Sample Queries

This section lists sample queries, apart from the standard *Select and Where clause* queries.

The sample queries use the following sample data sets. Copy the data sets, and save them as csv files.

Sample: Data_A_Followed_B

Trans_id	order_id	order_status	order_revenue
1	1	BOOKED	2345.98
2	2	BOOKED	4345.98
3	3	BOOKED	3468.87
4	1	PAID	2345.98
5	2	PAID	4345.98
6	1	SHIPPED	4345
7	3	PAID	3468.87
8	3	SHIPPED	3468.87
9	4	BOOKED	3456
10	5	BOOKED	6546

11	6	BOOKED	76547
12	2	SHIPPED	4345.98

Sample: Change_Detector

Msg_ID	Stock_ID	Stock_Price
1	1	87
2	1	87
3	1	87
4	1	87
5	2	41
6	3	65
7	3	65
8	3	65
9	3	65
10	3	65
11	2	41
12	2	41
13	2	41
14	2	41
15	2	41
16	2	41
17	1	91
18	1	91
19	1	91
20	1	105
21	1	105
22	1	105
23	1	105
24	2	112
25	2	112
26	2	112
27	2	112
28	3	176
29	3	176
30	3	176

5.6.1.1 A Followed By B

> **Note:**
>
> In the sample query below, update `q1` to the correct name of the previous stage: Replace `FROM q1` to `FROM <previous-stage-name>`.

Use the sample data file: Data_A_Followed_B.

```
SELECT

order_id AS order_id,
abInterval AS abInterval,
Trans_id AS Trans_id,
aState_Trans_id AS aState_Trans_id,
order_status AS order_status,
aState_order_status AS aState_order_status,
order_revenue AS order_revenue,
aState_order_revenue AS aState_order_revenue
FROM q1

MATCH_RECOGNIZE (
PARTITION BY
order_id
MEASURES

B.Trans_id AS Trans_id,
B.order_id AS order_id,
B.order_status AS order_status,
B.order_revenue AS order_revenue,
A.Trans_id AS aState_Trans_id,
A.order_status AS aState_order_status,
A.order_revenue AS aState_order_revenue,
(to_timestamp(B.ELEMENT_TIME) - to_timestamp(A.ELEMENT_TIME)) AS abInterval
PATTERN( A C*? B )
WITHIN 1 minutes
DEFINE
A as A.order_status like ".*BOOKED.*" ,
B as B.order_status like ".*SHIPPED.*"

) as M
```

Output

```
{"order_id":1,"abInterval":"+000000000
00:00:05.000000000","Trans_id":6,"aState_Trans_id":1,"order_status":"SHIPPED",
"aState_order_status":"BOOKED","order_revenue":4345.0,"aState_order_revenue":2
345.98}
{"order_id":3,"abInterval":"+000000000
00:00:05.000000000","Trans_id":8,"aState_Trans_id":3,"order_status":"SHIPPED",
"aState_order_status":"BOOKED","order_revenue":3468.87,"aState_order_revenue":
3468.87}
{"order_id":2,"abInterval":"+000000000
00:00:10.000000000","Trans_id":12,"aState_Trans_id":2,"order_status":"SHIPPED"
,"aState_order_status":"BOOKED","order_revenue":4345.98,"aState_order_revenue"
:4345.98}
```

5.6.1.2 A Not Followed by B

> **Note:**
>
> In the sample query below, update `q1` to the correct name of the previous stage: Replace `FROM q1` to `FROM <previous-stage-name>`.

Use the sample data: Data_A_Followed_B.

```
SELECT

Trans_id AS Trans_id,
order_id AS order_id,
order_status AS order_status,
order_revenue AS order_revenue
FROM q1

MATCH_RECOGNIZE (
PARTITION BY
order_id
MEASURES

A.Trans_id AS Trans_id,
A.order_id AS order_id,
A.order_status AS order_status,
A.order_revenue AS order_revenue
INCLUDE TIMER EVENTS
PATTERN( A B* )
DURATION 1 minutes
DEFINE
A as A.order_status like ".*BOOKED.*" ,
B as NOT (B.order_status like ".*SHIPPED.*" )

) as M
```

Output

```
{"Trans_id":9,"order_id":4,"order_status":"BOOKED","order_revenue":3456.0}
{"Trans_id":10,"order_id":5,"order_status":"BOOKED","order_revenue":6546.0}
{"Trans_id":11,"order_id":6,"order_status":"BOOKED","order_revenue":76547.0}
```

5.6.1.3 Detect Duplicates

> **Note:**
>
> In the sample query below, update `q1` to the correct name of the previous stage: Replace `FROM q1` to `FROM <previous-stage-name>`.

Use the sample data file: Data_A_Followed_B.

```
RSTREAM( SELECT

count(*) AS Number_of_Duplicates,
eventSource.order_id AS order_id,
current(eventSource.Trans_id) AS Trans_id,
current(eventSource.order_status) AS order_status,
current(eventSource.order_revenue) AS order_revenue
FROM q1 [now] as eventSource,
q1 [range 1 minutes] as dup

WHERE
eventSource.order_id = dup.order_id
GROUP BY
eventSource.order_id HAVING count(*) > 1)
```

Output

```
{"Number_of_Duplicates":2,"order_id":1,"Trans_id":4,"order_status":"PAID","ord
er_revenue":2345.98}
{"Number_of_Duplicates":2,"order_id":2,"Trans_id":5,"order_status":"PAID","ord
er_revenue":4345.98}
{"Number_of_Duplicates":3,"order_id":1,"Trans_id":6,"order_status":"SHIPPED","
order_revenue":4345.0}
{"Number_of_Duplicates":2,"order_id":3,"Trans_id":7,"order_status":"PAID","ord
er_revenue":3468.87}
{"Number_of_Duplicates":3,"order_id":3,"Trans_id":8,"order_status":"SHIPPED","
order_revenue":3468.87}
{"Number_of_Duplicates":3,"order_id":2,"Trans_id":12,"order_status":"SHIPPED",
"order_revenue":4345.98}
```

5.6.1.4 Change Event

> **Note:**
>
> In the sample query below, update `q1` to the correct name of the previous stage: Replace `FROM q1` to `FROM <previous-stage-name>`.

Use the sample data file: change_detector.

```
SELECT

Stock_ID AS Stock_ID,
Stock_Price AS Stock_Price,
orig_Stock_Price AS orig_Stock_Price,
Msg_ID AS Msg_ID,
orig_Msg_ID AS orig_Msg_ID
FROM q1

MATCH_RECOGNIZE (
```

```
PARTITION BY
Stock_ID
MEASURES

Z.Stock_ID AS Stock_ID,
last(X.Stock_Price) AS Stock_Price,
Z.Stock_Price AS orig_Stock_Price,
last(X.Msg_ID) AS Msg_ID,
Z.Msg_ID AS orig_Msg_ID
PATTERN( Z X+ )
WITHIN 1 minutes
DEFINE X as X.Stock_Price != Z.Stock_Price
) as M
```

Output

```
{"Stock_ID":1,"Stock_Price":105,"orig_Stock_Price":87,"Msg_ID":23,"orig_Msg_ID
":4}
{"Stock_ID":2,"Stock_Price":112,"orig_Stock_Price":41,"Msg_ID":27,"orig_Msg_ID
":16}
{"Stock_ID":3,"Stock_Price":176,"orig_Stock_Price":65,"Msg_ID":30,"orig_Msg_ID
":10}
```

5.6.1.5 Eliminate Duplicates

> **Note:**
>
> In the sample query below, update `q1` to the correct name of the previous stage: Replace `FROM q1` to `FROM <previous-stage-name>`.

Use the sample data file: change_detector.

```
ISTREAM( SELECT

Msg_ID AS Msg_ID,
Stock_ID AS Stock_ID,
Stock_Price AS Stock_Price
FROM q1 [range 1 minutes]
) DIFFERENCE USING (Stock_Price)
```

Output

```
{"Msg_ID":1,"Stock_ID":1,"Stock_Price":87}
{"Msg_ID":5,"Stock_ID":2,"Stock_Price":41}
{"Msg_ID":6,"Stock_ID":3,"Stock_Price":65}
{"Msg_ID":17,"Stock_ID":1,"Stock_Price":91}
{"Msg_ID":20,"Stock_ID":1,"Stock_Price":105}
{"Msg_ID":24,"Stock_ID":2,"Stock_Price":112}
{"Msg_ID":28,"Stock_ID":3,"Stock_Price":176}
```

6
Analyze

6.1 Using Geofences for Location-based Analytics

A geo fence is a virtual boundary in a real world geographical area. This virtual boundary can be used to find the object's position or location, with respect to the geo fence.

For example, the object position can be:

- Near to geo fence
- Exit geo fence
- Based on Stay Duration in geo fence
- Enters geo fence
- Present inside geo fence

6.1.1 Selecting a Tile Layer

Tile layer is the base map that provides immediate geographic context. Tiles are stored in the map tile server. These tile layers contains huge amount of data pertaining to:

- Roads, railways, waterways, etc.
- Restaurants, shops, stations, ATMs, and more
- Walking and cycling paths
- Buildings, campuses, etc.

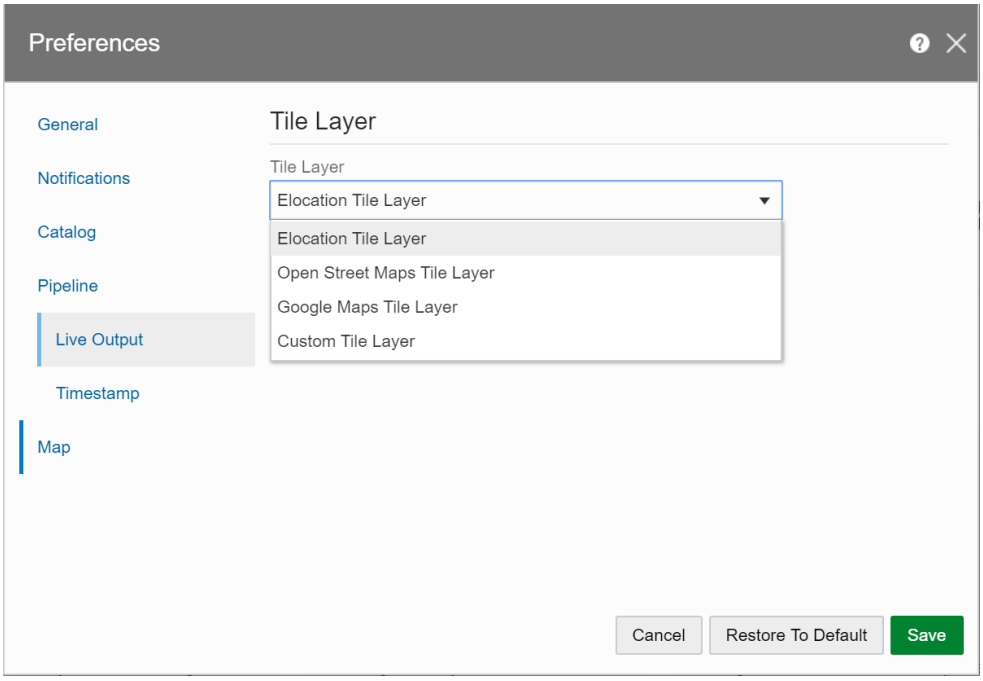
Oracle GoldenGate Stream Analytics supports four types of tile layers.

6.1.1.1 Elocation Tile Layer

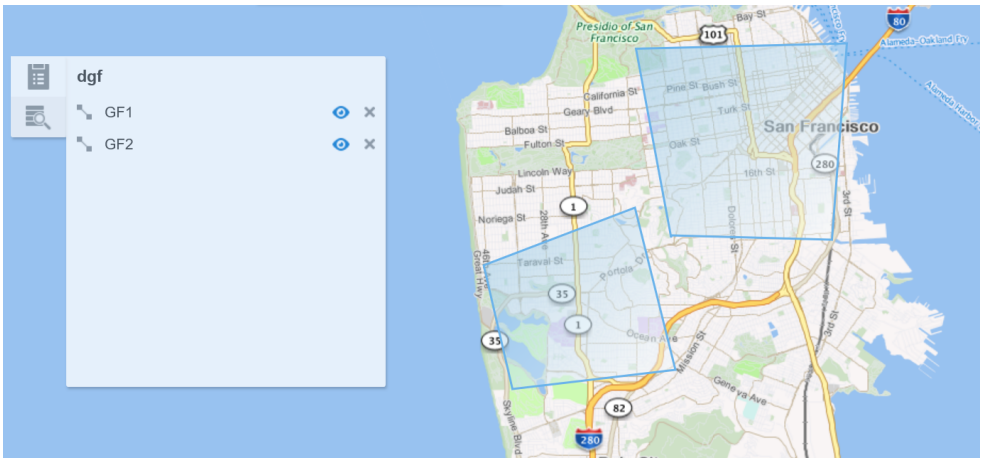
Elocation tile layer is an Oracle tile layer.

To apply the Elocation Tile Layer:

1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.
3. Click **Map**.
4. Under **Tile Layer**, choose **Elocation Tile Layer** option from the drop-down list.



5. Click **Save**. The map looks like this:

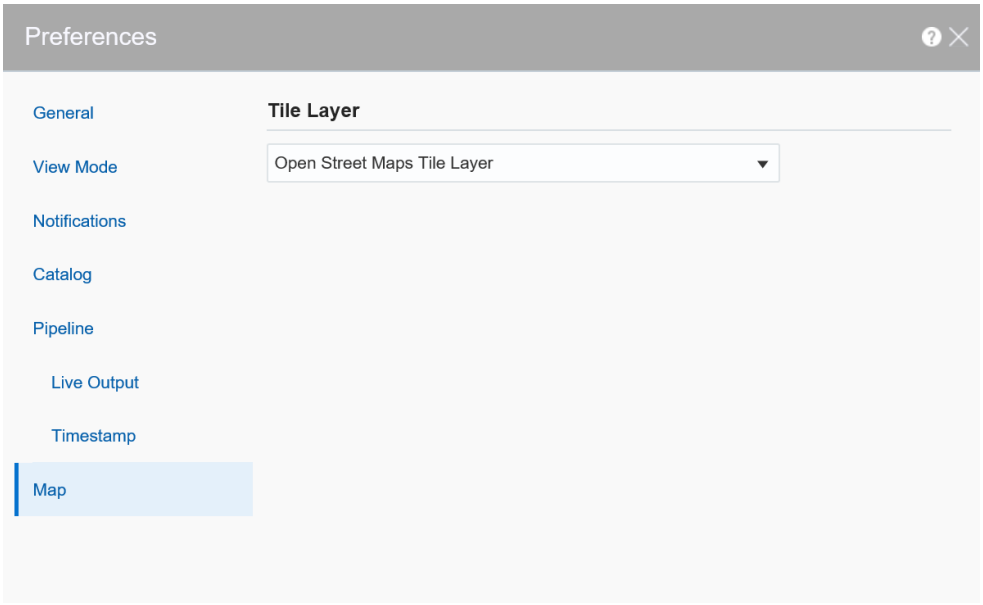


6.1.1.2 Open Street Maps Tile Layer

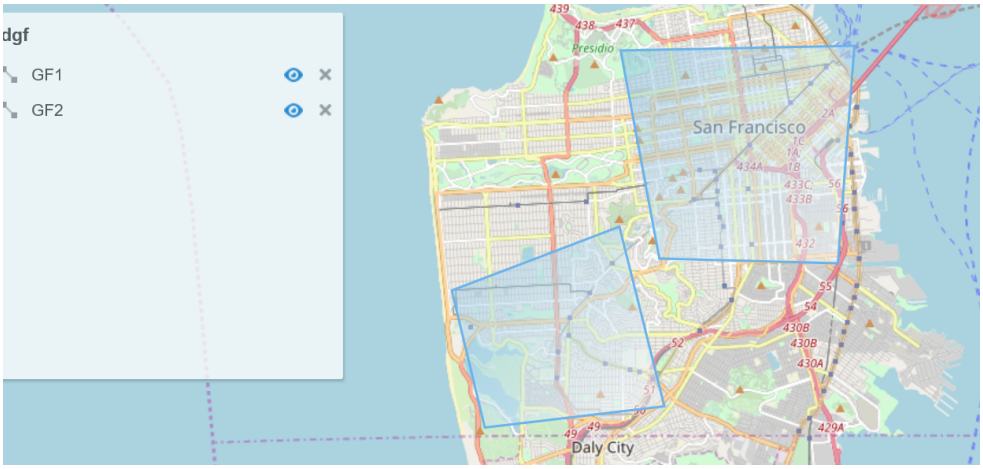
Open Street Maps tile layer is a free map.

To apply the Open Street Maps Tile Layer:

1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.
3. Click **Map**.
4. Under **Tile Layer**, choose **Open Street Maps Tile Layer** option from the drop-down list.



5. Click **Save**. The map looks like this:

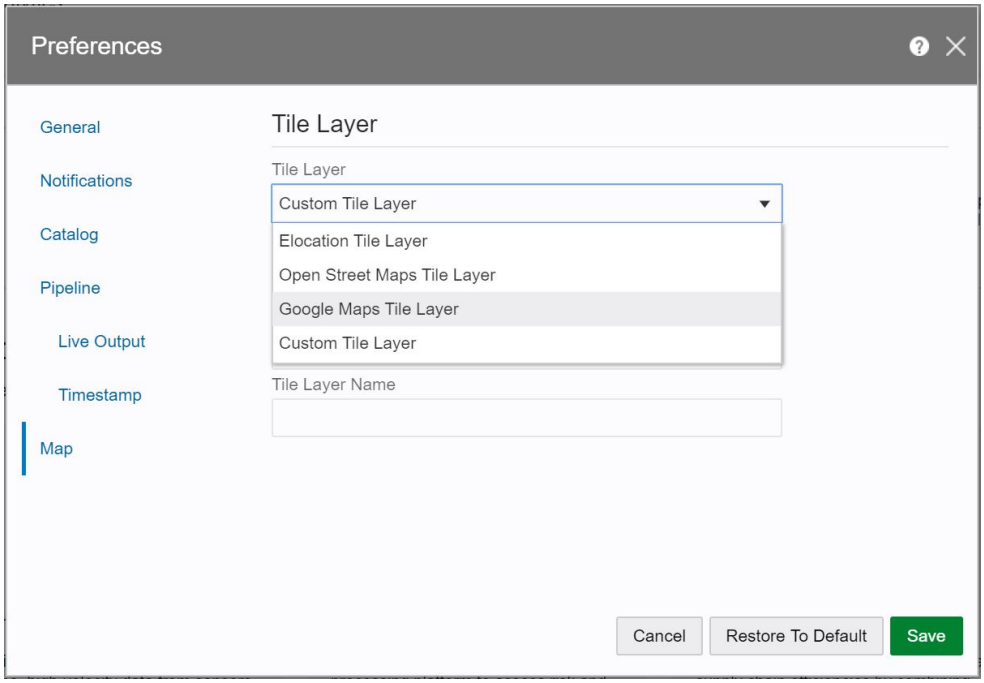


6.1.1.3 Google Maps Tile Layer

Oracle Stream Analytics has added the support for Google tile layer which displays Google maps in Spatial patterns and visualizations.

To apply the Google Tile Layer:

1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.
3. Click **Map**.
4. Under **Tile Layer**, choose **Google Maps Tile Layer** option from the drop-down list.



5. Provide details for the following fields:
   - **Lib URL** — the Google maps javascript lib url.

     > **Note:**
     >
     > If you do not provide the Lib Url, Oracle maps uses the default Lib Url:
     >
     > https://maps.google.com/maps/api/js?v=3&sensor=false

   - **Authentication Type**:
     - **API key** — API key for authentication which you can get from Google. The URL would look like this: https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap.
     - **Client ID** — your client ID which identifies you as a Maps API for Business customer
6. Click **Save**.

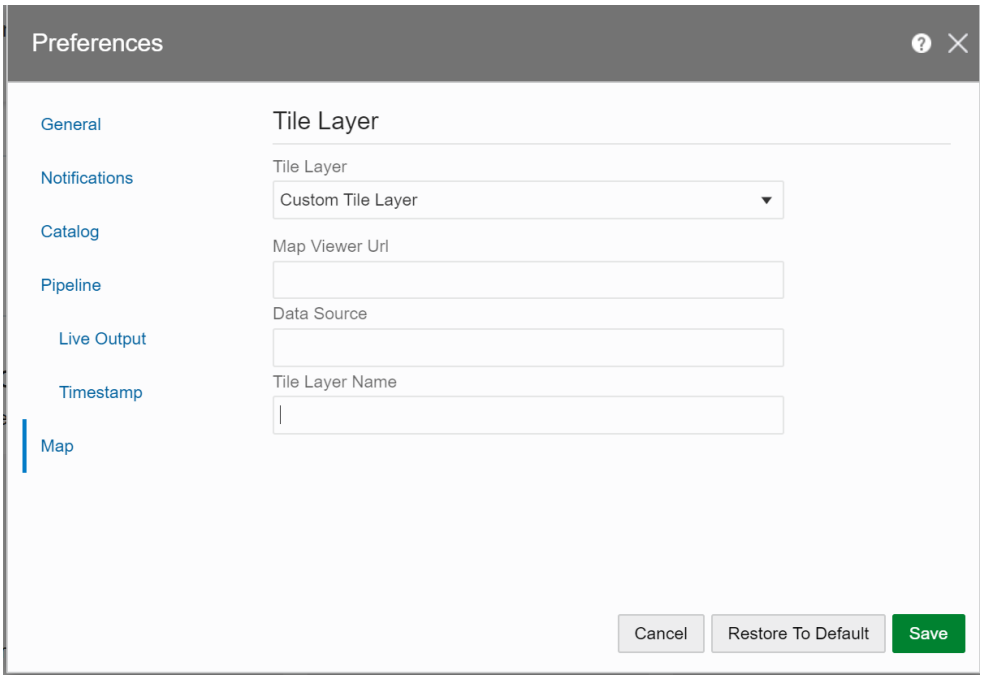
> **Note:**
>
> To use Google maps tile layer, the usage of the maps must meet the terms of service defined by Google (http://code.google.com/apis/maps/faq.html#tos).

6.1.1.4 Custom Tile Layer

Oracle Maps, by default, supports Elocation tile layer which has limited zoom levels. Oracle Stream Analytics now allows you to customize the zoom levels for a tile layer, in specific cases where you need a detailed, higher zoom view of confined spaces such as, restaurants, airports, etc.

To apply the Custom Tile Layer:

1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.
3. Click **Map**.
4. Under **Tile Layer**, choose **Custom Tile Layer** option from the drop-down list.



5. Provide details for the following fields:
   - **Map Viewer Url** — http://yourmapviewer => another address where you have an Oracle map viewer running
   - **Data Source** — a connection to a database where you define your style. It should be an OSA data source, or a custom one which allows to define our maps via mapbuilder.
   - **Tile Layer Name** — name of the defined tile layer
6. Click **Save**. The map looks like this:

> **Note:**
>
> Once you have modified the global parameters to customize the tile layer, the map is updated to use the custom tile layer. These customizations, will then be applied to all geofences.

6.1.2 Managing Geofences using the Map Editor

You can create, edit, and update manual geofence using the built-in map editor. Only polygon geo fences are allowed.

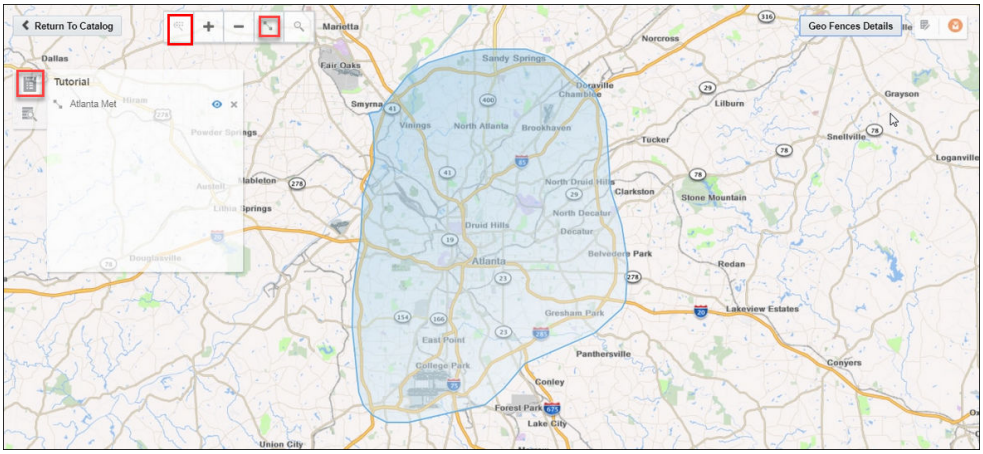
6.1.2.1 Creating a Geo Fence

To create a manual geo fence:

1. On the **Catalog** page, click **Create New Item** and select **Geo Fence** from the drop-down list.
2. On the **Type Properties** screen, enter the following details:
   - **Name**
   - **Description**
   - **Tags**
   - **Geo Fence Type**: Select **Manually Created Geo Fence** from the drop-down list.
3. Click **Save**.

   In the Geo Fence Editor:

   - You can create the geo fence.
   - Navigate through the map using the zoom icons
   - Zoom a specific area on the map, using the **Marquee Zoom** tool. Mark an area using the marquee zoom and that area in map is zoomed.
   - Mark the area around a region to create a geo fence, using the **Polygon Tool**.

   - Save the changes made to your geofence.

6.1.2.2 Deleting a Geofence

To delete a Manual Geofence:

1. Go to the **Catalog** page and hover the mouse over the geofence that you want to delete.
2. Click the delete icon that appears to your right side on the screen.
3. On the **Delete Confirmation** screen, click **Delete**.

6.1.3 Importing a Geofence from a Database

To import a geo fence from a database:

1. On the **Catalog** page, **Create New Item** and then select **Geo Fence** from the drop-down list.
2. On the **Type Properties** screen, enter the following details:
   - **Name**
   - **Description**
   - **Tags**
   - **Geo Fence Type**: Select **Geo Fence from Database** from the drop-down list.
3. Click **Next**.
4. On the **Geo Fence Details** screen, select a **Connection**.
5. Click **Next**.
6. On the **Shape** screen, select a database table from the drop-down list. Only the database tables that contain `SDO Geometry` type fields are available in the drop-down list.
   You can also define or change the key fields on this screen.
7. Click **Save**.

You can see all the geo fences contained in the selected database.

> **Note:**
>
> You cannot edit or update database-based geo fences.

6.1.4 Using Spatial Patterns in Pipeline Stages

Spatial patterns enables analysis of streams that contain geolocation data. Use them to determine how events relate to predefined geo fences in your maps.

6.1.4.1 Clearing Objects Outside a Geo Fence

Use the **Geo Filter** pattern to filter out objects that are not inside the Geofence.

For example, if users move from one geographical location to another, you can send promotional messages to the users when they are inside a specified geo fence.

To use this pattern, provide suitable values for the following parameters:

- **Geo Fence**: Select a geo fence to analyze.

- **Latitude**: Select a field containing the latitude value.
- **Longitude**: Select a field containing the longitude value.
- **Object Key**: Select a field that uniquely identifies the object. This field is the partitioning criteria and is also the unique object identifier.
- **Coordinate System**: Enter the default value 8307. This is the only value supported.

The outgoing shape displays **Status** and **PlaceName** as two extra columns in the output along with the incoming shape, where **Status** is `Inside` if the object is inside geo fence (else the event is not considered) and **PlaceName** is the name of geo fence with which status is being evaluated.

6.1.4.2 Tracking Objects using a Geo Fence

Use the **Geo Fence** pattern to track object relation with a virtual boundary called geo fence.

Relations can be `Enter`, `Exit`, `Stay`, or `Near` with respect to a geo fence. For example, you can trigger an alert when an object enters the geo fence. You can also analyze a stream containing geo-location data. It helps in determining how events are related to a polygon in a geo fence.

To use this pattern, provide suitable values for the following parameters:

- **Geo Fence**: Select a geo fence to analyze.
- **Latitude**: Select the field containing latitude value.
- **Longitude**: Select the field containing longitude value.
- **Object Key**: Select the object key field. This field is used as the partitioning criteria and also used to uniquely identify objects
- **Tracking Events**: Select the appropriate value.
- **Coordinate system**: The default and the supported value is 8307.
- **Distance Buffer**: This parameter is enabled only if you select **Near** option in **Tracking Events**. This field is a buffer for filtering results. Only those events or objects which are within the specified distance from the geo fence are displayed in events table with status as `Near`. This value must be less than 10000 kilometers.
- **Stay Duration**: This parameter is enabled only if you select **Stay** in **Tracking Events**. You can specify the stay duration. This duration is a filter for objects inside the geo fence. If an object stays for a duration more than the specified duration, only then the events are considered, else events are filtered out.

The outgoing shape displays **Status** and **PlaceName** as two extra columns in the output along with the incoming shape, where **Status** is one of `Enter`, `Exit`, `Stay`, or `Near` based on how the object behaves with geo fence. **PlaceName** is the name of geo fence with which status is being evaluated.

6.1.4.3 Getting Direction of a Moving Object

Use the **Direction** pattern to get the direction of a moving object.

For example, you can evaluate the direction of a moving truck.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select the field containing latitude value.
- **Longitude**: Select the field containing longitude value.

- **Object Key**: Select field that uniquely identifies object. E.g. Vehicle Id.
- **Coordinate System**: The default value is 8307 and this is the only value supported.

> **Note:**
>
> Make sure that you do not use any names for the fields that are already part of the incoming stream.

The outgoing shape displays **direction** as one of the columns, which is of type `String` along with the incoming shape.

6.1.4.4 Obtaining Geographic Coordinates

Use the **Geo Code** pattern to get geographic coordinates (like latitude and longitude) for an address or a zip code.

Ensure that you have set the proxy details in System Settings.

To use this pattern, provide suitable values for the following parameters:

- **Name**: Select the place name.
- **Street**: Select the street name.
- **City**: Select the city name.
- **Region**: Select the region.
- **Country**: Select the country. You can hard code the value such as US for United States and IN for India.
- **Postal Code**: Select the zip or postal code.

The output from the pattern are the latitude and longitude corresponding to the input.

6.1.4.5 Calculating Distance between Objects in a Stream

Use the **Interaction: Single Stream** pattern to get interaction of an object with every other object in a stream.

For example, you can see if a set of sailing ships are too close to each other.

To use this pattern, provide suitable values for the following parameters:

- **Geometry**: Select the field that contains the shape of the object and is of type `SDO_GEOMETRY`.
- **Object Key**: Select the field used to uniquely identify an object and is used for partitioning of data where supported.
- **Coordinate System**: The default value is 8307 and this is the only value supported.

The outgoing shape contains two more fields along with the incoming shape: `isInteract` and `distance`. `isInteract` is `true`if two shapes interact with each other, i.e., any or some portion of the two objects overlap. `distance` between them is `0`, if no overlapping is observed; `isInteract` is `false` and `distance` is shown between those two objects as a positive number.

6.1.4.6 Calculating Distance between Objects in Two Streams

Use the **Interaction: Two Stream** pattern to get interaction of an object in one stream with objects in another stream. Two shapes are said to interact with each other if any part of the shape overlaps. If two shapes interact, the distance between them is *zero*.

To use this pattern, provide suitable values for the following parameters:

- **Geometry**: Select a suitable value for geometry.
- **Object Key**: Select the object key.
- **Event Stream 2**: Select the second event stream.
- **Geometry**: Select a value for geometry within the second stream.
- **Object Key**: Select the object key within the second stream.
- **Coordinate System**: The default value is 8307 and this is the only value supported.

The outgoing shape contains two additional fields along with the incoming shape: `isInteract` and `distance`. `isInteract` is `true`if two shapes interact with each other, i.e., any or some portion of the two objects overlap. `distance` between them is `0`, if no overlapping is observed; `isInteract` is `false` and `distance` is shown between those two objects as a positive number.

6.1.4.7 Creating Geo Fence

Use the **Point to Polygon** pattern to create a Geo Fence using the default coordinate system, given the latitude, longitude, width, and length.

For example, if you know the length and breadth of a group of a fleet of ships, you can get the shape of a ship using the position coordinates, where the coordinates keep changing as the ship moves.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select the field containing latitude value.
- **Longitude**: Select the field containing the longitude value.
- **Object Key**: Select a suitable value for the object key.
- **Length**: Select field that contains the length of the object.
- **Width**: Select a field that contains the width of the object.
- **Coordinate System**: The default value is 8307 and this is the only value supported.
- **Buffer**: Enter a positive value to be used as the geometry buffer.

The outgoing shape contains derived shape (`Rectangle/Polygon` of type `SDO_Geometry`) of an event based on its coordinate (`latitude`,`longitude`) and dimension (`length`, `width`).

6.1.4.8 Monitoring Proximity between Objects in a Stream

Use the **Proximity: Single Stream** pattern to get proximity of each object with every other object in a stream.

For example, if there is stream of flying airplanes and the distance buffer is 1000 meters. You can raise an alert as the two planes come into a proximity of 1000 meters or less.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select field that contains the latitude value.
- **Longitude**: Select field that contains the longitude value.
- **Object Key**: Select field that uniquely identifies the object in the stream.
- **Coordinate System**: The default value is 8307 and this is the only value supported.
- **Distance Buffer**: Enter a proximity value for the distance buffer. Proximity of objects will be output only when they are within the specified distance buffer. Select an appropriate unit for the distance. This value must be less than 10000 kilometers.

The outgoing shape displays **distance** as another column, which is the distance between two object under consideration along with the incoming shape.

6.1.4.9 Monitoring Proximity between Objects in Two Streams

Use the **Proximity: Two Stream** pattern to determine the proximity between object in stream 1 with all other objects in stream 2.

The distance buffer acts as a filter in this pattern stage. For example, if there is a driver and passenger stream, you can get the proximity of each passenger with every other driver using a filter criteria of `within a distance of 1 km`.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select field containing latitude value from stream 1.
- **Longitude**: Select field containing longitude value from stream 1.
- **Object Key**: Select field that uniquely identifies object in stream 1.
- **Event Stream 2**: Select the second event stream.
- **Latitude**: Select field containing latitude value from stream 2.
- **Longitude**: Select field containing longitude value from stream 2.
- **Object Key**: Select field that uniquely identifies object in stream 2.
- **Coordinate System**: The default value is 8307 and this is the only value supported.
- **Distance Buffer**: Enter a proximity value for the distance buffer. This field acts as a filter criteria of two objects and the objects that do not fall in this distance (distance between them is more than chosen distance buffer) are filtered from result set. This value must be less than 10000 kilometers.

> **Note:**
>
> When a pipeline with this pattern has a database reference with cache enabled, the pattern does not display any output in the live output stream.

The outgoing shape displays **distance** as another column, which is the distance between two object under consideration along with the incoming shape.

6.1.4.10 Obtaining the Proximity of an Object from a Geo Fence

Use the **Proximity: Stream with Geo Fence** pattern to get proximity of an object with a virtual boundary or geo fence.

For example, if you have certain stores in the city of California, you can send promotional messages as soon as the customer comes into a proximity of 1000 meters from any of the stores.

To use this pattern, provide suitable values for the following parameters:

- **Geo Fence**: Select a geo fence that you like to analyze.
- **Latitude**: Select the field containing latitude value.
- **Longitude**: Select the field containing longitude value.
- **Object Key**: Select the field that uniquely identifies object. Example, Vehicle Id.
- **Coordinate System**: The default value is 8307 and this is the only value supported.
- **Distance Buffer**: Enter a proximity value for the distance buffer. This field acts as a filter criteria for events and the events that do not fall in this distance (distance between them is more than chosen distance buffer) are filtered from result set. This value must be less than 10000 kilometers.

The outgoing shape displays **distance** as another column, which is the distance between the object and geo fence under consideration along with the incoming shape.

6.1.4.11 Finding Nearest Place using the Geographical Coordinates

Use the **Reverse Geo Code: Near By** pattern to obtain nearest place for the specified geographical coordinates.

Ensure that you have set the proxy details in System Settings.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select the latitude.
- **Longitude**: Select the longitude.
- **Object Key**: Select the object key.
- **Coordinate system**: The default value is 8307 and this is the only value supported.

The outgoing shape displays **PlaceName** as an additional column along with the incoming shape. This column is the nearest place for specified longitude and latitude.

6.1.4.12 Finding Nearest Place Details using the Geographical Coordinates

Use the **Reverse Geo Code: Near By Place** pattern to obtain the near by location with granular information like city, country, street etc. for the specified latitude and longitude.

Ensure that you have set the proxy details in System Settings.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select a field containing the latitude value.
- **Longitude**: Select a field containing the longitude value.
- **Object Key**: Select a field that uniquely identifies the object.
- **Coordinate system**: The default value is 8307 and this is the only value supported.

The outgoing shape displays additional columns for place corresponding to the coordinates (`latitude`,`longitude`)- `houseNumber`, `street`, `city`, `region`, `country`, and `postal code`.

6.1.4.13 Determining Average Speed

Use the **Spatial: Speed** pattern to determine the average speed using all data points from a time window. This pattern uses the default slide value. For example, to analyze the average speed of a car.

To use this pattern, provide suitable values for the following parameters:

- **Latitude**: Select field name that contains the latitude value.
- **Longitude**: Select field name that contains the longitude value.
- **Object Key**: Select a suitable value for the object key. Example of object key is Vehicle ID, or anything that uniquely differentiates the moving object.
- **Coordinate System**: The default value is 8307 and this is the only value supported.
- **Window Range**: Select a time range over which the speed is being calculated for an event. For example, if `window range=5 seconds` and object key is `vehicle ID`, then all the events with same `vehicle ID` received over last 5 seconds are used to calculate the average speed of that event.

The outgoing shape contains **speed** as an added field along with the incoming fields. This is a numeric field, but the **speed** is measured in `miles per hour`.

6.2 Transforming and Analyzing Data using Patterns

The visual representation of the event stream varies from one pattern type to another based on the key fields you choose. A pattern provides you with a simple way to explore event streams, based on common business scenarios.

To access the available patterns:

- On the Home page, click **Patterns**.

To view all the available patterns:

- Click **View All** under the **Show Me** panel.

To view a specific category of patterns:

- Click the category name(s) in the **Show Me** panel. All the patterns in the selected categories are displayed.
-

Category	Pattern
Enrichment	Reverse Geo Code: Near By Left Outer Join
Outlier	Fluctuation
Inclusion	Union Left Outer Join
Missing Event	'A' Not Followed by 'B' Detect Missing Event

Category	Pattern
Spatial	Proximity: Stream with Geo Fence Geo Fence Spatial: Speed Interaction: Single Stream Reverse Geo Code: Near By Geo Code Spatial: Point to Polygon Interaction: Two Stream Proximity: Two Stream Direction Reverse Geo Code: Near By Place Proximity: Single Stream Geo Filter
Filter	Eliminate Duplicates Fluctuation
State	'A' Not Followed by 'B' Inverse W Detect Missing Event W 'A' Followed by 'B' ‘B’ Not Preceded by ‘A’ Delay Event Time Window Snapshot Row Window Snapshot Current And Previous Pattern
Finance	Inverse W W
Shape Detector	Inverse W W
Trend	'A' Not Followed by 'B' Top N Change Detector Up Trend Detect Missing Event Down Trend 'A' Followed by 'B' Detect Duplicates Bottom N
Machine Learning	Oracle Machine Learning Service
Statistical	Correlation Quantile
Transform	ToJson Split

6.2.1 Adding a Pattern Stage

A pattern is a template of an Oracle GoldenGate Stream Analytics application, with a business logic built into it. You can create pattern stages within the pipeline. Patterns are not stand-alone artifacts, they need to be embedded within a pipeline.

For detailed information about the various type of patterns, see Transforming and Analyzing Data using Patterns.

To add a pattern stage:

1. Open a pipeline in the **Pipeline Editor**.
2. Right-click the stage after which you want to add a pattern stage, click **Add a Stage**, and then select **Pattern**.
3. Choose the required pattern from the list of available patterns.
4. Enter a **Name** and **Description** for the pattern stage.

   The selected pattern stage is added to the pipeline.
5. Click **Parameters** and provide the required values for the parameters.
6. Click **Visualizations** and add the required visualizations to the pattern stage.

6.2.2 Detecting Missing Events

Use the **Detect Missing Event** pattern to detect missing events. For example, if a feed has multiple sensors sending readings every 5 seconds, this pattern detects sensors that have stopped sending readings. This also indicates that the sensors are either broken or disconnected.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select a field as a partition criterion.

  For example, your stream contains events issues by a number of sensors. All sensors send the same but individual data. You would want to compare readings of a sensor to previous readings of the same sensor and not just a previous event in your stream, which is very likely to be from a different sensor. Select a field that would uniquely identify your sensors, such as sensor id. This field is optional. For example, if your stream contains readings from just one sensor, you do not need to partition your data.
- **Window**: Enter a time period, within which missing events are detected. If there is no event from a sensor within this specified time interval after the last event, an alert is triggered.

Outgoing Shape

The outgoing shape is the same as incoming shape. If there are no missing heartbeats, no events are output. If there is a missing heartbeat, the previous event, which was used to calculate the heartbeat interval is output.

6.2.3 Calculating Quantile Value

Use the **Quantile** pattern to calculate the value of quantile function. It returns the percentile value of all data in the specified window range. For example, a 25th percentile of a dataset is a value where 25% of data points are less than the value returned from the 25th percentile.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select a field as the partition criteria.
- **Observable Parameter**: Select as field as the parameter to calculate the quantile.
- **Phi-quantile**: Select the percentile value to calculate the quantile of the selected event stream. Values can only be from 1 to 99.
- **Window**: Select the range that determines the amount of data to consider.
- **Slide**: Select the frequency for newly updated output to be pushed downstream and into the browser.

The outgoing shape is the same as the incoming shape.

6.2.4 Identifying Correlation between Two Numeric Patterns

Use the **Correlation Pattern** pattern to identify the correlation between two numeric parameters. The output will define if the two parameters are positively correlated (value of 1), or negatively correlated (-1), or not correlated(value of 0).

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select a field that uniquely identifies the object, for example, Sensor ID.
- **Observable Parameter 1**: Select first field to correlate.
- **Observable Parameter 2**: Select second field to correlate.
- **Window**: Select a time range to retain the data, while identifying the correlation between parameter 1 and parameter 2. Default slide value is used when no slide value is specified. A slide value same as window range will output the correlation at the end of the time window. Slide value less than the window range will output more frequently.
- **Slide**: Set the frequency at which you want to refresh the data.

The outgoing shape is same as the incoming shape.

6.2.5 Detecting Duplicate Events

The Detect Duplicates pattern detects duplicate events in your stream according to the criteria you specify and within a specified time window. Events may be partially or fully equivalent to be considered duplicates.

For example, when you suspect that your aggregates are offset, you can check your stream for duplicate events.

To use this pattern, provide suitable values for the following parameters:

- **Duplicate Criteria**: Select the fields to be compared. If all the configured fields have identical values, the incoming event will be considered a duplicate and an outgoing event will be fired.
- **Window**: Select the time period within which to search for duplicates.

  For example, if you set the window to 10 seconds, a duplicate event that arrives 9 seconds after the first one will trigger an outgoing event, while a duplicate event that arrives 11 seconds after the first one will not do so.

Outgoing Shape

The outgoing shape is the same as the incoming shape with one extra field: `Number_of_Duplicates`. This extra field will carry the number of duplicate events that have been discovered. All the other fields will have values of the last duplicate event.

6.2.6 Eliminating Duplicate Events

Use the **Eliminate Duplicates** pattern to look for duplicate events in your stream within a specified time window, and remove all but the first occurrence. A duplicate event is an event that has one or more field values identical to values of the same field(s) in another event. You can specify what fields are analyzed for duplicate values. You can configure the pattern to compare just one field or the whole event.

For example, use it when you know that your stream contains duplicates that might offset your aggregates, such as counts.

To use this pattern, provide suitable values for the following parameters:

- **Duplicate Criteria**: Select the fields to be compared. If all the configured fields have identical values, the second, third, and subsequent events will be dropped.
- **Window**: Select a time period, within which the duplicates should be discarded.

  For example, if you set the window to 10 seconds, a duplicate event that arrives 9 seconds after the first one will be discarded, while a duplicate event that arrives 11 seconds after the first one will be accepted and let through.

The outgoing shape is the same as the incoming shape.

6.2.7 Detecting Event Value Changes

Use the **Change Detector** pattern to look for changes in the values of your event fields and report the changes once they occur within a specified range window. For example, if an event arrives with value `value1` for field `field1`, and any of the following incoming events, within a specified range window, contains a value different from `value1`, an alert is triggered. You can designate more than one field to look for changes.

For example, a sensor reading that is supposed to be the same for certain periods of time and changes in readings may indicate issues.

The default configuration of this pattern stage is to alert on change of any selected fields.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select the partition criteria.

  For example, your stream contains events issued by a number of sensors. All sensors send the same but individual data. You would want to compare readings of a sensor to previous readings of the same sensor and not just a previous event in your stream, which is very likely to be from a different sensor. Select a field that would uniquely identify your sensors, such as sensor Id. This field is optional. For example, if your stream contains readings from just one sensor, you do not need to partition your data.
- **Window range**: Select a time period within which the values of designated fields are compared for changes.

  For example, if you set the window range to 10 seconds, an event with changes in observed fields will trigger an alert if it arrives within 10 seconds after the initial event. The clock starts at the initial event.

- **Change Criteria**: Select a list of fields to be compared. If the fields contain no changes, no alerts will be generated.
- **Alert on group changes**: Select this option default group changes support. If it is `OFF`, then alert on at least one field changes. If it is `ON`, then sends alert on every field change.

Outgoing Shape

The outgoing shape is based on the incoming shape, the difference being that all the fields except the one in the partition criteria parameter will be duplicated to carry both the initial event values and the change event values.

Example:

Your incoming event contains the following fields:

- `sensor_id`
- `temperature`
- `pressure`
- `location`

Normally, you would use sensor_id to partition your data, to look for changes in temperature. So, select sensor_id in the partition criteria parameter and temperature in the change criteria parameter. Use a range window that fits your use case. In this scenario, you will have the following outgoing shape:

- `sensor_id`
- `temperature`
- `orig_temperature`
- `pressure`
- `orig_pressure`
- `location`
- `orig_location`

The *orig_* fields carry values from the initial event. In this scenario, `temperature` and `orig_temperature` values are different, while pressure and `orig_pressure`, `location`, and `orig_location` may have identical values.

6.2.8 Detecting Data Field Value Changes

Use the **Fluctuation** pattern to detect when an event data field value changes in a specific upward or downward fashion within a specific time window. For example, use this pattern to identify the variable changes in an Oil Pressure value are maintained within acceptable ranges.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select the fields to be used as partition criteria.
- **Tracking Value**: Select a field value to track the event data and create a pattern in the live output stream.
- **Window**: Select a rolling time period, the frequency at which you want to refresh the data.
- **Deviation Threshold %**: Select the percentage of deviation you want to be included in the pattern. This is the interval in which the pipeline looks for a matching pattern.

The outgoing shape is same as the incoming shape.

6.2.9 Monitoring Sequence of Events

Use the **'A' Followed by 'B'** pattern to look for particular events following one another and to output an event when the specified sequence of events occurs.

Use it when you need to be aware of a certain succession of events happening in your flow. For example, if an order status `BOOKED` is followed by an order status `SHIPPED` (skipping status `PAID`), you need to raise an alert.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select the fields to be used as partition criteria. In the order example above, it may be `order_id`.
- **State A: field**: Select an initial state field, whose value will be used in the comparison of two events. In our example, it will be `order_status`.
- **State A: value**: Select the initial field state value. In our example, `BOOKED`.
- **State B: field**: Select a consecutive state field, whose value will be used in the comparison of two events. In our example, it will be `order_status` again.
- **State B: value**: Select the consecutive field state value. In our example, `SHIPPED`.
- **Duration**: Select the time period within which to look for state changes.

Outgoing Shape

The outgoing shape is based on the incoming shape. A new `abInterval` field is added to carry the value of the time interval between the states in nanosecond. Also, all but the partition criteria fields are duplicated to carry values from both a and b states. For example, if you have the following incoming shape:

- `order_id`
- `order_status`
- `order_revenue`

You will get the following outgoing shape:

- `order_id`
- `abInterval`
- `order_status` (this is the value by which you partition your stream)
- `aState_order_status` (this is the value of order_status in state A, in our example 'BOOKED')
- `order_revenue` (this is the value of order_revenue in state B)
- `aState_order_revenue` (this is the value of order_revenue in state A)

6.2.10 Outputting Highest Value Events

Use the **Top N** pattern to output N events with highest values from a collection of events, arriving within a specified time window. The events here are sorted the way you specify, and not in the default order of arrival.

For example, use it to get N highest values of pressure sensor readings.

To use this pattern, provide suitable values for the following parameters:

- **Window Range**: Select a rolling time period within which the events will be collected and ordered per your ordering criteria.
- **Window Slide**: Select the frequency for the newly updated output to be pushed downstream and into the browser.
- **Order by Criteria**: Select a list of fields to use to order the collection of events.
- **Number of Events**: Select the number of top value events to output.

The outgoing shape is the same as the incoming shape.

6.2.11 Outputting Lowest Value Events

Use the **Bottom N** pattern to output N events with lowest values from a collection of events, arriving within a specified time window. The events here are sorted the way you specify and not in the default order of arrival.

For example, use it to get N lowest values of pressure sensor readings.

To use this pattern, provide suitable values for the following parameters:

- **Window Range**: Select a rolling time period within which the events will be collected and ordered per your ordering criteria.
- **Window Slide**: Select the frequency for newly updated output to be pushed downstream and into the browser.
- **Order by Criteria**: Select a list of fields to use to order the collection of events.
- **Number of Events**: Select the number of bottom value events to output.

The outgoing shape is the same as the incoming shape.

6.2.12 Monitoring Invariably Increasing Numeric Values

Use the **Up Trend** pattern to detect an invariably increasing numeric value, over a period of time.

Use the pattern if you need to detect situations of a constant increase in one of your numeric values. For example, detect a constant increase in pressure from one of your sensors.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select fields to be used as partition criteria.

  For example, your stream contains events issues by a number of sensors. All sensors send the same but individual data. You would want to compare readings of a sensor to previous readings of the same sensor and not just a previous event in your stream, which is very likely to be from a different sensor. Select a field that would uniquely identify your sensors, such as sensor id. This field is optional. For example, if your stream contains readings from just one sensor, you do not need to partition your data.
- **Duration**: Select a time period within which the values of the designated field are analyzed for the upward trend.
- **Tracking value**: Select a field to be analyzed for upward trend.

Outgoing Shape

The outgoing shape is based on the incoming shape with an addition of two new fields. For example, if your incoming event contains the following fields:

- `sensor_id`
- `temperature`
- `pressure`
- `location`

Normally, you would use `sensor_id` to partition your data and say you want to look for the upward trend in temperature. So, select `sensor_id` in the partition criteria parameter and temperature in the tracking value parameter. Use a duration that fits your use case. In this scenario, you will have the following outgoing shape:

- `sensor_id`
- `startValue` (this is the value of temperature that starts the trend)
- `endValue` (this is the value of temperature that ends the trend)
- `temperature` (the value of the last event)
- `pressure` (the value of the last event)
- `location` (the value of the last event)

6.2.13 Monitoring Invariably Decreasing Numeric Values

Use the **Down Trend** pattern to detect an invariably decreasing a numeric value, over a period of time.

For example, detect a constant drop in pressure from one of your sensors.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select a field to be used as a partition criterion.

  For example, your stream contains events issues by a number of sensors. All sensors send the same but individual data. You would want to compare readings of a sensor to previous readings of the same sensor and not just a previous event in your stream, which is very likely to be from a different sensor. Select a field that would uniquely identify your sensors, such as sensor id. This field is optional. For example, if your stream contains readings from just one sensor, you do not need to partition your data.
- **Duration**: Select a time period within which the values of the designated field are analyzed for the downward trend.
- **Tracking value**: Select a field to be analyzed for downward trend.

Outgoing Shape

The outgoing shape is based on the incoming shape with an addition of two new fields. Let's look at an example. Your incoming event contains the following fields:

- `sensor_id`
- `temperature`
- `pressure`

- `location`

Normally, you would use `sensor_id` to partition your data and say you want to look for the downward trend in temperature. So, select `sensor_id` in the partition criteria parameter and temperature in the tracking value parameter. Use a duration that fits your use case. In this scenario, you will have the following outgoing shape:

- `sensor_id`
- `startValue` (this is the value of temperature that starts the trend)
- `endValue` (this is the value of temperature that ends the trend)
- `temperature` (the value of the last event)
- `pressure` (the value of the last event)
- `location` (the value of the last event)

The pattern is visually represented based on the data you have entered/selected.

6.2.14 Identifying the Missing First Event in a Sequence

The 'B' Not Preceded by 'A' pattern will look for a missing event in a particular combination of events and will output the first event which is found where the first event is not preceded by the second event.

Use it when you need to be aware of a specific event not preceded by another event in your flow. For example, if an order status `BOOKED` is not preceded by an order status `PAID` within a certain time period, you may need to raise an alert.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: (Optional) a field to partition your stream by. In the order example above, it may be `order_id`.
- **State A: Field**: an initial state field, whose value will be used in the comparison of two events. In our example, it will be `order_status`.
- **State A: Value**: the initial field state value. In our example, `BOOKED`.
- **State B: Field**: a consecutive state field, whose value will be used in the comparison of two events. In our example, it will be `order_status` again.
- **State B: Value**: the consecutive field state value. In our example, `PAID`.
- **Duration**: the time period, within which to look for state changes.

Outgoing Shape

The outgoing shape is the same as incoming shape. If the second (state B) event does not arrive within the specified time window, the first (state A) event is pushed to the output.

6.2.15 Identifying the Second Missing Event in a Sequence

The 'A' Not Followed by 'B' pattern will look for a missing second event in a particular combination of events and will output the first event when the expected second event does not arrive within the specified time period.

Use it when you need to be aware of a specific event not following its predecessor in your flow. For example, if an order status `BOOKED` is not followed by an order status `PAID` within a certain time period, you may need to raise an alert.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: (Optional) a field to partition your stream by. In the order example above, it may be `order_id`.
- **State A: field**: an initial state field, whose value will be used in the comparison of two events. In our example, it will be `order_status`.
- **State A: value**: the initial field state value. In our example, `BOOKED`.
- **State B: field**: a consecutive state field, whose value will be used in the comparison of two events. In our example, it will be `order_status` again.
- **State B: value**: the consecutive field state value. In our example, `SHIPPED`.
- **Duration**: the time period, within which to look for state changes.

Outgoing Shape

The outgoing shape is the same as incoming shape. If the second (state B) event does not arrive within the specified time window, the first (state A) event is pushed to the output.

6.2.16 Analyzing Data using Double Bottom Charts

Use the **W** pattern for technical analysis of financial trading markets. This pattern is also known as a double bottom chart pattern.

Use this pattern to detect when an event data field value rises and falls in “W” fashion over a specified time window. For example, use this pattern when monitoring a market data feed stock price movement to determine a buy/sell/hold evaluation.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select a field to be used as a partition criterion. For example, a ticker symbol.
- **Window**: Select a time period within which the values of the designated field are analyzed for the W shape.
- **Tracking value**: Select a field to be analyzed for the W shape.

Outgoing Shape

The outgoing shape is based on the incoming shape with an addition of five new fields. The new fields are:

- `firstW`
- `firstValleyW`
- `headW`
- `secondValleyW`
- `lastW`

The new fields correspond to the tracking value terminal points of the W shape discovered in the feed. The original fields correspond to the last event in the W pattern.

6.2.17 Analyzing Data using Double Top Charts

Use the **Inverse W** pattern for the technical analysis of financial trading markets, and to see the financial data in a graphical form. This pattern is also known as a double top chart pattern.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select a field to be used as a partition criterion. For example, a ticker symbol.
- **Window**: Select a time period within which the values of the designated field are analyzed for the inverse W shape.
- **Tracking value**: Select a field to be analyzed for the inverse W shape.

Outgoing Shape

The outgoing shape is based on the incoming shape with an addition of five new fields. The new fields are:

- `firstW`
- `firstPeakW`
- `headInverseW`
- `secondpeakW`
- `lastW`

The new fields correspond to the tracking value terminal points of the inverse W shape discovered in the feed. The original fields correspond to the last event in the inverse W pattern.

6.2.18 Correlating Current and Previous Events

Use the **Current and Previous Events** pattern to automatically correlate the current and previous events.

To use this pattern, provide suitable values for the following parameters:

- **Partition Criteria**: Select the fields to be used the partition criteria.

Input Schema or Payload Shape `[DiskID, Usage]`

Output schema/payload from this pattern for non-partitioned input is `[DiskID, Usage, PREV_DiskId, PREV_Usage]`.

For input partitioned by DiskID, the output schema from the pattern is `[DiskID, Usage, PREV_Usage]`.

Below is an example with values:

For non-partitioned input:

Input - `[Disk1, 40gb], [Disk2, 60gb], [Disk1, 45gb]`

Output - `[Disk2, 60gb, Disk1, 40gb], [Disk1, 45gb, Disk2, 60gb]`

For input partitioned by Disk ID :

Input - `[Disk1, 40gb], [Disk2, 60gb], [Disk1, 45gb]`

Output `[Disk1, 45gb, 40gb]`. There is no output for Disk2 until another event for Disk2 arrives.

6.2.19 Delaying Delivery of Events to Downstream Node

Use the **Delay Event** pattern to delay delivering an event to downstream node in the pipeline, for a specified number of seconds. A practical use case is to wind up a campaign event or promotion.

To use this pattern, provide suitable values for the following parameters:

- **Delay in Seconds**: Select a time period for which you want to delay the processing of an event.

6.2.20 **Outputting Contents to Downstream Node**

Use the **Row Window Snapshot** pattern to output entire window contents to a downstream node, on the arrival of a new event, based on the specified maximum number of events a window can hold.

For example:

- To rebuild an ML model in real-time
- To continually use the last X values in a time-series forecasting algorithm, to predict future values

To use this pattern, provide suitable values for the following parameters:

- **Maximum number of rows the window will hold**
- **Key fields for partitioning the window**
- **Time in seconds before event in the window expires**

[PARTITION BY StockSymbol, ROWS 500, RANGE 1 MINUTE] will dump the entire window contents on the arrival of a new event. The window will hold a maximum of 500 events. An event will expire after a minute, allowing newer events.

Partitioning key creates separate window for each value of the key. For example, a separate window for 500 Oracle quote events, 500 Microsoft quote events and so on.

6.2.21 Outputting Unexpired Contents to Downstream Node

Use the **Time Window Snapshot** pattern to output entire window contents to a downstream node, on the arrival of a new event, based on the time window specified for each event.

To use this pattern, provide suitable values for the following parameters:

- **Window range in seconds or duration in seconds before event expires**
- **Frequency in seconds for window snapshot output**

[RANGE 10 MINUTES] will dump entire window contents to downstream node on the arrival of a new event. Event will expire from the window after 10 minutes.

> **Note:**
>
> Window will automatically dump contents either when a new event arrives or when an event expires.

6.2.22 Merging Two Streams having Identical Shapes

Use the **Union** pattern to merge two streams having identical shapes.

For example, you have two similar sensors sending data into two different streams, and you want to process the streams simultaneously, in one pipeline.

To use this pattern, provide suitable values for the following parameters:

- **Second event stream**: Select the stream you want to merge with your primary stream. Make sure you select a stream with an identical shape.

The outgoing shape is the same as the incoming shape.

6.2.23 Joining Flows with Streams and References

Use the **Left Outer** join pattern to join a stream or a reference, using the left outer join semantics.

The result of this pattern always contains the data of the left table even if the join-condition does not find any matching data in the right table.

To use this pattern, provide suitable values for the following parameters:

- **Enriching Reference/Stream**: Select the stream or reference you want to join to your flow.
- **Correlation Criteria**: Select the fields based on which the stream/ reference will be joined.
- **Window Range of the Primary Stream**: Select a rolling time window to make a collection of events in your primary flow, to be joined with the enriching stream/ reference.
- **Window Slide of the Primary Stream**: Select the frequency for data to be pushed downstream and to the UI.
- **Window Range of the Enriching Stream**: Select a rolling time window to make a collection of events in your enriching stream, to be joined with the primary flow. Disabled, if a reference is used.
- **Window Slide for the Enriching Stream**: Select the frequency for the data to be pushed downstream and to the UI. Disabled, if a reference is used.

The outgoing shape is a sum of two incoming shapes.

6.2.24 Transforming Events into JSON

Use the **ToJson** pattern to transform event(s) coming from a stage in the pipeline into a JSON text.

Use this pattern to transform multiple events into a single JSON document, and send it to a downstream system through OSA pipeline targets. For example, you can configure a Database target after the toJson pattern stage, to write the json payload (of a single or multiple events), into a database table

To use this pattern, provide suitable values for the following parameters:

- **Enable batching**: Select this option to transform multiple events as an array of single JSON document and output it as JSON text. By Default, it uses all the events of a partition, within the batch duration, to transform as an array of JSON document.
- **Batch Size**: Select the number of events to be included in a batch.

If the size is configured to the value greater than 0 (say n), it will transform maximum n events as a JSON array of single JSON document and output as a JSON text.

If the size is set to the default **0**, all the events of a partition within the batch duration will be transformed as an array of single JSON document and output as JSON text.

- **Upload Json File**: You can upload a sample JSON file to be used to infer JSON path for field mapping.
- **Field Mapping**:
  - **Json Path**: Lists all the paths in uploaded JSON file.
  - **Fields**: Lists all the fields from the previous stage. You can map the JSON path with one of the fields from the drop-down list.

6.2.25 Transforming a Single Event from a Stage into Multiple Events

Use the **Split** pattern to transform a single event from a stage into multiple events, by splitting the value field. For example, you can flatten an array of json element or json object from the source, to individually process it and also to push it to some targets.

The output of this pattern, is one or more events, corresponding to each single event of the previous stage. The output events are a clone of the source event. The attribute of the selected field is split into an array, based on the selected type. Each value of an array correspond to an output event with the new value of the selected attribute.

To use this pattern, provide suitable values for the following parameters:

- **Split**: Select one of the fields from the previous event, to split.
- **Type**: Select the value type of the split field, from the drop-down list.
- **Separator**: Set the text separator for delimited text type. Default separator is comma.

6.2.26 Merging Two Continuous Events into a Single Event

Use the **Continuous Merge** pattern to merge two or more continuous events into a single event, based on the key attributes.

The output of this pattern, is a single event corresponding to two or more merged events of the previous stage. The attribute of the selected field is merged into an array, based on the selected type. Each value of an array correspond to an output event with the new value of the selected attribute.

To use this pattern, provide suitable values for the following parameters:

- **Key Fields**: Select one or more attributes from the previous stage.
- **Merged Field Name**: Enter a name for the merged, output field.

6.2.27 Applying OML Models to get the Scoring of Events (Preview Feature)

Use the **Oracle Machine Learning Service** pattern to use OML models to apply the scoring on the ingested events.

To use this pattern, provide suitable values for the following parameters:

- **OML server url**: Enter the OML service endpoint where the autonomous data warehouse is located for the Machine Learning model in the region.
- **Tenant**: Enter the tenant ID hosting the OML model.

- **OML Service Name.**: Provide the OML Service Name.
- **Username**: Username for the OML service or the ADW database, for the OML user.
- **Password**: Password for the OML service or the ADW database, for the OML user.
- **OML Model**: Select an OML model that you want to apply to the current stage.
- **Input Fields**: Choose the input parameter fields to map with the OML model.

6.2.28 Detecting Contiguous Events

Use the **Segment Detector** pattern to detect contiguous events (range/segment) having unchanged values for the selected attributes, over a specified period of time.

For example, to detect the range of stability over a period of time in:

- The range of constant speed of a vehicle
- The range of constant/ stable temperature of an electric appliance

To use this pattern, provide suitable values for the following parameters:

- **Event Stream**: Event stream is the stream (previous stage) over which the pattern will be applied.
- **Time Window**: Time window specifies the period of time during which this pattern detect the segment.
- **Equality Criteria**: Enter the attributes (criteria) to detect the segment.
- **Partition Criteria**: Enter the attributes (criteria) to partition the output.
- **Alert on group changes**: Select for notifications on all observable parameters changes.

The Segment Detector pattern outputs the attribute values of the first and last event in the detected segment (range). Each of the attributes of the first event is prefixed with orig_ in the output shape, whereas the attribute of the last event name is same as in the previous stage.

6.2.29 Creating Pivot Columns

Use the **Pivot** pattern to pivot all or selected attributes of an incoming event, into new columns, based on the selected pivot value and key provided.

To use this pattern, provide suitable values for the following parameters:

- **Event Stream**: Event stream is the stream (previous stage) over which the pattern will be applied.
- **Time Window**: Time window specifies the period of time during which the pivot value for the pivot key gets memorized.
- **Pivot Value**: Select the attributes to pivot.
- **Pivot Key**: Select the values of the selected attributes in **Pivot Value**,that will be displayed as new attributes in the output.
- **Partition Criteria**: Enter the attributes (criteria) to partition the output.
- **Retain pivoted columns**: When selected, it includes the pivoted column (pivot value) in the output
- **Keep all events**: When selected, it outputs all the events from the previous stage.

6.3 Using Machine Learning Models for Scoring and Prediction

The predictive model is an algorithm that you apply to streaming data to predict outcomes. In a pipeline, you use a predictive model in a scoring stage to do probability scoring.

In GoldenGate Stream Analytics, a predictive model is an ONNX file, that you upload and store in the system:

- GGSA supports ONNX models with single-dimensional outputs of size 1. The output should be of `int`, `float`, `double` or `Boolean` datatype.

6.3.1 Importing a Predictive Model

To import a predictive model:

1. On the **Catalog** page, click **Create New Item**, and select **Predictive Model** from the drop-down list.
2. On the **Type Properties** screen, enter the following details and click **Next** :
   - **Name**
   - **Description**
   - **Tags**
   - **Predictive Model Type**
- On the **Predictive Model Details**, enter the following details and click **Save**:
  1. For **Predictive Model URL**, upload your ONNX file.
  2. In the **Model Version** field, enter the version of this artifact. For example, `1.0`.
  3. (Optional) In the **Version Description**, enter a meaningful description for your model.
  4. In the **Algorithm** field, accept the default. The algorithm is derived from the model you have uploaded.
  5. (Optional) In the **Tool** drop-down list, select the tool with which you created your model.

6.3.2 Adding a Scoring Stage

To add a scoring stage:

1. Open the required pipeline in Pipeline Editor.
2. Right-click the stage after which you want to add a scoring stage, click **Add a Stage**, and then select **Scoring**.
3. Enter a meaningful name and suitable description for the scoring stage and click **Save**.
4. In the stage editor, enter the following details:
   a. **Model name**: Select the predictive model that you want to use in the scoring stage
   b. **Model Version**: Select the version of the predictive model
   c. **Mapping**: Select the corresponding model fields that appropriately map to the stage fields

You can add multiple scoring stages based on your use case.

6.4 Integrating with Druid Timeseries Database for Realtime Interactive Analytics

A cube is a data structure that helps you to quickly analyze data, on multiple dimensions. GoldenGate Stream Analytics cubes are powered by Druid, which is a distributed, in-memory OLAP data store.

A pipeline outputs the processed data into the Kafka streams which in turn feeds the cube.

You can use cubes for:

- Interactive analysis of historical data
- Analysis of univariate, bivariate, and multivariate data
- Exploration of historical data, using a rich set of 30 visualizations. These visualizations range from simple table, line, bar to the advanced visualizations such as sankey, boxplot, maps, etc. You can save the result of the cube explorations to use them with dashboards, for both the operational and strategical analysis needs of the business users.

6.4.1 Creating a Connection to Druid

To create a connection to Druid:

1. On the **Catalog** page, click **Create New Item**.
2. Hover the mouse over **Connection** and select **Druid** from the submenu.
3. On the **Type Properties** screen, enter the following details:
   - **Name**: Enter a unique name for the connection. This is a mandatory field.
   - **Display Name**: Enter a display name for the connection. If left blank, the **Name** field value is copied.
   - **Description**
   - **Tags**
   - **Connection Type**: The selected connection is displayed.
4. Click **Next**.
5. On the **Connection Details** screen, enter the following details:
   - **Zookeepers**: Enter the zookeeper URL.
6. Click **Test Connection**, to ensure that you have successfully created a connection.
7. Click **Save.**

6.4.2 Creating a Cube

Druid must be running, prior to creating a cube. Visit http://druid.io/downloads.html, to download and install Druid.

To create a cube:

1. On the **Catalog** page, click **Create New Item**, and select **Cube** from the drop-down list.
2. On the **Type Properties** screen, enter the following details:

- **Name**
- **Description**
- **Tags**
- **Source Type**: Select **Published Pipeline** from the drop-down list.

3. Click **Next**.
4. On the **Ingestion Details** screen, enter the following details:
   - **Connection**: Select a connection from the drop-down list.
   - **Pipeline**: Select a pipeline from the drop-down list.
   - **Kafka Target** Select a Kafka target from the drop-down list.
   - **Timestamp**: Select a column from the pipeline to be used as the timestamp.
   - **Timestamp format**: Select or set a suitable format for the timestamp using Joda time format. This is a mandatory field. The default value is *auto*.
   - **Metrics**: Select metrics for creating measures.
   - **Dimensions**: Select dimensions for group by.
   - **High Cardinality Dimensions**: Select high cardinality dimensions such as unique IDs. Hyperlog approximation will be used.
5. Click **Next**.
6. Select the required values for the metric on the **Metric Capabilities** screen.
7. On the **Advanced Settings** screen, enter the following details:
   - **Segment granularity**: Select the granularity with which you want to create segments
   - **Query granularity**: Select the minimum granularity to be able to query results and the granularity of the data inside the segment
   - **Task count**: Select the maximum number of reading tasks in a replica set. This means that the maximum number of reading tasks is `taskCount*replicas` and the total number of tasks (reading + publishing) is higher than this. The number of reading tasks is less than `taskCount if taskCount > {numKafkaPartitions}`.
   - **Task duration**: Select the length of time before tasks stop reading and begin publishing their segment. The segments are only pushed to deep storage and loadable by historical nodes when the indexing task completes.
   - **Maximum rows in memory**: Enter a number greater than or equal to 0. This number indicates the number of rows to aggregate before persisting. This number is the post-aggregation rows, so it is not equivalent to the number of input events, but the number of aggregated rows that those events result in. This is used to manage the required JVM heap size. Maximum heap memory usage for indexing scales with `maxRowsInMemory*(2 + maxPendingPersists)`.
   - **Maximum rows per segment**: Enter a number greater than or equal to 0. This is the number of rows to aggregate into a segment; this number is post-aggregation rows.
   - **Immediate Persist Period**: Select the period that determines the rate at which intermediate persists occur. This allows the data cube is ready for query earlier before the indexing task finishes.
   - **Report Parse Exception**: Select this option to throw exceptions encountered during parsing and halt ingestion.

- **Advanced IO Config**: Specify name-value pair in a CSV format. Available configurations are `replicas`, `startDelay`, `period`, `useEarliestOffset`, `completionTimeout`, and `lateMessageRejectionPeriod`.
- **Advanced Tuning Config**: Specify name-value pair in CSV format. Available configurations are `maxPendingPersists`, `handoffConditionTimeout`, `resetOffsetAutomatically`, `workerThreads`, `chatThreads`, `httpTimeout`, and `shutdownTimeout`.

8. Click **Save**.

6.4.3 Exploring a Cube

When you create druid based cube, you can explore data in it.

To explore a cube:

1. In the **Catalog**, click the cube that you want to explore.
2. On the **Cube Exploration**screen, construct a query by setting the following parameters:
   - **Visualization Type**: Select a visualization type from the drop-down list.
   - **Time**: Set the time-related form attributes such as time granularity, origin (starting point of time), and time range

     .
   - **Group By**: Select the parameters to aggregate the query data

     .
   - **Not Grouped By**: Select the parameters to query atomic rows.

     .
   - Options
   - Filters — columns that you can use in filters
   - Result Filters — columns that you can use in result filters
3. Click **Query** to run the query with the defined parameters.
4. Click **Save As** to save the cube exploration. You can save it as a visualization, choose to add it to an existing dashboard, not to add it to a dashboard, or add it to a new dashboard.

7 Visualize

Visualizations are graphical representations of the streaming data in a pipeline. You can add visualizations on all the stages in a pipeline, except on a target stage.

Adding Realtime Charts

Creating and Managing Dashboards

7.1 Adding Realtime Charts

7.1.1 Adding an Area Chart

Area visualization represents data as a filled-in area. Area visualization requires at least two groups of data along an axis. The X-axis is a single consecutive dimension, such as a date-time field, and the data lines are unlikely to cross. Y axis represents the metrics (measured value). X axis can also have non date-time categories. This visualization is mainly suitable for presenting accumulative value changes over time.

To add an area visualization:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Area Chart**.
4. Enter/select values for the following fields:
   - **Name**: a suitable name for the visualization. This is a mandatory field.
   - **Description**: a suitable description. This is an optional field.
   - **Tags**: suitable tags to for easy identification. This is an optional field.
   - **Y Axis Field Selection**: the column to be used as the Y axis. This is a mandatory field.
   - **Axis Label**: a label for the Y axis. This is an optional field.
   - **X Axis Field Selection**: the column to be used as the X axis. This is a mandatory field.
   - **Axis Label**: a label for the X axis. This is an optional field.
   - **Orientation**: select this check box if you want the visualization to appear with a horizontal orientation in the Pipeline Editor. This is optional and you can decide based on your usecase or requirement if you want to change the orientation.
   - **Data Series Selection**: the column to be used as the data series. This is an optional field.
5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.2 Adding a Bar Chart

Bar visualization is one of the widely used visualization types which represents data as a series of vertical bars. It is best suited for comparison of the values represented along y axis where different categories are spread across x axis. In a Bar visualization vertical columns represent metrics (measured values). The horizontal axis displays multiple or non-consecutive categories.

To add a bar visualization:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Bar Chart**.
4. Enter/select values for the following fields:
   - **Name**: a suitable name for the visualization. This is a mandatory field.
   - **Description**: a suitable description. This is an optional field.
   - **Tags**: suitable tags to for easy identification. This is an optional field.
   - **Y Axis Field Selection**: the column to be used as the Y axis. This is a mandatory field.
   - **Axis Label**: a label for the Y axis. This is an optional field.
   - **X Axis Field Selection**: the column to be used as the X axis. This is a mandatory field.
   - **Axis Label**: a label for the X axis. This is an optional field.
   - **Orientation**: select this check box if you want the visualization to appear with a horizontal orientation in the Pipeline Editor. This is optional and you can decide based on your usecase or requirement if you want to change the orientation.
5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.3 Adding a Bubble Chart

A bubble chart is a good option when you want to add an additional dimension to a scatter plot chart. Scatter charts compare two values, but you can add bubble size as the third variable in a bubble chart and thus enable comparison. A good example to use bubble chart is to show marketing expenditures vs revenue vs profit.

To add a bubble chart:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Bubble Chart**.
4. Enter/select values for the following fields:
   - **Name**: a suitable name for the visualization. This is a mandatory field.
   - **Description**: a suitable description. This is an optional field.
   - **Tags**: suitable tags to for easy identification. This is an optional field.

- **Y Axis Field Selection**: the column to be used as the Y axis. This is a mandatory field.
- **Axis Label**: a label for the Y axis. This is an optional field.
- **X Axis Field Selection**: the column to be used as the X axis. This is a mandatory field.
- **Axis Label**: a label for the X axis. This is an optional field.
- **Bubble Size Field Selection**: select the field that you want to use as the bubble size. This is a mandatory field.

5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.4 Adding a Line Chart

Line visualization represents data as a line, as a series of data points, or as data points that are connected by a line. Line visualization require data for at least two points for each member in a group. The X-axis is a single consecutive dimension, such as a date-time field, and the data lines are likely to cross. X axis can also have non date-time categories. Y axis represents the metrics (measured value). It is preferred to use line visualization when data set is continuous in nature. It is best suited for trend-based plotting of data over a period of time.

To add a line visualization:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Line Chart**.
4. Enter/select values for the following fields:
   - **Name**: a suitable name for the visualization. This is a mandatory field.
   - **Description**: a suitable description. This is an optional field.
   - **Tags**: suitable tags to for easy identification. This is an optional field.
   - **Y Axis Field Selection**: the column to be used as the Y axis. This is a mandatory field.
   - **Axis Label**: a label for the Y axis. This is an optional field.
   - **X Axis Field Selection**: the column to be used as the X axis. This is a mandatory field.
   - **Axis Label**: a label for the X axis. This is an optional field.
   - **Orientation**: select this check box if you want the visualization to appear with a horizontal orientation in the Pipeline Editor. This is optional and you can decide based on your usecase or requirement if you want to change the orientation.
   - **Data Series Selection**: the field that you want to use for data selection series.
5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.5 Adding a Pie Chart

A pie chart is a circular graph that represents statistical data in slices. The size of each slice is proportional to the quantity of the value it represents.

To add a pie chart:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Pie Chart**.
4. Enter/select values for the following fields:
    - **Name**: a suitable name for the visualization. This is a mandatory field.
    - **Description**: a suitable description. This is an optional field.
    - **Tags**: suitable tags to for easy identification. This is an optional field.
    - **Measure**: the field to be used as the measure of the visualization. This is a mandatory field.
    - **Group**: the field to be used as the group for the visualization. This is a mandatory field.
    - **Use 3D rendering**: select this check box if you want to render the visualization with a 3D effect. This is an optional field.
5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.6 Adding a Scatter Plot

Scatter charts are primarily used for correlation and distribution analysis. This type of chart is good for showing the relationship between two different variables where one correlates to another.

To add a scatter visualization:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Scatter Chart**.
4. Enter/select values for the following fields:
    - **Name**: a suitable name for the visualization. This is a mandatory field.
    - **Description**: a suitable description. This is an optional field.
    - **Tags**: suitable tags to for easy identification. This is an optional field.
    - **Y Axis Field Selection**: the column to be used as the Y axis. This is a mandatory field.
    - **Axis Label**: a label for the Y axis. This is an optional field.
    - **X Axis Field Selection**: the column to be used as the X axis. This is a mandatory field.
    - **Axis Label**: a label for the X axis. This is an optional field.
    - **Data Series Selection**: the field that you want to use for data series selection. This is an optional field.

5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.7 Adding a Stacked Bar Chart

A stacked visualization displays sets of values stacked in a single segmented column instead of side-by-side in separate columns. It is used to show a composition. Bars for each set of data are appended to previous sets of data. The size of the stack represents a cumulative data total.

To add a stacked visualization:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Stacked Bar Chart**.
4. Enter/select values for the following fields:
   - **Name**: a suitable name for the visualization. This is a mandatory field.
   - **Description**: a suitable description. This is an optional field.
   - **Tags**: suitable tags to for easy identification. This is an optional field.
   - **Y Axis Field Selection**: the column to be used as the Y axis. This is a mandatory field.
   - **Axis Label**: a label for the Y axis. This is an optional field.
   - **X Axis Field Selection**: the column to be used as the X axis. This is a mandatory field.
   - **Axis Label**: a label for the X axis. This is an optional field.
   - **Orientation**: select this check box if you want the visualization to appear with a horizontal orientation in the Pipeline Editor. This is optional and you can decide based on your usecase or requirement if you want to change the orientation.
5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.8 Adding a Thematic Map

A thematic map is used to represent a particular theme in data connected to a geographical area. This type of map depicts the political, cultural, agricultural, sociological, and many other aspects of the geographic region, be it a city, state, country, ore region.

To add a thematic map:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required stage and click the **Visualizations** tab.
3. Click **Add a Visualization** and then click **Thematic Map**.
4. Enter/select values for the following fields:
   - **Name**: a suitable name for the visualization. This is a mandatory field.
   - **Description**: a suitable description. This is an optional field.
   - **Tags**: suitable tags to for easy identification. This is an optional field.

- **Map Type**: the map of the region that you want to use. This is a mandatory field.
- **Location Field**: the field that you want to use as the location. This is a mandatory field.
- **Data Field**: the field that you want to use as the data field. This is a mandatory field.
- **Show Data Value**: select this check box if you want to display the data value as marker on the visualization. This is an optional field.

5. Click **Create**.

The visualization is created and you can see the data populated in it.

7.1.9 Updating Visualizations

You can perform the following edit and delete operations on the visualizations:

Edit Visualization

To edit a visualization:

1. On the stage that has visualizations, click the **Visualizations** tab.
2. Identify the visualization that you want to edit and click the pencil icon next to the visualization name.
3. In the **Edit Visualization** dialog box that appears, make the changes you want. You can even change the Y Axis and X Axis selections. When you change the Y Axis and X Axis values, you will notice a difference in the visualization as the basis on which the graph is plotted has changed.

The following are the other updates you can make to the visualizations:

- **Maximize Visualizations**: You can open the visualization in a new window/tab using the **Maximize Visualizations** icon in the visualization canvas.
- **Change Orientation**: Based on the data that you have in the visualization or your requirement, you can change the orientation of the visualization. You can toggle between horizontal and vertical orientations by clicking the **Flip Chart Layout** icon in the visualization canvas.
- **Delete Visualization**: You can delete the visualization if you no longer need it in the pipeline. In the visualization canvas, click the **Delete** icon available beside the visualization name to delete the visualization from the pipeline. Be careful while you delete the visualization, as it is deleted with immediate effect and there is no way to restore it once deleted.
- **Delete All Visualizations**: You can delete all the visualizations in the stage if you no longer need them. In the visualization canvas, click the **Delete All** icon to delete all the visualizations of the stage at one go. Be careful while you delete the visualizations, as the effect is immediate and there is no way to restore the deleted visualizations.

7.2 Creating and Managing Dashboards

7.2.1 Adding a Dashboard

1. On the **Catalog** page, click **Create New Item** and select**Dashboard** from the drop-down list.
2. On the **Type Properties** screen, enter the following details:

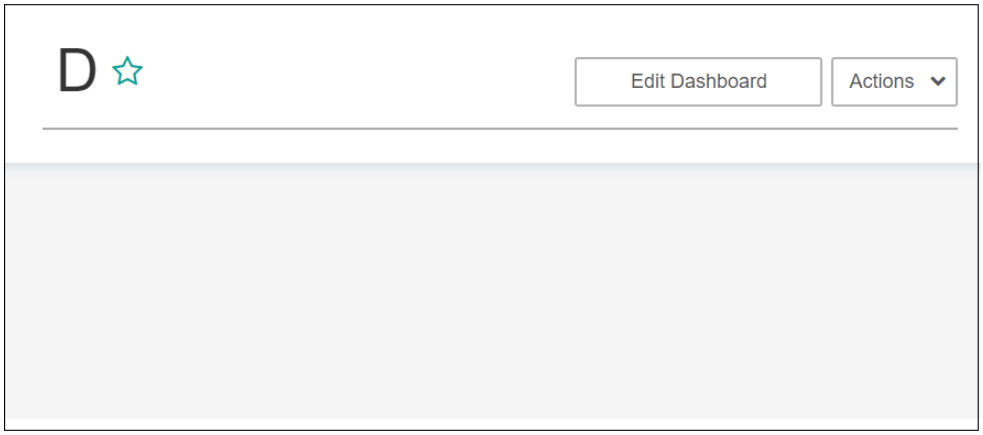
- **Name**
- **Description**
- **Tags**

3. Click **Next**.
4. On the **Source Details** page, enter the following details:
   - **Begin**: Select this option to build a dashboard from Beginning or Latest offset. Available options are **latest** and **earliest**. If you select the **latest** option, the dashboard shows only the new records that have been received after opening the dashboard. If you select the **earliest** option, then the records are read from the beginning of the OSA pipeline topics.
   - **CSS**: Enter a custom stylesheet for the dashboard.
5. Click **Save**.

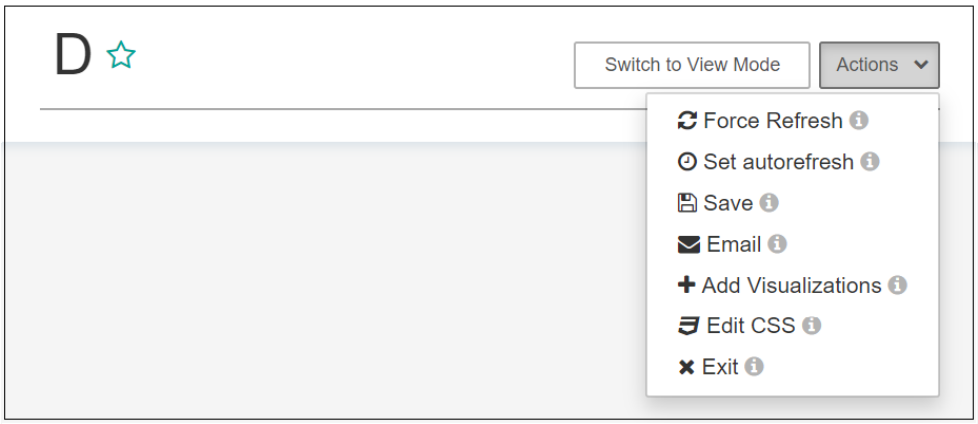
7.2.2 Editing a Dashboard

1. On the **Catalog** page click the dashboard that you want to edit.

   The dashboard opens in the View Mode.



2. Click **Edit Dashboard** to view the dashboard editing options under **Actions.**



3. Click **Force Refresh** to force refresh the dashboard.

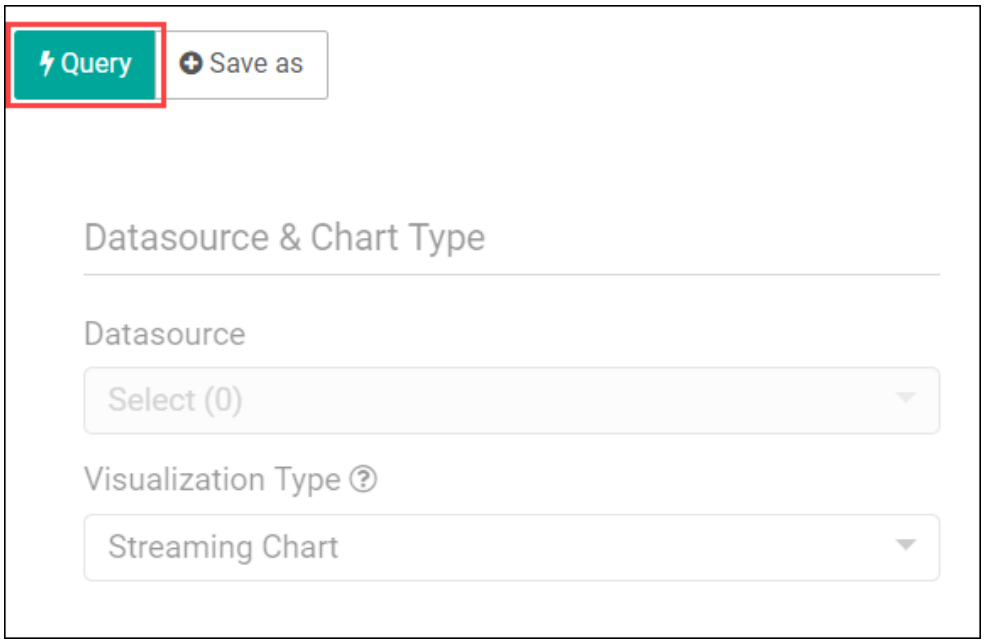
4. Click **Set autorefresh** to select the refresh frequency for the dashboard. This is applicable only for cube based visualizations not applicable for streaming charts created out of pipeline.

   This is just a client side setting and is not persisted.

   > **Note:**
   >
   > The Refresh options work only for druid-based visualizations, and not for streaming visualizations because they have continuous data flow.

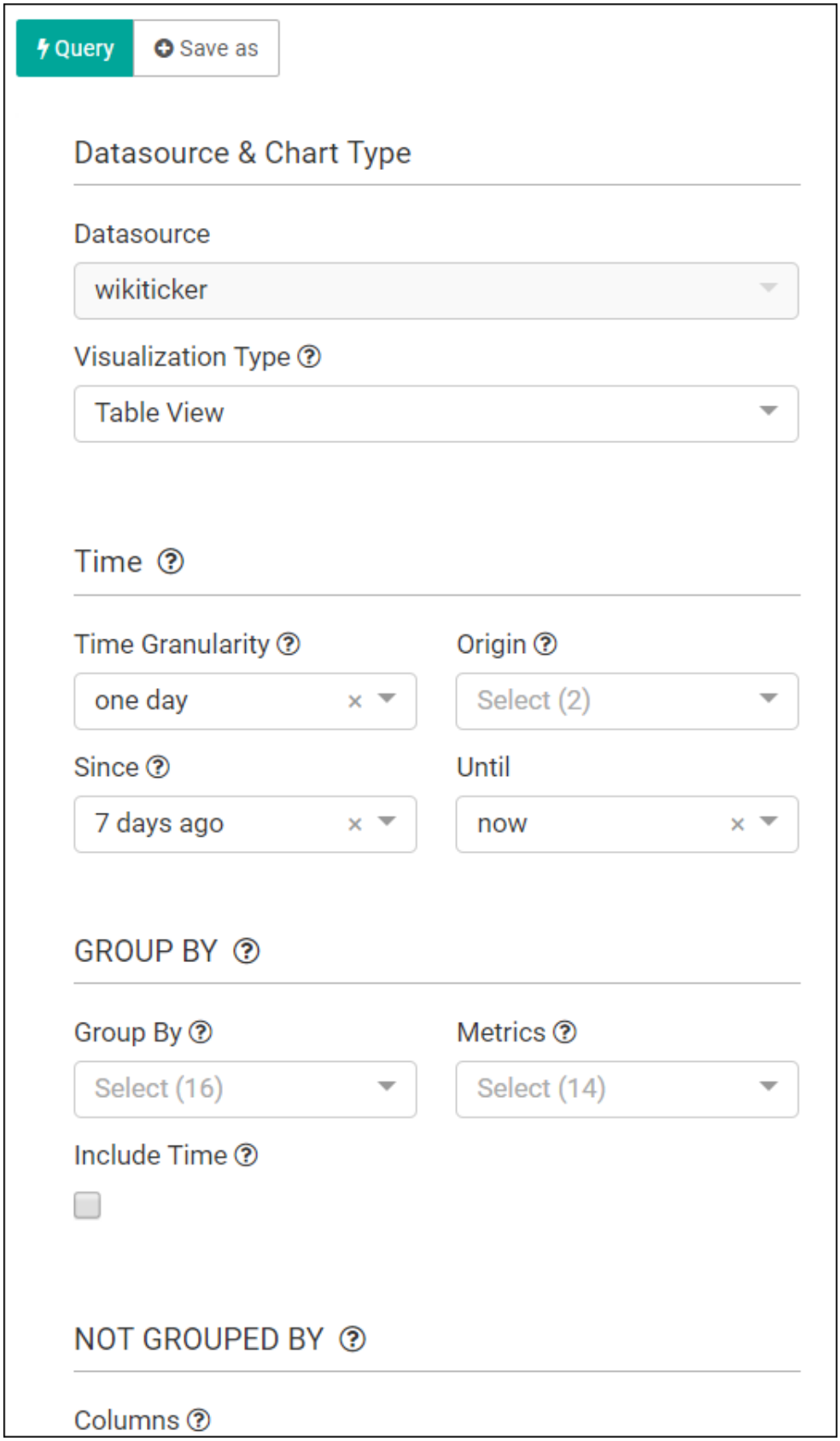
5. Click the **Save** icon to save the changes you have made to the dashboard.
6. Click the **Edit CSS** icon to edit and apply a CSS to the dashboard. You can also edit the CSS in the live editor.

   You can email the dashboard link to someone using the **Email the link** icon.
7. Click **Add visualizations** to see a list of existing visualizations. Visualizations from the pipelines and as well as from the cube explorations appear here. Go through the list, select one or more visualizations and add them to the dashboard.
8. Hover over the added visualization, click the **Explore chart** icon to open the chart editor of the visualization.



You can see the metadata of the visualization. You can also move the chart around the canvas, refresh it, or remove it from the dashboard.

A cube exploration looks like the following:



The various options like time granularity, group by, table timestamp format, row limit, filters, and result filters add more granularity and details to the dashboard.

9. Click **Save as**, to make the following changes to the dashboard:
    - Overwrite the visualization
    - Overwrite the current visualization with a different name
    - Add the visualization to an existing dashboard
    - Add the visualization to a new dashboard

7.2.3 Sharing a Dashboards with Peers

To share a dashboard:

1. On the **Catalog** page, click the dashboard you want to share.
2. Copy the URL of the dashboard from the browser's address bar. Share this URL with users who want to access the dashboard.

7.2.4 Deleting a Dashboard

To delete a dashboard:

1. Go to the **Catalog** page and hover the mouse over the dashboard that you want to delete.
2. Click the delete icon that appears to your right side on the screen.
3. On the **Delete Confirmation** screen, click **Delete**.

7.2.5 Importing a Dashboard with all its Dependencies

1. On the **Catalog** page, click **Import**.
2. In the **Import** dialog box, click **Select**, to locate and select the exported zip file on your computer.
3. Click **Import**.

The imported dashboard and its dependent artifacts are available on the **Catalog** page.

7.2.6 Exporting a Dashboard with all its Dependencies

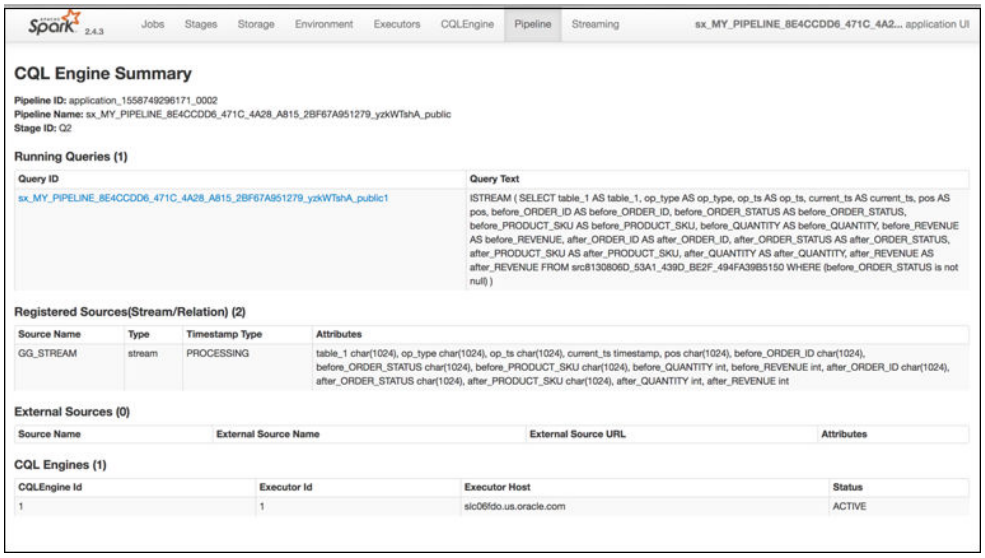
1. On the **Catalog** page, hover the mouse over, or select the dashboard that you want to export another GGSA instance.
2. Click the **Export** option that appears to your right side on the screen.
3. The selected dashboard and its dependent artifacts are exported as a JSON zip file, to your computer's default `Downloads` folder.

You can share, or import this zip file on the required user instances, to add the dashboard with all its dependencies.

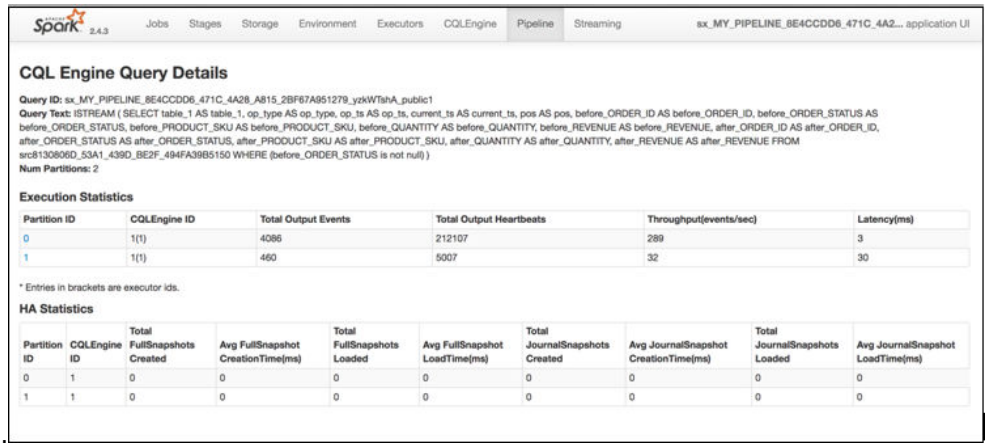
8
Monitor

8.1 Execution and HA Statistics

Click on a Query, in the CQL Engine Summary page.



You can view the Execution and HA statistics, in the CQL Engine Query details page that is displayed.



- Following are the details displayed on this page:
  - **Query ID:** System generated identifier for query
  - **Query Text:** Query String

- **Number of Partitions:** This fields shows the degree of parallelism of the query. The degree of parallelism is defined by total number of input partitions processed by a query.

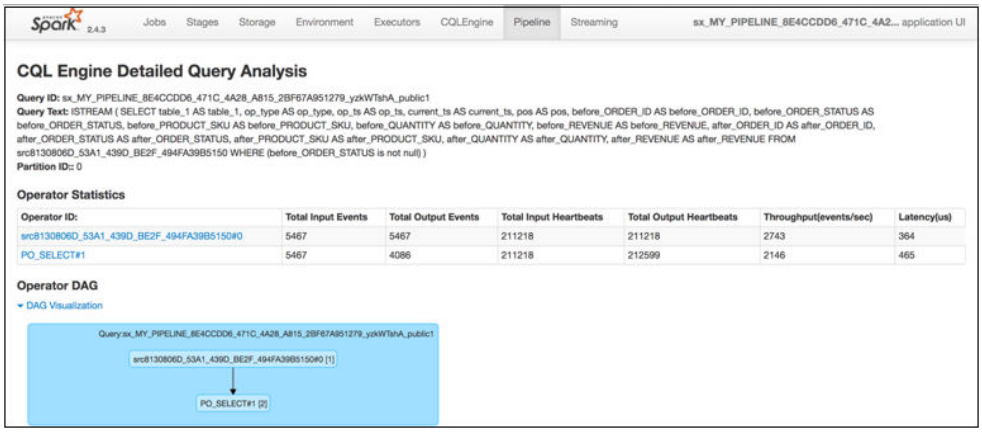
  Degree of parallelism depends on the many factors such as query constructs, number of input kafka partitions and number of executors assigned to application.

- **Execution Statistics Table:** This section shows the detailed execution statistics of each operator.
  - Partition ID: Partition Sequence Id
  - CQL Engine ID: Sequence ID of CQL Engine on which the partition is being processed.
  - Total Output Events: Number of output events emitted by CQL query for each partition.
  - Total Output Heartbeats: Number of heartbeat events emitted by CQL query for each partition. Please note that heartbeats are special events which ensures timestamp progression in Oracle Stream Analytics pipeline.
  - Throughput: Ratio of total number of events processed and total time spent in processing for each partition.
  - Latency: Average turnaround time taken to process a partition of stream.
- **HA Statistics:** This table shows the real-time statistics about query's HA operations. Note that unit of time is in MILLISECONDS.
  - Partition ID: Partition Sequence ID
  - CQL Engine ID: Sequence ID of CQL Engine on which the partition is being processed.
  - Total Full Snapshots Created: Total number of times the full state of query is serialized and saved.
  - Avg Full Snapshot Creation Time: Average time spent in serializing and saving the full state of query.
  - Total Full Snapshots Loaded: Total number of times the full state of query is de-serialized and loaded in query plan.
  - Avg Full Snapshot Load Time: Average time spent in de-serializing and loading the full state of query.
  - Total Journal Snapshots Created: Total number of times the journaled state of query is serialized and saved.
  - Avg Journal Snapshot Creation Time: Average time spent in serializing and saving the journaled state of query.
  - Total Journal Snapshots Loaded: Total number of times the journaled state of query is de-serialized and loaded in query plan.
  - Avg Journal Snapshot Load Time: Average time spent in de-serializing and loading the journaled state of query.

    Full Snapshot is the complete state of query. The query state represent the internal data structure and state of each operator in query plan. Journal snapshot is partial and incremental snapshot having a start time and end time. Oracle Stream Analytics optimizes the state preservation by using Journal snapshot if possible.

8.2 Detailed Query Analysis

Click on a specific partition in Execution Statistics table, the CQL Engine Detailed Query Analysis page is displayed.

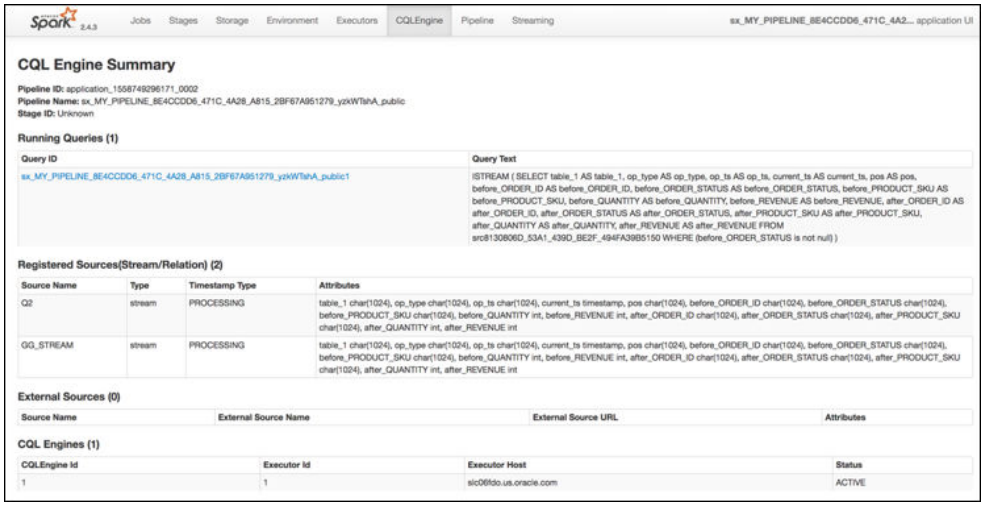


This page contains details about each execution operator of CQL query for a particular partition of a stage in pipeline.

- **Query ID**: System generated identifier for query
- **Query Text:** Query String
- **Partition ID:** All operator details are corresponding to this partition id.
- Operator Statistics:
  - Operator ID: System Generated Identifiers
  - Total Input Events: Total number of input events received by each operator.
  - Total Output Events: Total number of output events generated by each operator.
  - Total Input Heartbeats: Total number of heartbeat events received by each operator.
  - Total Output Heartbeats: Total number of heartbeat events generated by each operator.
  - Throughput(events/second): Ratio of total input events processed and total time spent in processing for each operator.
  - Latency(ms): Total turnaround time to process an event for each operator.
- **Operator DAG**: This is visual representation of the query plan. The DAG will show the parent-child details for each operator. You can further drill down the execution statistics of operator. Please click on the operator which will open **CQL Operator Details Page**.

8.3 Complete CQL Engine Statistics

Click on the CQL Engine tab, to view the complete CQL engine statistics for all stages and all queries in the Pipeline.



This page contains additional information about each execution operator, apart from the CQL Engine Query Details page, provides all essential metrics for each operator.

Pipeline ID: Unique pipeline id in Spark Cluster

Pipeline Name: Name of Oracle Stream Analytics Pipeline given by user in Oracle Stream Analytics UI.

Stage ID: Unique stage ID in DAG of stages for Oracle Stream Analytics Pipeline.

Running Queries: This section displays list of CQL queries running to compute the CQL transformation for a stage. This table displays a system-generated Query ID and Query Text. Check Oracle Continuous Query Language Reference for CQL Query syntax and semantics. To see more details about query, click on the query id hyperlink in the table entry to open **CQL Engine Query Details** page.

Registered Sources: This section displays internal CQL metadata about all the input sources which the query is based upon. For every input stream of the stage, there will be one entry in this table.

Each entry contains source name, source type, timestamp type and stream attributes. Timestamp type can be PROCESSING or EVENT timestamped. If stream is PROCESSING timestamped, then timestamp of each event will be defined by system. If stream is EVENT timestamped, then timestamp of each event is defined by one of the stream attribute itself. A source can be Stream or Relation.

External Sources: This section displays details about all external sources with which input stream is joined. The external source can be a database table or coherence cache.

CQL Engines: This section displays a table having details about all instances of CQL engines used by the pipeline. Here are details about each field of the table:

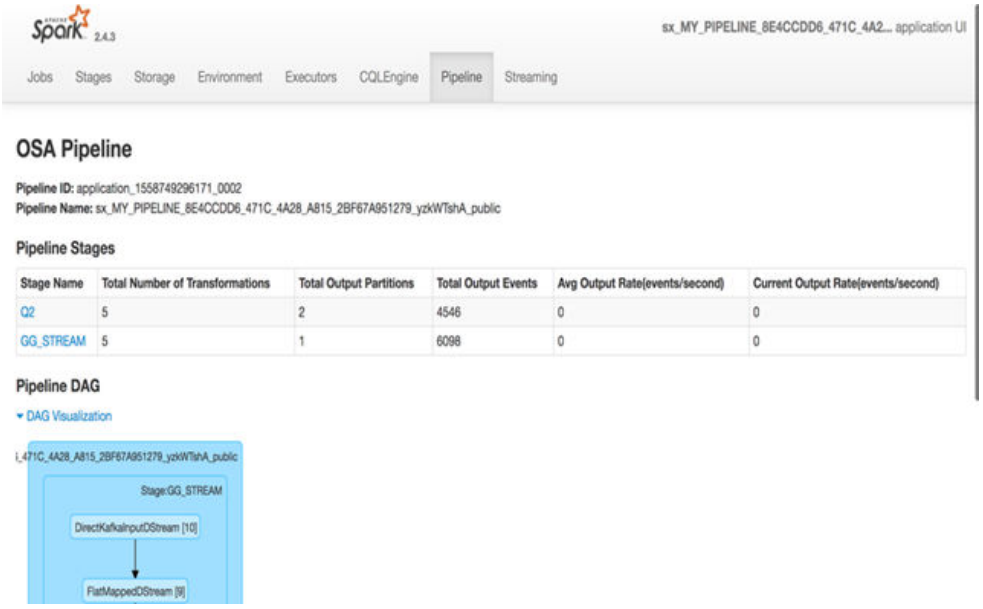
- CQLEngine Id: System generated id for a CQL engine instance.
- ExecutorId: Executor Id with which the CQL engine is associated.
- Executor Host: Address of the cluster node on which this CQL engine is running.

- Status: Current Status of CQL Engine. Status can be either ACTIVE or INACTIVE. If it is ACTIVE, it means that CQL Engine instance is up and running, Otherwise CQL Engine is stopped explicitly.

9
Reference

9.1 Pipeline Details

Click the **Pipeline** tab to view the Pipeline details page.



This page has following information about a pipeline:

- **Pipeline ID:** Unique pipeline id in Spark Cluster
- **Pipeline Name:** Name of Oracle Stream Analytics Pipeline given by user in Oracle Stream Analytics UI.
- **Pipeline Stages Table:** This section displays a table having detailed runtime metrics of each pipeline stage. Each entry in the table is corresponding to a pipeline stage in Oracle Stream Analytics UI pipeline graph.
- **Pipeline DAG:** This is a visual representation of all stages in form of a DAG where it displays the parent-child relation various pipeline stages. This diagram also shows the information about the transformations in each of the pipeline stage.
- The Pipeline Stages table contains the following columns.
- **Total Number of Transformations:** This measurement is number of spark transformations applied to compute each stage.

Oracle Stream Analytics supports various types of stages e.g. Query Stage, Pattern Stage, Custom Stage etc. For each pipeline stage, Oracle Stream Analytics defines a list of transformations which will be applied on the input stream for the stage. The output from final transformation will be the output of stage.

- **Total Output Partitions:** This measurement is total number of partitions in the output stream of each stage.

  Every pipeline stage has its own partitioning requirements which are determined from stage configurations. For example, If a QUERY stage defines a summary function and doesn't define a group-by column, then QUERY stage will have only one partition because there will be no partitioning criteria available.

- **Total Output Events:** This measurement is total number of output events (not micro-batches) emitted by each stage.

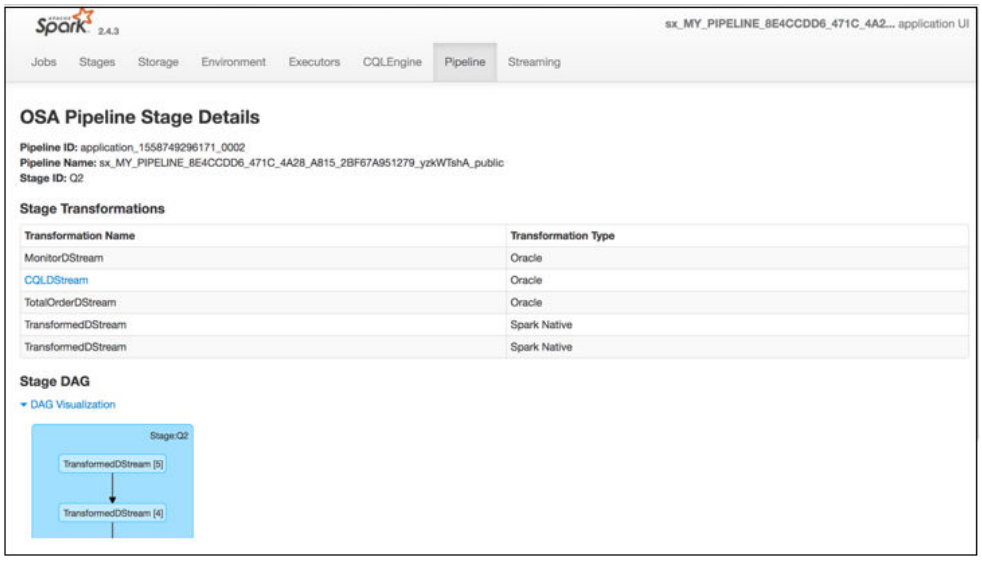
- **Average Output Rate:** This measurement is the rate at which each stage has emitted output events so far. The rate is a ratio of total number of output events so far and total application execution time.

  If the rate is ZERO, then it doesn't always mean that there is ERROR in stage processing. Sometime stage doesn't output any record at all (e.g. No event passed the Filter in Query stage).This can happen if rate of output events is less than 1 events/second.

- **Current Output Rate:** This measurement is the rate at which each stage is emitting output events. The rate is ratio of total number of output events and total application execution time since last metrics page refresh. To get better picture of current output rate, please refresh the page more frequently.

9.2 Stage Details

Click a Name in the **Name** column of the Pipeline Stages table to navigate to stage details page.



This page provides details about all the transformations in specific stage.

- **Pipeline ID:** Unique pipeline id in Spark Cluster
- **Pipeline Name:** Name of Oracle Stream Analytics Pipeline given by user in Oracle Stream Analytics UI.
- **Stage ID:** Unique stage id in DAG of stages for Oracle Stream Analytics Pipeline.

- **Stage DAG:** This is a visual representation of all transformations in form of a DAG where it displays the parent-child relation various pipeline transformations.

  Note Oracle Stream Analytics allows different transformation types including Business Rules but drilldowns are allowed only on CQLDStream transformations. Inside CQLDStream transformation, the input data is transformed using a continuously running query (CQL) in a CQL Engine. Note that there will be one CQL engine associated with one Executor.

- **Stage Transformations Table:**This table displays details about all transformations being performed in a specific stage. Each entry in the table is corresponding to a transformation operation used in computation of the stage. You can observe that final transformation in every stage is MonitorDStream. The reason is that MonitorDStream pipes output of stage to Oracle Stream Analytics UI Live Table.

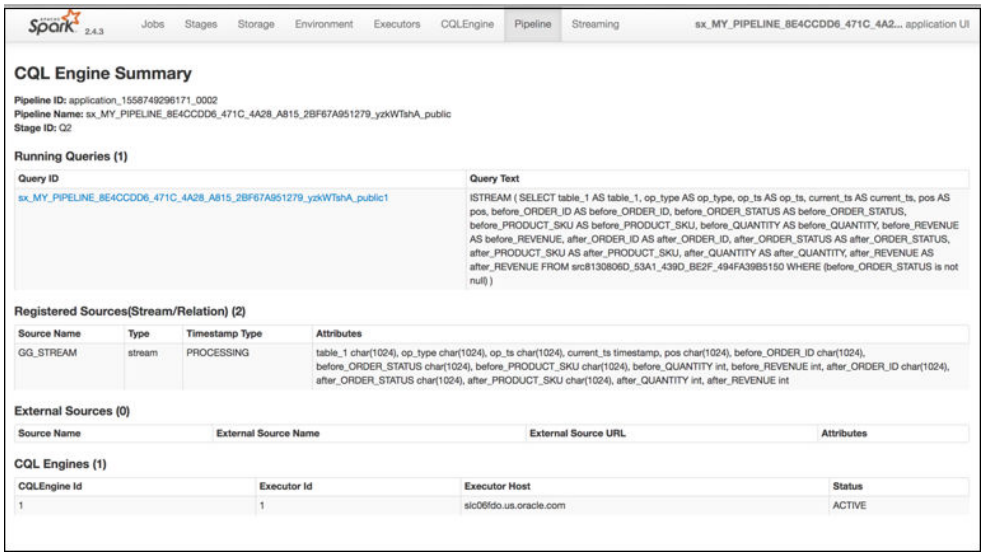
  – **Transformation Name:** Name of output DStream for the transformation. Every transformation in spark results into an output dstream.

  – **Transformation Type:** This is category information of each transformation being used in the stage execution. If transformation type is "Oracle", it is based on Oracle's proprietary transformation algorithm. If the transformation type is "Native", then transformation is provided by Apache Spark implementation.

CQLDStream transformation allows further drill down as described in the Query Details section.

9.3 Query Details

In the **Stage Transformation** table, click on a CQLDStream transformation to open **CQL Engine Summary** page for query-specific details.



The **CQL Engine Summary** page for the query has details like query text, stream sources feeding the query, external sources if any and all CQL engines with which the query is registered.

9.4 Internal Kafka Topics

The internal Kafka topics and Group ID's used by GGSA are standardized to the following naming conventions:

Kafka Topics

Topic	Resource	Operations
sx_backend_notification_<UUID>	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_messages_<UUID>	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_<stage_name>_public	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_<stage_name>_draft	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_public_<offset_number>_<stage_name>_offset	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE

Group IDs

Group ID	Resource	Operations
sx_<UUID>_receiver	Group	DESCRIBE,READ
sx_<UUID>	Group	DESCRIBE,READ
sx_<application_name>_public_<offset_number>_<stage_name>	Group	DESCRIBE,READ

10

Troubleshoot

10.1 Pipeline Debug and Monitoring Metrics

For every running Oracle Stream Analytics pipeline, there is a corresponding spark application deployed to the Spark Cluster. If Oracle Stream Analytics pipeline is deployed and running in draft mode or published mode, user can monitor and analyze the corresponding Spark application using real-time metrics provided by Oracle Stream Analytics. These metrics provide detailed run-time insights for every pipeline stage and user can drill down to operator level details. Note that these metrics are in addition to metrics provided by Spark. For each application, Oracle Stream Analytics provides detailed monitoring metrics which user can use to analyze if pipeline is working as per expectation.

The following topics explain how to access monitoring and debug metrics.

10.1.1 Spark Standalone

When running on Spark Standalone, the Spark Master URL is same as the host name in **Spark REST URL** field plus the **Spark standalone master console port** field. Spark Master URL is `http://<Spark REST URL>:< Spark standalone master console port>`. You can obtain Spark Master URL from the **System Settings** page.

The Spark Master page also displays a list of running applications. Click on an application to see details such as application ID, name, owner, current status.

For pipelines in 'Published' status, you can navigate to the metrics console from GGSA's catalog page as shown in the screenshot below.

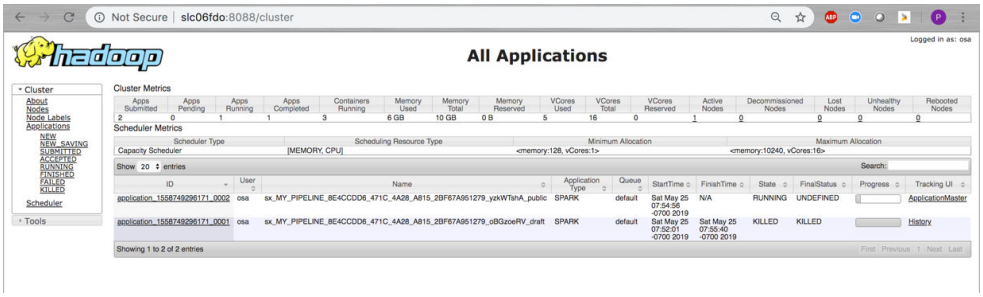
You can check the status of a published application on the catalog page, as shown in the screenshots below:

10.1.2 Spark on YARN

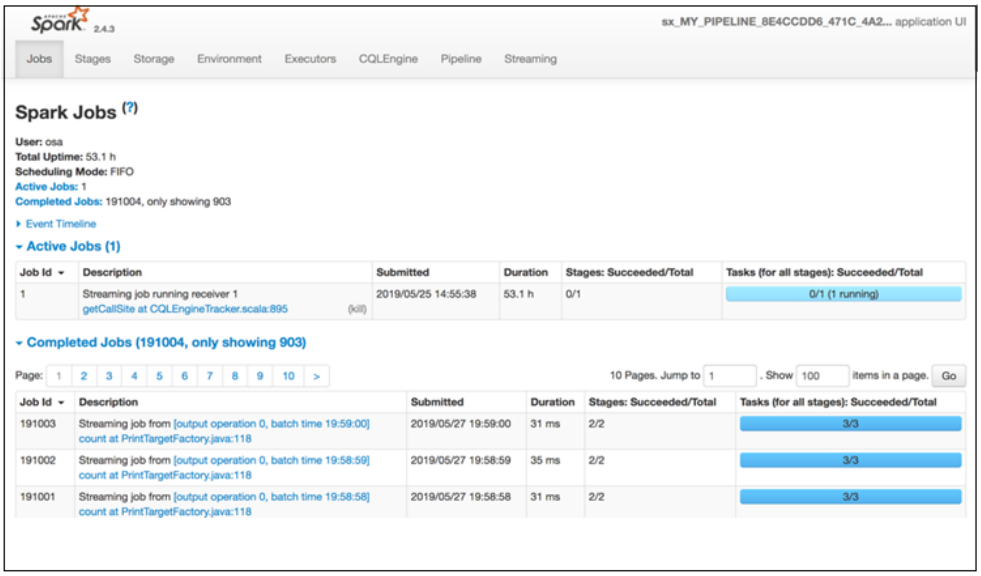
For published pipelines it is easier to navigate to Application Master console from GoldenGate Stream Analytics' catalog page.

1. For draft pipelines, you can navigate to YARN Applications page using the **YARN Resource Manager URL** and **YARN Master console port** values in **System Settings**.
2. Application master url is `http://<YARN Resource Manager URL>:<Yarn master Console Port>`.

   This page displays all the applications running in YARN:

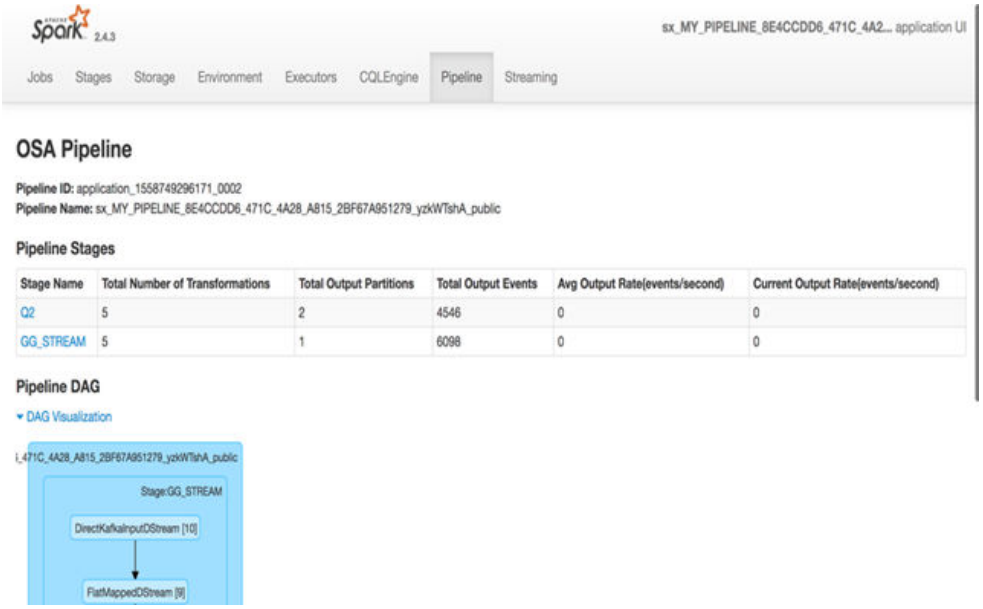


3. After identifying the application from the list, click the **ApplicationMaster** link, to open the Spark Application Details page:



10.1.3 Pipeline Details

Click the **Pipeline** tab to view the Pipeline details page.



This page has following information about a pipeline:

- **Pipeline ID:** Unique pipeline id in Spark Cluster
- **Pipeline Name:** Name of Oracle Stream Analytics Pipeline given by user in Oracle Stream Analytics UI.
- **Pipeline Stages Table:** This section displays a table having detailed runtime metrics of each pipeline stage. Each entry in the table is corresponding to a pipeline stage in Oracle Stream Analytics UI pipeline graph.
- **Pipeline DAG:** This is a visual representation of all stages in form of a DAG where it displays the parent-child relation various pipeline stages. This diagram also shows the information about the transformations in each of the pipeline stage.
- The Pipeline Stages table contains the following columns.
- **Total Number of Transformations:** This measurement is number of spark transformations applied to compute each stage.

  Oracle Stream Analytics supports various types of stages e.g. Query Stage, Pattern Stage, Custom Stage etc. For each pipeline stage, Oracle Stream Analytics defines a list of transformations which will be applied on the input stream for the stage. The output from final transformation will be the output of stage.
- **Total Output Partitions:** This measurement is total number of partitions in the output stream of each stage.

  Every pipeline stage has its own partitioning requirements which are determined from stage configurations. For example, If a QUERY stage defines a summary function and doesn't define a group-by column, then QUERY stage will have only one partition because there will be no partitioning criteria available.
- **Total Output Events:** This measurement is total number of output events (not micro-batches) emitted by each stage.

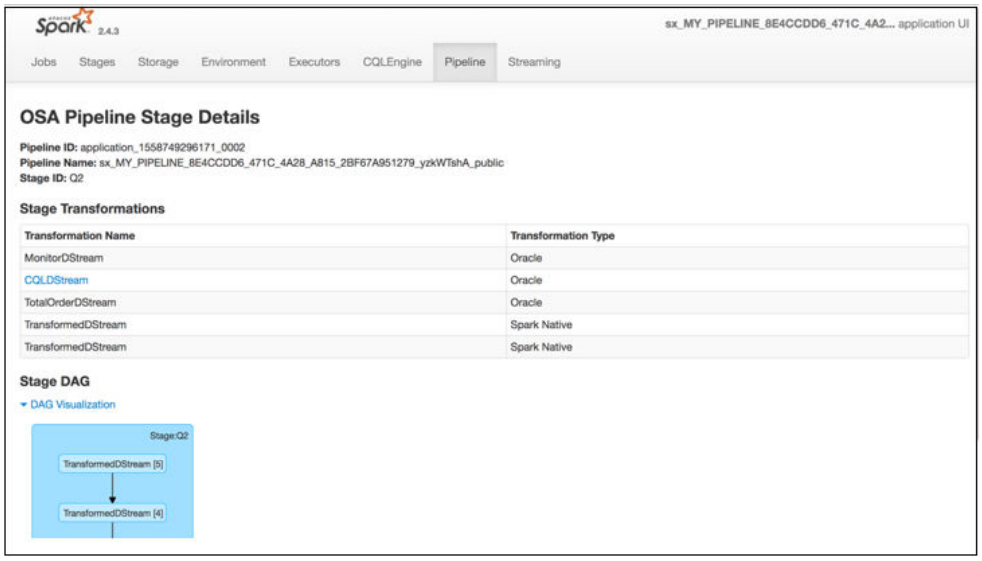
- **Average Output Rate:** This measurement is the rate at which each stage has emitted output events so far. The rate is a ratio of total number of output events so far and total application execution time.

  If the rate is ZERO, then it doesn't always mean that there is ERROR in stage processing. Sometime stage doesn't output any record at all (e.g. No event passed the Filter in Query stage).This can happen if rate of output events is less than 1 events/second.

- **Current Output Rate:** This measurement is the rate at which each stage is emitting output events. The rate is ratio of total number of output events and total application execution time since last metrics page refresh. To get better picture of current output rate, please refresh the page more frequently.

10.1.4 Stage Details

Click a Name in the **Name** column of the Pipeline Stages table to navigate to stage details page.



This page provides details about all the transformations in specific stage.

- **Pipeline ID:** Unique pipeline id in Spark Cluster
- **Pipeline Name:** Name of Oracle Stream Analytics Pipeline given by user in Oracle Stream Analytics UI.
- **Stage ID:** Unique stage id in DAG of stages for Oracle Stream Analytics Pipeline.
- **Stage DAG:** This is a visual representation of all transformations in form of a DAG where it displays the parent-child relation various pipeline transformations.

  Note Oracle Stream Analytics allows different transformation types including Business Rules but drilldowns are allowed only on CQLDStream transformations. Inside CQLDStream transformation, the input data is transformed using a continuously running query (CQL) in a CQL Engine. Note that there will be one CQL engine associated with one Executor.

- **Stage Transformations Table:**This table displays details about all transformations being performed in a specific stage. Each entry in the table is corresponding to a transformation operation used in computation of the stage. You can observe that final transformation in every stage is MonitorDStream. The reason is that MonitorDStream pipes output of stage to Oracle Stream Analytics UI Live Table.

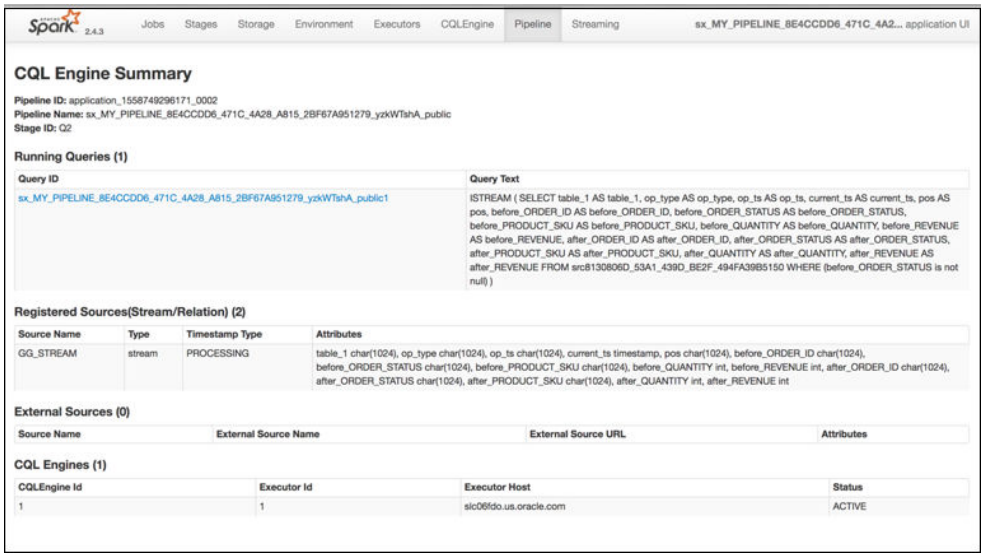
  – **Transformation Name:** Name of output DStream for the transformation. Every transformation in spark results into an output dstream.

- **Transformation Type:** This is category information of each transformation being used in the stage execution. If transformation type is "Oracle", it is based on Oracle's proprietary transformation algorithm. If the transformation type is "Native", then transformation is provided by Apache Spark implementation.

CQLDStream transformation allows further drill down as described in the Query Details section.

10.1.5 Query Details

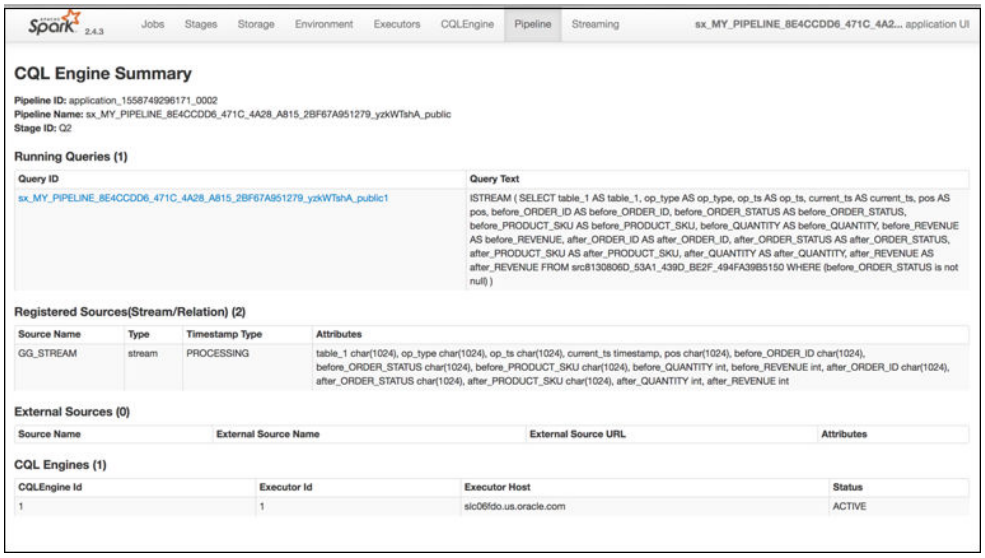
In the **Stage Transformation** table, click on a CQLDStream transformation to open **CQL Engine Summary** page for query-specific details.



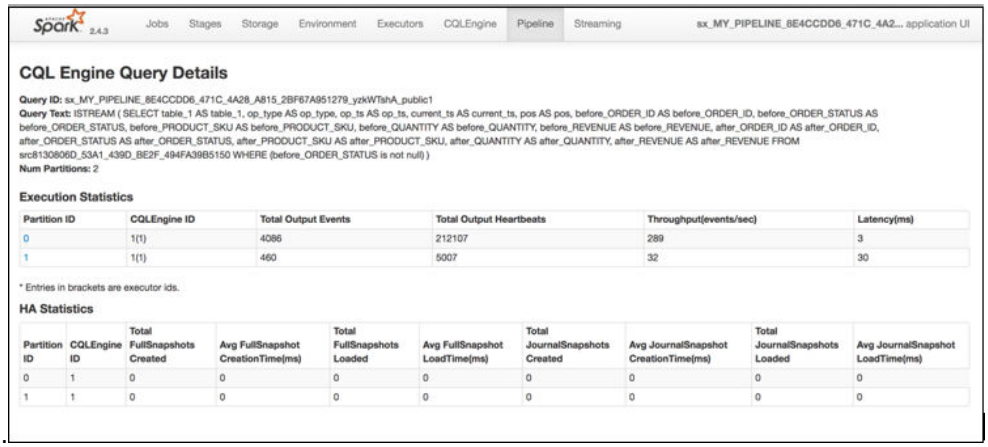
The **CQL Engine Summary** page for the query has details like query text, stream sources feeding the query, external sources if any and all CQL engines with which the query is registered.

10.1.6 Execution and HA Statistics

Click on a Query, in the CQL Engine Summary page.



You can view the Execution and HA statistics, in the CQL Engine Query details page that is displayed.



- Following are the details displayed on this page:
  - **Query ID:** System generated identifier for query
  - **Query Text:** Query String
  - **Number of Partitions:** This fields shows the degree of parallelism of the query. The degree of parallelism is defined by total number of input partitions processed by a query.

    Degree of parallelism depends on the many factors such as query constructs, number of input kafka partitions and number of executors assigned to application.
  - **Execution Statistics Table:** This section shows the detailed execution statistics of each operator.
    - Partition ID: Partition Sequence Id
    - CQL Engine ID: Sequence ID of CQL Engine on which the partition is being processed.

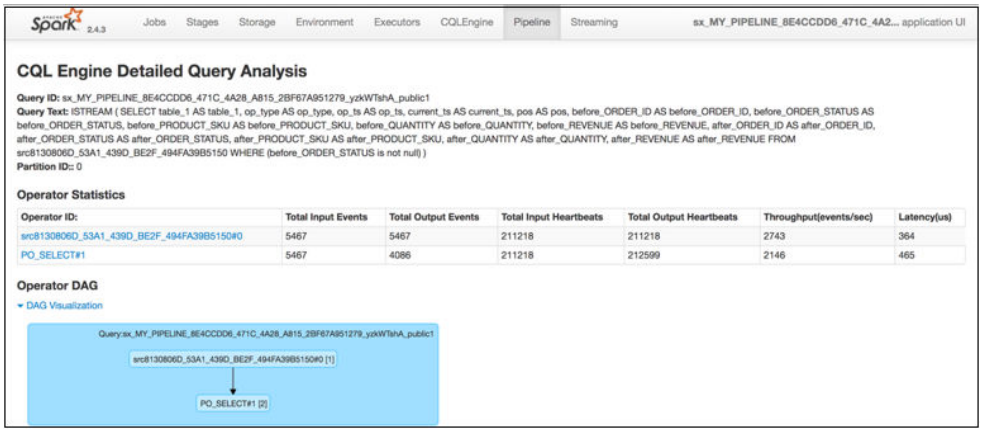
  - Total Output Events: Number of output events emitted by CQL query for each partition.
  - Total Output Heartbeats: Number of heartbeat events emitted by CQL query for each partition. Please note that heartbeats are special events which ensures timestamp progression in Oracle Stream Analytics pipeline.
  - Throughput: Ratio of total number of events processed and total time spent in processing for each partition.
  - Latency: Average turnaround time taken to process a partition of stream.

- **HA Statistics:** This table shows the real-time statistics about query's HA operations. Note that unit of time is in MILLISECONDS.
  - Partition ID: Partition Sequence ID
  - CQL Engine ID: Sequence ID of CQL Engine on which the partition is being processed.
  - Total Full Snapshots Created: Total number of times the full state of query is serialized and saved.
  - Avg Full Snapshot Creation Time: Average time spent in serializing and saving the full state of query.
  - Total Full Snapshots Loaded: Total number of times the full state of query is de-serialized and loaded in query plan.
  - Avg Full Snapshot Load Time: Average time spent in de-serializing and loading the full state of query.
  - Total Journal Snapshots Created: Total number of times the journaled state of query is serialized and saved.
  - Avg Journal Snapshot Creation Time: Average time spent in serializing and saving the journaled state of query.
  - Total Journal Snapshots Loaded: Total number of times the journaled state of query is de-serialized and loaded in query plan.
  - Avg Journal Snapshot Load Time: Average time spent in de-serializing and loading the journaled state of query.

    Full Snapshot is the complete state of query. The query state represent the internal data structure and state of each operator in query plan. Journal snapshot is partial and incremental snapshot having a start time and end time. Oracle Stream Analytics optimizes the state preservation by using Journal snapshot if possible.

10.1.7 Detailed Query Analysis

Click on a specific partition in Execution Statistics table, the CQL Engine Detailed Query Analysis page is displayed.

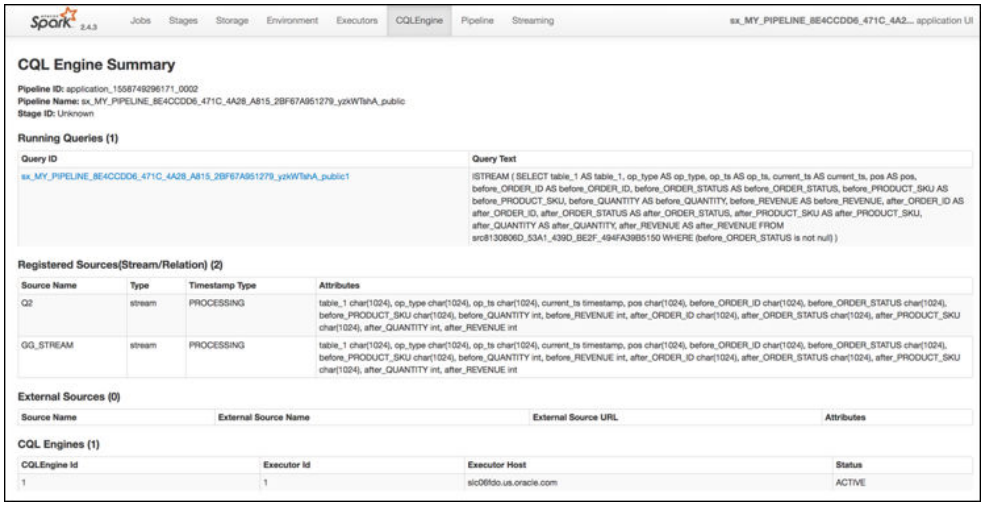


This page contains details about each execution operator of CQL query for a particular partition of a stage in pipeline.

- **Query ID**: System generated identifier for query
- **Query Text:** Query String
- **Partition ID:** All operator details are corresponding to this partition id.
- Operator Statistics:
  - Operator ID: System Generated Identifiers
  - Total Input Events: Total number of input events received by each operator.
  - Total Output Events: Total number of output events generated by each operator.
  - Total Input Heartbeats: Total number of heartbeat events received by each operator.
  - Total Output Heartbeats: Total number of heartbeat events generated by each operator.
  - Throughput(events/second): Ratio of total input events processed and total time spent in processing for each operator.
  - Latency(ms): Total turnaround time to process an event for each operator.
- **Operator DAG**: This is visual representation of the query plan. The DAG will show the parent-child details for each operator. You can further drill down the execution statistics of operator. Please click on the operator which will open **CQL Operator Details Page**.

10.1.8 Complete CQL Engine Statistics

Click on the CQL Engine tab, to view the complete CQL engine statistics for all stages and all queries in the Pipeline.



This page contains additional information about each execution operator, apart from the CQL Engine Query Details page, provides all essential metrics for each operator.

Pipeline ID: Unique pipeline id in Spark Cluster

Pipeline Name: Name of Oracle Stream Analytics Pipeline given by user in Oracle Stream Analytics UI.

Stage ID: Unique stage ID in DAG of stages for Oracle Stream Analytics Pipeline.

Running Queries: This section displays list of CQL queries running to compute the CQL transformation for a stage. This table displays a system-generated Query ID and Query Text. Check Oracle Continuous Query Language Reference for CQL Query syntax and semantics. To see more details about query, click on the query id hyperlink in the table entry to open **CQL Engine Query Details** page.

Registered Sources: This section displays internal CQL metadata about all the input sources which the query is based upon. For every input stream of the stage, there will be one entry in this table.

Each entry contains source name, source type, timestamp type and stream attributes. Timestamp type can be PROCESSING or EVENT timestamped. If stream is PROCESSING timestamped, then timestamp of each event will be defined by system. If stream is EVENT timestamped, then timestamp of each event is defined by one of the stream attribute itself. A source can be Stream or Relation.

External Sources: This section displays details about all external sources with which input stream is joined. The external source can be a database table or coherence cache.

CQL Engines: This section displays a table having details about all instances of CQL engines used by the pipeline. Here are details about each field of the table:

- CQLEngine Id: System generated id for a CQL engine instance.
- ExecutorId: Executor Id with which the CQL engine is associated.
- Executor Host: Address of the cluster node on which this CQL engine is running.

- Status: Current Status of CQL Engine. Status can be either ACTIVE or INACTIVE. If it is ACTIVE, it means that CQL Engine instance is up and running, Otherwise CQL Engine is stopped explicitly.

10.1.9 Internal Kafka Topics

The internal Kafka topics and Group ID's used by GGSA are standardized to the following naming conventions:

Kafka Topics

Topic	Resource	Operations
sx_backend_notification_<UUID>	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_messages_<UUID>	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_<stage_name>_public	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_<stage_name>_draft	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE
sx_<application_name>_public_<offset_number>_<stage_name>_offset	Topic	CREATE,DELETE,DESCRIBE,DESCRIBE_CONFIGS,READ,WRITE

Group IDs

Group ID	Resource	Operations
sx_<UUID>_receiver	Group	DESCRIBE,READ
sx_<UUID>	Group	DESCRIBE,READ
sx_<application_name>_public_<offset_number>_<stage_name>	Group	DESCRIBE,READ

10.2 Common Issues and Remedies

This section provides a comprehensive list of items to verify if pipelines are not running as expected.

10.2.1 Pipeline

Common issues encountered while deploying pipelines are listed in this section.

10.2.2 Pipeline

Common issues encountered while deploying pipelines are listed in this section.

10.2.2.1 Pipelines are not running as expected

If a pipeline is not running as expected, verify the following:

Ensure that Pipeline is Deployed Successfully

To verify Pipeline Deployment on Apache Spark Installation based Spark Cluster:

1. Open Spark Master console user interface.
2. If you see the status as `Running`, then the pipeline is currently deployed and running successfully.

Ensure that the Input Stream is Supplying Continuous Stream of Events to the Pipeline

To check for a continuous supply of events from the input stream:

1. Go to the **Catalog**.
2. Locate and click the stream you want to troubleshoot.
3. Check the value of the **topicName** property under the **Source Type Parameters** section.
4. Since this topic is created using Kafka APIs, you cannot consume this topic with REST APIs.

   Listen to the Kafka topic hosted on a standard Apache Kafka installation.

   You can listen to the Kafka topic using utilities from a Kafka Installation. `kafka-console-consumer.sh` is a utility script available as part of any Kafka installation.

   Follow these steps to listen to Kafka topic:

   a. Determine the Zookeeper Address from Apache Kafka Installation based Cluster.
   b. Use the following command to listen the Kafka topic:

   ```
   ./kafka-console-consumer.sh --zookeeper IPAddress:2181 --topicName
   ```

Ensure that the Output Stream is available in the Monitor Topic

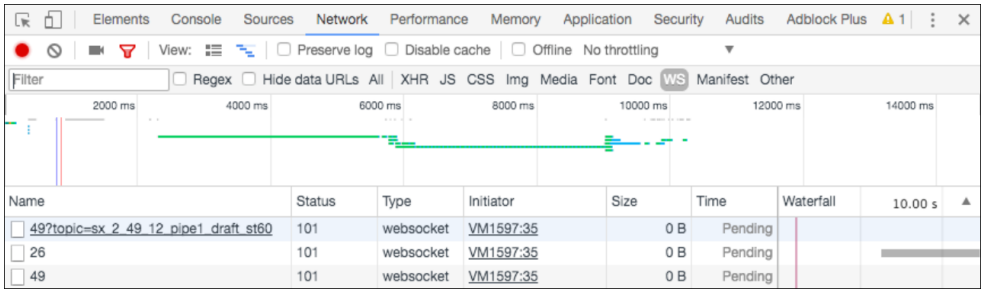
To check if the output stream is available in monitor topic:

1. Navigate to **Catalog**.
2. Open the required pipeline.
3. Ensure that you stay in pipeline editor and do not click **Done**. Otherwise the pipeline gets undeployed.
4. Right-click anywhere in the browser and select **Inspect**.
5. Go to WS tab under the **Network** tab.
6. Refresh the browser.

New websocket connections are created.

7. Locate a websocket whose URL has a parameter with the name `topic`.

   The value of topic `param` is the name of Kafka Topic where the output of this stage (query or pattern) is pushed.



   The topic name is `AppName_StageId`. The pipeline name can be derived from topic name by removing the `_StageID` from topic name. In the above snapshot, the pipeline name is `sx_2_49_12_pipe1_draft`.

Ensure that Caching is Working if a Pipeline is Correlating a Stream with Reference

1. Go to Spark application master UI.
2. Open Pipeline Summary Page after clicking on Pipeline Tab. The pipeline summary page shows a table of stages with various metrics.
3. Click on the stage id corresponding to the Query stage of the pipeline in which you're correlating stream with reference.
4. In the Pipeline Stage Details page, click the **CQLDStream** stage to open CQL Engine Summary page.
5. In the CQL Engine Summary page, locate the External Sources section. Note the source id for the reference which is used in stage.
6. Open the CQL Engine Detailed Query Analysis page.
7. Click the operator that has the operator id same as source id to open CQL Engine Detailed Operator Analysis page. Click the entry corresponding to the operator.
8. Look for the Cache Statistics section. If there is no such section, then caching is not enabled for the stage. If you see the non-zero entries in Cache Hits and Cache Misses, then caching is enabled and active for the pipeline stage.

10.2.2.2 GGSA Pipeline getting Terminated

A GGSA pipeline can terminate if the targets and references used the pipeline are unreachable or resources are unavailable. You can could see following exceptions on the logs:

```
com.tangosol.net.messaging.ConnectionException

SQLException in the Spark logs.
```

In case of a kafka source, republish the pipeline to read records from where it left off before terminating.

10.2.2.3 Live Table Shows `Listening Events` with No Events in the Table

There can be multiple reasons why status of pipeline has not changed to `Listening Events` from `Starting Pipeline`. Following are the steps to troubleshoot this scenario:

1. The live table shows output events of only the currently selected stage. At any time, only one stage is selected. Try switching to a different stage. If you observe output in live table for another stage, the problem can be associated with the stage. To debug further, go to step 5.

   If there is no output in any stage, then move to step 2.

2. Ensure that the pipeline is still running on Spark Cluster. See Ensure that the Pipeline is Deployed Successfully
3. If the Spark application for your pipeline is killed or aborted, then it suggests that the pipeline has crashed. To troubleshoot further, you may need to look into application logs.
4. If application is in ACCEPTED, NEW or SUBMITTED state, then application is waiting for cluster resource and not yet started. If there are not enough resources, check the number of VCORES in Big Data Cloud Service Spark Yarn cluster. For a pipeline, Stream Analytics requires minimum 3 VCORES.
5. If application is in RUNNING state, use the following steps to troubleshoot further:
   a. Ensure that the input stream is pushing events continuously to the pipeline.
   b. If the input stream is pushing events, ensure that each of the pipeline stages is processing events and providing outputs.
   c. If both of the above steps are verified successfully, then ensure that the pipeline is able to push the output events of each stage to its corresponding monitor topic:
      i. Determine the monitor topic for the stage, where the output of stage is being pushed into. See **Determine the Topic Name where Output of Pipeline Stage is Propagated** .
      ii. Listen to the monitor topic and ensure that the events are continuously being pushed in topic. To listen to the Kafka topic, you must have access to Kafka cluster where topic is created. You can listen to the Kafka topic using utilities from a Kafka Installation. `kafka-console-consumer.sh` is a utility script available as part of any Kafka installation.
      iii. If you don't see any events in the topic, then this can be an issue related to writing output events from stage to monitor topic. Check the server logs and look for any exception and then report to the administrator.
      iv. If you can see outputs events in monitor topic, then the issue can be related to reading output events in web browser.

Determine the Topic Name where Output of Pipeline Stage is Propagated

Here are the steps to find the topic name for a stage:

1. Open the Pipeline Summary Page for your pipeline. If you don't know the corresponding application name for this pipeline, see **Ensure that the Output Stream is available in the Monitor Topic** for instructions.
2. This page will provide the Pipeline Name and various stage ids. For every pipeline stage, you will see an entry in the table.
3. For every stage, the output topic id will be **PipelineName_StageID**.
4. Click **Done** in the pipeline editor and then go back to Catalog and open the pipeline again.

10.2.2.4 Live Table Still Shows `Starting Pipeline`

There can be multiple reasons why status of pipeline has not changed to `Listening Events` from `Starting Pipeline`. Following are the steps to troubleshoot this scenario:

1. Ensure that the pipeline has been successfully deployed to Spark Cluster. For more information, see **Ensure that Pipeline is Deployed Successfully** . Also ensure that the Spark cluster is not down and is available.
2. If the deployment failed, check the Jetty logs to see the exceptions related to the deployment failure and fix the issues.
3. If the deployment is successful, verify that OSA webtier has received the pipeline deployment from Spark.
4. Click **Done** in the pipeline editor and then go back to Catalog and open the pipeline again.

Ensure that a Pipeline Stage is Still Processing Events

To verify if a particular pipeline stage is still processing events:

1. Go to Spark application master UI.
2. Open Pipeline Summary Page after clicking on Pipeline Tab. The pipeline summary page shows a table of stages with various metrics.
3. Check if Total Output Events is a non-zero value. Refresh the page to see if the value increases or stays the same. If the value remains same and doesn't change for a long time, then drill down into stage details.

10.2.2.5 Time-out Exception in the Spark Logs when you Unpublish a Pipeline

In the Jetty log look for the following message:

```
OsaSparkMessageQueue:182 - received:
oracle.wlevs.strex.spark.client.spi.messaging.AcknowledgeMessage Undeployment
Ack: oracle.wlevs.strex.spark.client.spi.messaging.AcknowledgeMessage
```

During an application shutdown, a pipeline may take several minutes to unpublish completely.

So, if you do not see the above message, then you may need to increase the **osa.spark.undeploy.timeout** value accordingly.

Also in the High Availability mode, at the time of unpublishing a pipeline, the snapshot folder is deleted.

In HA mode, if you do not receive the above error message on time, and see the following error:

```
Undeployment couldn't be complete within 60000 the snapshot folder may not be
completely cleaned.
```

it only means that the processing will not be impacted, but some disk space will be occupied.

10.2.2.6 Piling up of Queued Batches in HA mode

If a GGSA pipeline is deployed in the High Availability mode in a Spark Standalone cluster, every time there is a target or reference failure, Spark spins off new drivers. In case these targets and references are not recoverable, it results in a loop of queued up batches.
To resolve this issue, you have to unpublish the application manually.

10.2.2.7 Null Record from Summary in Query Stage

When you publish a pipeline for the first time, with a summary in one of the Query stages, the first record is null on all columns. This causes the pipeline to fail, if it has targets where key is necessary.

To solve this issue, check if there is a summary stage added before a target stage, and add a query stage with a filter checking for null values for the cache keys.

10.2.3 Stream

Common issues encountered with streams are listed in this section.

10.2.3.1 Cannot See Any Kafka Topic or a Specific Topic in the List of Topics

Use the following steps to troubleshoot:

1. Go to Catalog and select the Kafka connection which you are using to create the stream.
2. Click Next to go to Connection Details tab.
3. Click Test Connection to verify that the connection is still active.
4. Ensure that topic or topics exist in Kafka cluster. You must have access to Kafka cluster where topic is created. You can list all the Kafka topics using utilities from a Kafka Installation. `kafka-console-consumer.sh` is a utility script available as part of any Kafka installation.
5. If you can't see any topic using above command, ensure that you create the topic.
6. If the test connection failed, and you see error message like `OSA-01266 Failed to connect to the ZooKeeper server`, then the Kafka cluster is not reachable. Ensure that Kafka cluster is up and running.

10.2.3.2 Input Kafka Topic is Sending Data but No Events Seen in Live Table

This can happen if the incoming events are not adhered to the expected shape for the Stream. To check if the events are dropped due to shape mismatch, use the following steps to troubleshoot:

1. Verify if lenient parameter under Source Type Properties for the Stream is selected. If it is FALSE, then the event may have been dropped due to shape mismatch. To confirm this, check the application logs for the running application.
2. If the property is set to TRUE, debug further:
   a. Make sure that Kafka Cluster is up and running. Spark cluster should be able to access Kafka cluster.
   b. If Kafka cluster is up and running, obtain the application logs for further debugging.

10.2.4 Connection

Common issues encountered with connections are listed in this section.

10.2.4.1 Database Connection Failure

To test a Database connection:

1. From the **Catalog** page, select the database connection that you want to test.
2. Click **Next**.
3. On the **Connection Details** tab, click **Test Connection**.
   - If the test is successful, it indicates that the connection is active.

- If the test fails, the following error messages are displayed:
  - `OSA-01260 Failed to connect to the database. IO Error: The Network Adapter could not establish the connection`: This error message indicates that the DB host is not reachable from GGSA design time.
  - `OSA-01260 Failed to connect to the database. ORA-01017: invalid username/password; logon denied`: This error message indicates that your login credentials are incorrect.

10.2.4.2 Druid Connection Failure

To test a Druid connection:

1. From the **Catalog** page, select the Druid connection that you want to test.
2. Click **Next**.
3. On the **Connection Details** tab, click **Test Connection**.
   - If the test is successful, it indicates that the connection is active.
   - If the test fails, the following error message is displayed:

     `OSA-01460 Failed to connect to the druid services at zooKeeper server`: This error indicates that the druid zookeeper is not reachable. Ensure that the druid services and zookeeper cluster are up and running.

10.2.4.3 Coherence Connection Failure

GGSA does not provide a **Test connection** option for a coherence cluster. Refer to Oracle Coherence documentation to find utilities and tools to test a coherence connection.

10.2.4.4 JNDI Connection Failure

To test a JNDI connection:

1. From the **Catalog** page, select the JNDI connection that you want to test.
2. Click **Next**.
3. On the **Connection Details** tab, click **Test Connection**.
   - If the test is successful, it indicates that the connection is active.
   - If the test fails, the following error messages are displayed:
     - `OSA-01707 Communication with server failed. Ensure that server is up and server url(s) are specified correctly`: This error indicates that either the server is down or server url(s) is incorrectly specified. Server url should be of the format `host1:port1,host2:port2`.
     - `OSA-01706 JNDI connection failed. User: weblogic, failed to be authenticated`: This error indicates that the login credentials are incorrect.

10.2.5 Target

Common issues encountered with targets are listed in this section.

10.2.5.1 Cannot see any Events in Targets

If the pipeline is in the draft mode, it cannot push events to targets. Only published pipelines can push events to targets.

10.2.6 Geofence

Common issues encountered with geofences are listed in this section.

10.2.6.1 Name and Description Fields are not displayed for the DB-based Geofences

If name and description fields are not displayed for database-based geofence, ensure to follow steps mentioned below:

1. Go to Catalog and click Edit for the required database-based geofence.
2. Click Edit for Source Type Properties and then Next.
3. Ensure that the mapping for Name and Description is defined in Shape section.
4. Once these mappings are defined, you can see the name and description for the geofence.

10.2.6.2 DB-based Geofence is not Working

To ensure that a database-based geofence is working:

1. Go to Catalog and open the required database-based geofence.
2. Ensure that the connection used in geofence is active by clicking test button in database connection wizard.
3. Ensure that table used in geofence is still valid and exists in DB.
4. Go to the geofence page and verify that the issue is resolved.

10.2.7 Cube

Common issues encountered with cubes are listed in this section.

10.2.7.1 Unable to Explore Cube which was Working Earlier

If you are unable to explore a cube which was working earlier, follow the steps mentioned below:

1. Check if the druid zookeeper or the associate services for indexer, broker, middle manager or overlord is down.
2. Click the Druid connection and navigate to next page.
3. Test the connection. This step will tell you if the services are down and need to be looked into.

10.2.7.2 Cube Displays "Datasource not Ready"

If you keep seeing “Datasource not Ready” message when you explore a cube, follow the steps mentioned below:

1. Go to the druid indexer logs. Generally, it is `http:/DRUID_HOST:3090/console.html`.

2. Look for entry in running tasks **index_kafka_<cube-name>_<somehash>**. If there is no entry in running tasks, look for the same entry in pending tasks or completed tasks.
3. If the entry lies in pending tasks, it means that workers are running out of capacity and datasource will get picked for indexing as soon as it's available.
4. In such cases, either wait OR increase the worker capacity and restart druid services OR kill some existing datasource indexing tasks (they will get started again after sometime).
5. If the entry lies in completed tasks with "FAILED" status, it means that indexing failed either due to incorrect ingestion spec or due to resource issue.
6. You can find the exact reason by clicking "log (all)" link and navigating to the exception.
7. If it is due to ingestion, try changing the timestamp format. (Druid fails to index, if the timestamp is not in JODA timeformat OR if the timeformat specified does not match with format of timestamp value).

10.2.8 Dashboard

Common issues encountered with dashboards are listed in this section.

10.2.8.1 Visualizations Appearing Earlier are No Longer Available in Dashboard

Use the following steps to troubleshoot:

1. For missing streaming visualization, it might be due to the following reasons:
   a. Corresponding pipeline/stage for the missing visualizations no longer exists
   b. The visualization itself is removed from catalog or the pipeline editor
2. For missing exploration visualization (created from cube), it might happen as cube or visualization might have been deleted already.

10.2.8.2 Dashboard Layout Reset after You Resized/moved the Visualizations

Use the following steps to troubleshoot:

1. This might happen, if you forget to save the dashboard after movement/ resizing of visualizations.
2. Make sure to click Save after changing the layout.

10.2.8.3 Streaming Visualizations Do not Show Any Data

Use the following steps to troubleshoot:

1. Go to visualization in pipeline editor and make sure that live output table is displaying data.
2. If there is no output, ensure that the pipeline is deployed and running on cluster. Once you have the live output table displaying data, it shows up on the streaming visualization.

10.2.9 Live Output

Common issues encountered with live output are listed in this section.

10.2.9.1 Issues with Live Output

For every pipeline, there will be one Spark streaming pipeline running on Spark Cluster. If a Stream Analytics pipeline uses one or more Query Stage or Pattern Stage, then the pipeline will run one or more continuous query for each of these stages.

For more information about continuous query, see Understanding Oracle CQL.

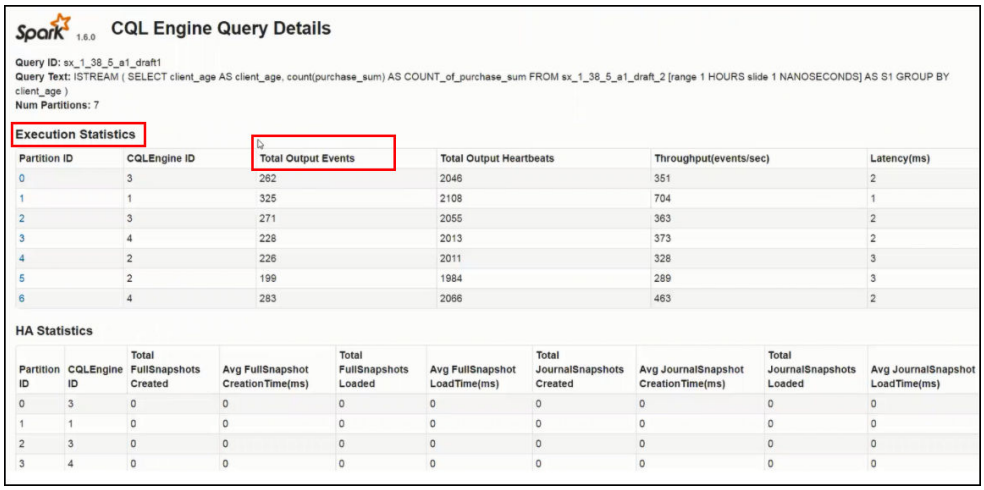
If there are no output events in Live Output Table for Query Stage or Pattern Stage, use the following steps to determine or narrow down the problem:

- Ensure that the Pipeline is Deployed Successfully
- Ensure that the input Stream is Supplying Continuous Stream of Events to the Pipeline
- Ensure that CQL Queries for each query stage emit output
- Ensure that the output of stage is available

Ensure that CQL Queries for Each Query Stage Emit Output

To check if the CQL queries are emitting output events to monitor CQL Queries using CQL Engine Metrics:

1. Open CQL Engine Query Details page. For more information, see Access CQL Engine Metrics.
2. Check that at least one partition has **Total Output Events** greater than zero under the **Execution Statistics** section.



   If your query is running without any error and input data is continuously coming, then the **Total Output Events** will keep rising.

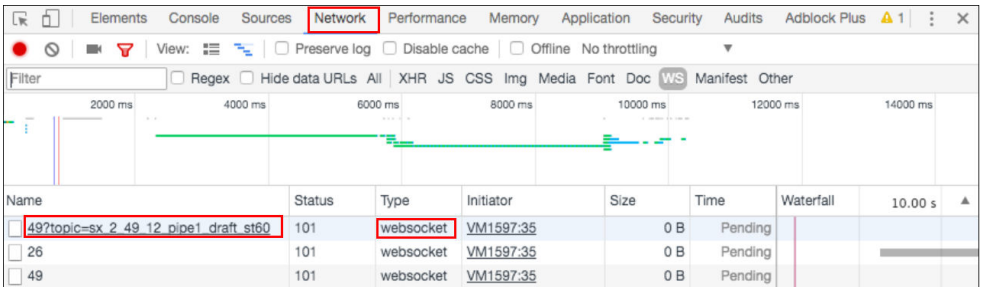
Ensure that the Output of Stage is Available

1. Ensure that you stay in the Pipeline Editor and do not click **Done**. Else, the pipeline gets undeployed.
2. Right-click anywhere in the browser and click **Inspect**.
3. Select **Network** from the top tab and then select **WS**.
4. Refresh the browser.

   New websocket connections are created.

5. Locate a websocket whose URL has a parameter with name `topic`.

   The value of the topic param is the name of the Kafka topic where the output of this stage is pushed.



6. Listen to the Kafka topic where output of the stage is being pushed.

   Since this topic is created using Kafka APIs, you cannot consume this topic with REST APIs. Follow these steps to listen to the Kafka topic:

   a. Listen to the Kafka topic hosted on a standard Apache Kafka installation.

      You can listen to the Kafka topic using utilities from a Kafka Installation. `kafka-console-consumer.sh` is a utility script available as part of any Kafka installation.

      To listen to Kafka topic:

      i. Determine the Zookeeper Address from Apache Kafka Installation based Cluster.

      ii. Use following command to listen the Kafka topic:

```
./kafka-console-consumer.sh --zookeeper IPAddress:2181 --topic
sx_2_49_12_pipe1_draft_st60
```

10.2.9.2 Missing Events due to Faulty Data

If the CQLEngine encounters faulty data in user defined functions, the exceptions of the events with faulty data are logged in executor logs, and the processing continues uninterrupted.

Sample logs of the dropped events and exceptions:

```
20/04/02 14:41:42 ERROR spark: Fault in CQL query processing.
Detailed Fault Information [Exception=user defined
function(oracle.cep.extensibility.functions.builtin.math.Sqrt@74a5e306)
runtime error while execution,
 Service-Name=SparkCQLProcessor, Context=phyOpt:1;
queries:sx_SquareRootPipeline_osaadmin_draft1


20/04/02 14:41:42 ERROR spark: Continue on exception by dropping faulty event.


20/04/02 14:41:42 ERROR spark: Dropped event details
<TupleValue><ObjectName>sx_SquareRootPipeline_osaadmin_draft_1</
ObjectName><Timestamp>1585838502000000000</Timestamp>
<TupleKind>PLUS</TupleKind><IntAttribute name="squareNumber"><Value>-2</
Value></IntAttribute><IsTotalOrderGuarantee>true</IsTotalOrderGuarantee></
TupleValue>:
```

10.2.10 Pipeline Deployment Failure

Sometimes pipeline deployment fails with the following exception:

```
Spark pipeline did not start successfully after 60000 ms.
```

This exception usually occurs when you do not have free resources on your cluster.

Workaround:

Use external Spark cluster or get better machine and configure the cluster with more resources.