Oracle® Fusion Middleware

Using Oracle Stream Analytics

E93286-03
April 2019

Oracle Fusion Middleware Using Oracle Stream Analytics,

E93286-03

Primary Author: Oracle Corporation

Contents

Preface

Using Oracle Stream Analytics describes how to get started with the product, how to build a simple pipeline, and how to build pipelines for specific use cases.

Topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

This document is intended for developers and administrators of Oracle Stream Analytics.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

Documentation for Oracle Stream Analytics is available on Oracle Help Center.

Also see the following documents for reference:

- *Understanding Oracle Stream Analytics*
- *Developing Custom Jars and Custom Stages in Oracle Stream Analytics*
- *Quick Installer for Oracle Stream Analytics*
- *Known Issues in Oracle Stream Analytics*
- *Spark Extensibility for CQL in Oracle Stream Analytics*

Conventions

The following text conventions are used in this document:

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

1
Administering Oracle Stream Analytics

Administering Oracle Stream Analytics is essential to get the required results.

The tasks listed in this chapter can only be performed by a user who has the administrator privileges.

Topics:

- Managing Users in Oracle Stream Analytics
- Configuring Oracle Stream Analytics System Settings
- Configuring User Preferences

Managing Users in Oracle Stream Analytics

After you install Oracle Stream Analytics, it is important to setup and manage users who can use the application.

In this release of Oracle Stream Analytics, user details are stored in a database. When you create Oracle Stream Analytics 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.

Adding Users

Though you can continue using Oracle 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 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 and click **Create** Make sure that you provide the same value in both **Password** and **Confirm Password** fields.

   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-18.1.0.0.1`.

   This is top-level folder in the folder where you have extracted your zip installer.
2. Execute the following command:

   ```
   java -cp ./lib/jetty-util-9.4.7.v20170914.jar
   org.eclipse.jetty.util.security.Password NewUser <password>
   ```

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

   You can see a result something similar to the following 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-18.1.0.0.0/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 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 Stream Analytics as `NewUser` using `<password>`. Repeat these steps to create as many users as you require.

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`.

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 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;
   ```

Configuring Oracle Stream Analytics System Settings

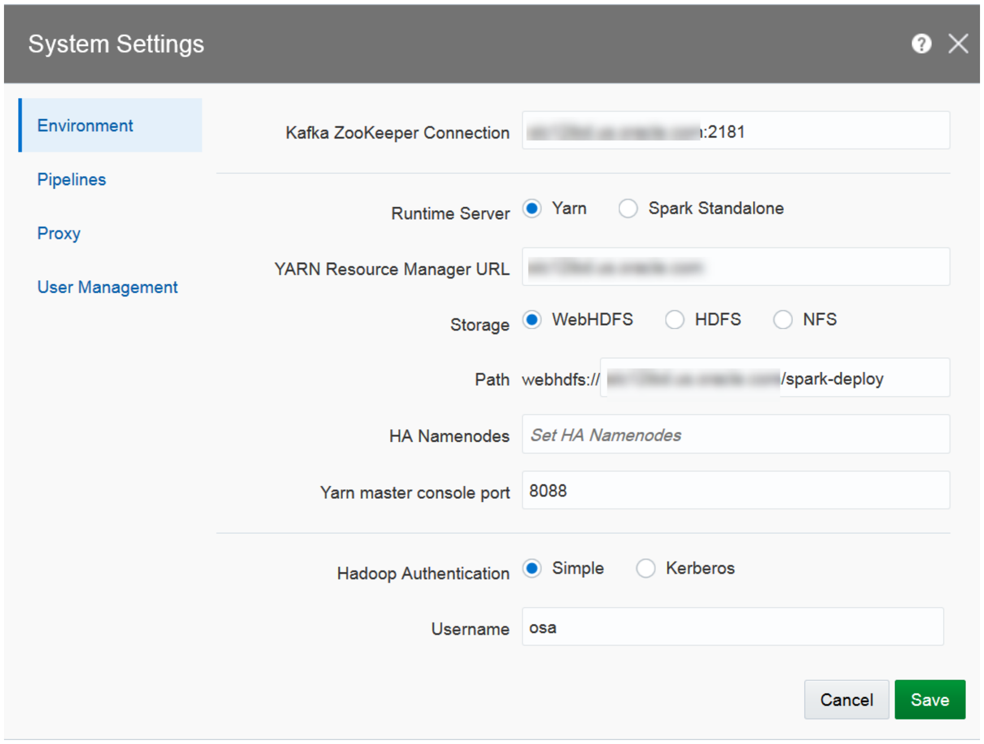
This topic applies only to Oracle user-managed services.

Only users with the *Administrator* role can set the system settings in Oracle Stream Analytics.

To set/update system settings:

1. Click the user name in the top right corner of the screen.
2. Click **System Settings**.

   The System Settings page opens.
3. Click **Environment**.
4. Specify the server names and URLs where the Kafka Zookeeper, Yarn Resource Manager, or Spark Standalone, Path, HA Namenodes are deployed and running. Hadoop authentication is an optional setting.



Kafka ZooKeeper Connection	[redacted]:2181
Runtime Server	[✓ Yarn] Spark Standalone
YARN Resource Manager URL	[redacted]
Storage	[✓ WebHDFS] HDFS NFS
Path	webhdfs:// [redacted]/spark-deploy
HA Namenodes	*Set HA Namenodes*
Yarn master console port	8088
Hadoop Authentication	[✓ Simple] Kerberos
Username	osa

- **Kafka Zookeeper Connection** — the URL where the Zookeeper server or servers are configured, separated by comma. Kakfa is used as internal transport.
- **Runtime Server** — the runtime server you want your Oracle Stream Analytics instance to run on
- **YARN Resource Manager URL** — the URL where the YARN Resource Manager is configured, if the runtime server is Yarn
- **Spark REST URL** — Spark standalone REST URL, if the runtime server is Spark standalone. To submit an Oracle Stream Analytics pipeline to Spark, the

pipeline needs to be copied to a storage location that is accessible by all Spark nodes.

- **Storage** — the type of storage for pipelines
- **Path** — the path where the storage exists. The user who accesses this folder must have the write permission to it. The folder at this path is either created by the administrator, or Oracle Stream Analytics creates the folder at runtime.
- **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>` etc.
- **Hadoop Authentication** — the type of authentication configured on the Hadoop cluster. Oracle Stream Analytics supports only simple authentication.

5. Click **Pipelines**. Specify the various settings for the pipeline within Oracle Stream Analytics.
   - **Batch Duration** — the default duration of the batch for each pipeline
   - **Executor Count** — the default number of executors per pipeline
   - **Cores per Executor** — the default number of cores. A minimum value of 2 is required.
   - **Executor Memory** — the default allocated memory for each executor instance in megabytes
   - **Cores per Driver** — the default number of cores
   - **Driver Memory** — the default allocated memory per driver instance in megabytes
   - **High Availability** — toggle the default HA value as on/off for each pipeline
6. Click **Proxy**. If you set proper proxy, the back-end system will use these settings to reach the REST target. The proxy settings are also required for geo code related spatial patterns.
7. Click **User Management**. You see the list of available users here. You can add/ delete users from this screen.
8. Click **Save**.

Note:

On the Yarn Cluster, make sure that you do not set a high value for `yarn.nm.liveness-monitor.expiry-interval-ms` (for example, 3000 ms instead of 10 minutes). This property determines the default value for how long to wait until a node manager is considered dead. Using a high value for this property does not process the events and the events are lost.

Configuring User Preferences

This topic applies only to Oracle user-managed services.

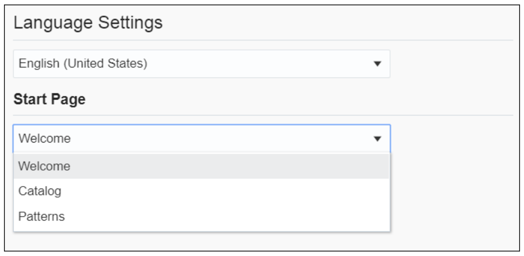
Based on the preferences the users set in this page, the characteristics of Oracle Stream Analytics vary.

To set/update user preferences:

1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.

General

Provides a set of general preferences that you can view and set according to your requirements.



Language Settings

English (United States)

Start Page

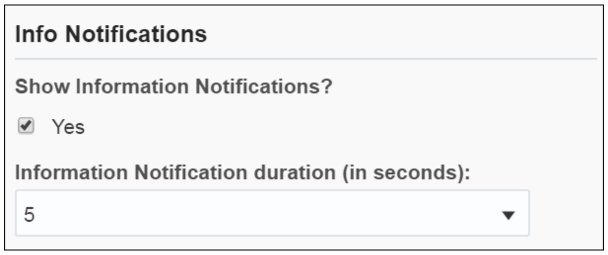
Welcome

Start Page

Select if you want the Home page, the **Catalog** page, or the **Patterns** page to appear as the **Start** Page.

Notifications

Provides a set of notifications preferences that you can view and set according to your requirements.



Info Notifications

Show Information Notifications?

✓ Yes

Information Notification duration (in seconds):

5

Show Information Notifications

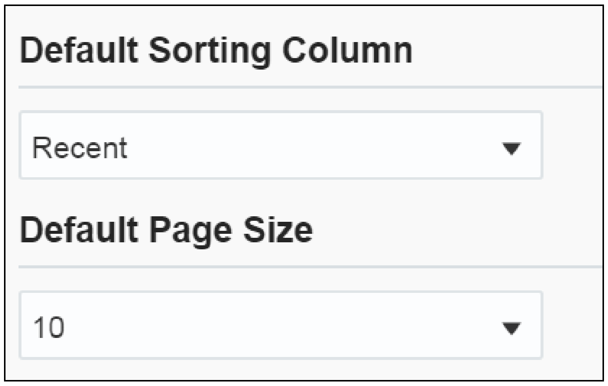
Select this option if you want the information notifications to appear in the pipeline. This option is selected by default.

Information Notification duration (in seconds)

Choose the number of seconds for which the notifications appear. The default value is 5.

Catalog

Provides a set of catalog preferences that you can view and set according to your requirements.



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 Sorting Order

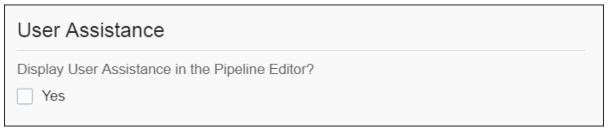
Select the order by which you want the columns to be sorted. This value will be used as the default value for all columns until you change the value again.

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.

Pipeline

Provides a set of pipeline preferences that you can view and set according to your requirements.



Select **Yes** if you want to display the User Assistance text for the pipelines in the Pipeline Editor.

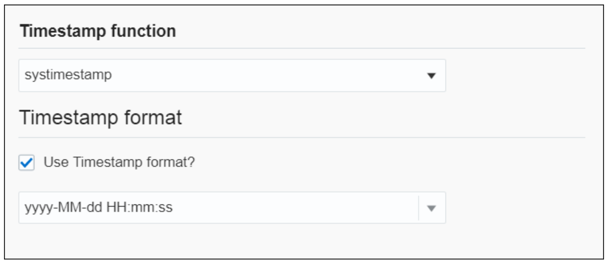
Live Output Stream

Provides a set of pipeline live output stream preferences that you can view and set according to your requirements.

Select a value that you want to be applied as the default table size for the data in Live Output Stream of a pipeline.

Timestamp

Provides a set of pipeline timestamp preferences that you can view and set according to your requirements.



Timestamp function

systimestamp

Timestamp format

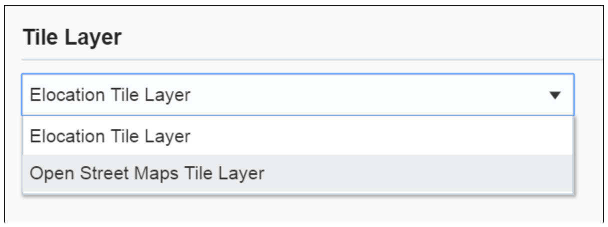
[✓] Use Timestamp format?

yyyy-MM-dd HH:mm:ss

Map

Provides a set of map preferences that you can view and set according to your requirements.

Select a value that you want to be used as the default tile layer preference in the geo fences.



Tile Layer

Elocation Tile Layer

2

Working with Oracle Stream Analytics

This topic applies only to Oracle user-managed services.

Oracle Stream Analytics has various artifacts like connections, references, streams, targets, cubes, dashboards, predictive models, custom jars, and many more. The artifacts are the important resources using which you can create pipelines.

Topics:

Home Page

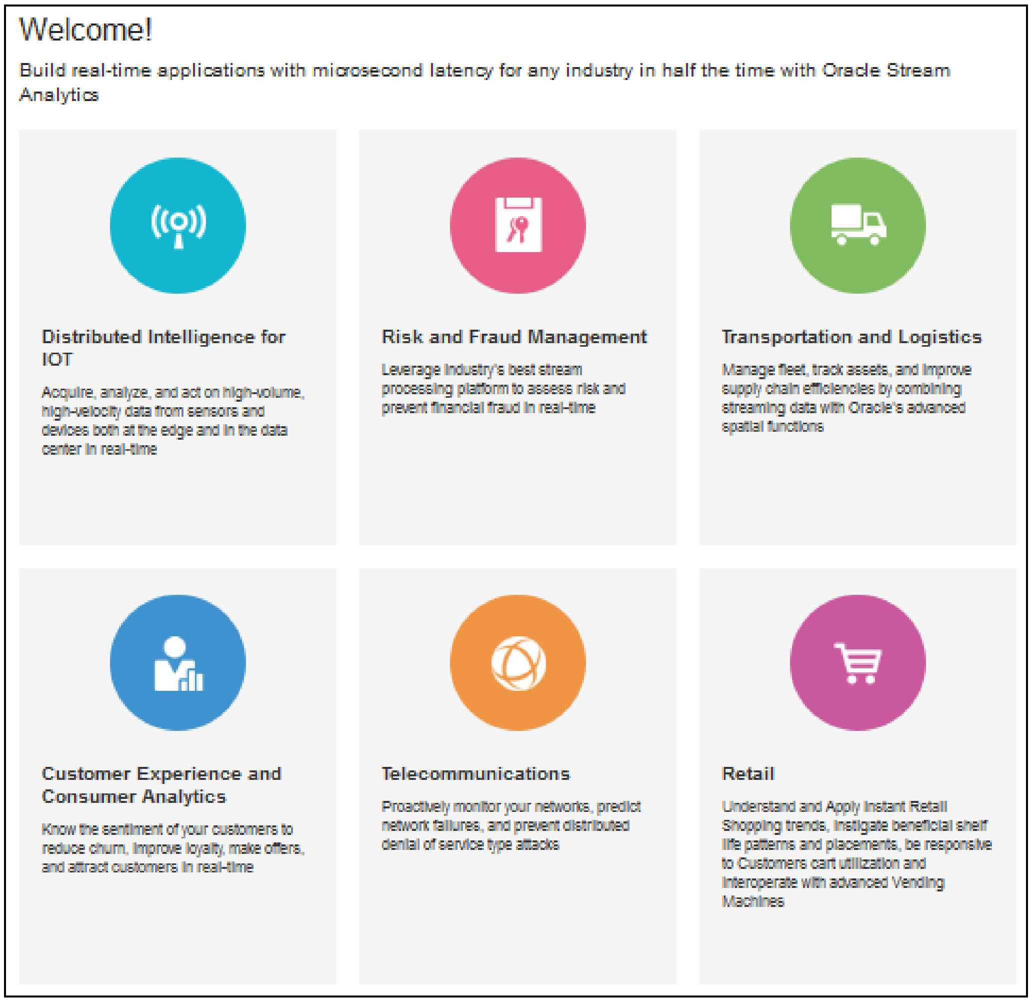
The Home page is the first page that you see when you login to Oracle Stream Analytics. This page lists the industry verticals that Oracle Stream Analytics supports.

Each industry vertical has a tag associated with it and the tags are case-sensitive.

- *Distributed Intelligence for IOT* - Acquire, analyze, and act on high-volume, high-velocity data from sensors and devices both at the edge and in the data center in real-time. Tag for this vertical is *IOT*.

- *Risk and Fraud Management* - Leverage industry's best stream processing platform to assess risk and prevent financial fraud in real-time. Tag for this vertical is *risk*.
- *Transportation and Logistics* - Manage fleet, track assets, and improve supply chain efficiencies by combining streaming data with Oracle's advanced spatial functions. Tag for this vertical is *transportation*.
- *Customer Experience and Consumer Analytics* - Know the sentiment of your customers to reduce churn, improve loyalty, make offers, and attract customers in real-time. Tag for this vertical is *customer*.
- *Telecommunications* - Pro actively monitor your networks, predict network failures, and prevent distributed denial of service type attacks. Tag for this vertical is *telecom*.
- *Retail* — Understand and Apply instant Retail Shopping trends, instigate beneficial shelf life patterns and placements, be responsive to Customers cart utilization and interoperate with advanced Vending Machines. Tag for this vertical is *retail*.

The Home page is as shown below:



Welcome!

Build real-time applications with microsecond latency for any industry in half the time with Oracle Stream Analytics

Distributed Intelligence for IOT

Acquire, analyze, and act on high-volume, high-velocity data from sensors and devices both at the edge and in the data center in real-time

Risk and Fraud Management

Leverage industry's best stream processing platform to assess risk and prevent financial fraud in real-time

Transportation and Logistics

Manage fleet, track assets, and improve supply chain efficiencies by combining streaming data with Oracle's advanced spatial functions

Customer Experience and Consumer Analytics

Know the sentiment of your customers to reduce churn, improve loyalty, make offers, and attract customers in real-time

Telecommunications

Proactively monitor your networks, predict network failures, and prevent distributed denial of service type attacks

Retail

Understand and Apply instant Retail Shopping trends, instigate beneficial shelf life patterns and placements, be responsive to Customers cart utilization and interoperate with advanced Vending Machines

You can navigate to the Catalog or the Patterns page from the home page to get started with Oracle Stream Analytics.

About the Catalog

The Catalog page is the location where resources including pipelines, streams, references, maps, connections, targets, dashboards, predictive models, custom jars, visualizations, and cubes are listed. This is the go-to place for you to perform any tasks in Oracle Stream Analytics.

You can mark a resource as a favorite in the Catalog by clicking on the Star icon. Click the icon again to remove it from your favorites. You can also delete a resource or view its topology using the menu icon to the right of the favorite icon.

The tags applied to items in the Catalog are also listed on the screen below the left navigation pane. You can click any of these tags to display only the items with that tag in the Catalog. The tag appears at the top of the screen. Click **Clear All** at the top of the screen to clear the Catalog and display all the items.

You can include or exclude pipelines, streams, references, predictive models, geo fences, connections, targets, custom jars, visualizations, dashboards, and cubes using the **View All** link in the left panel under **Show Me**. When you click **View All**, a check mark appears beside it and all the components are displayed in the Catalog.

When you want to display or view only a few or selective items in the Catalog, deselect **View All** and select the individual components. Only the selected components will appear in the Catalog.

Typical Workflow for Administering Oracle Stream Analytics

The typical workflow lists the artifacts required to create a pipeline in Oracle Stream Analytics.

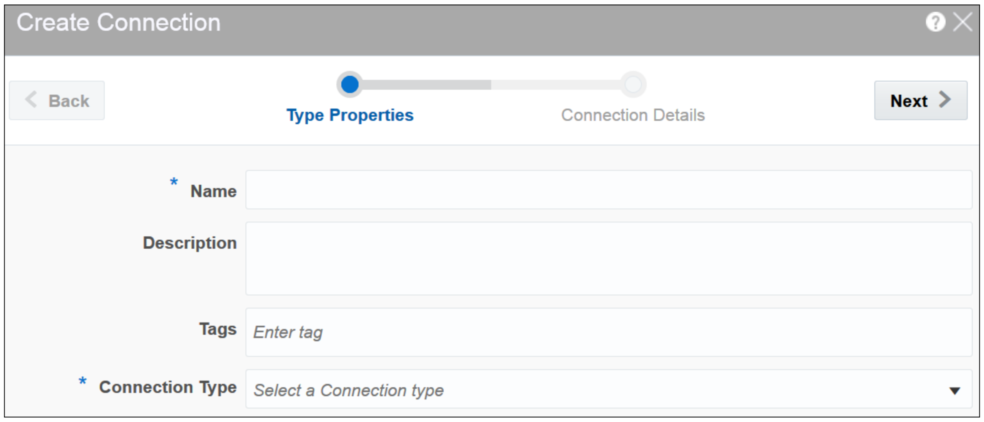
The prerequisites for a pipeline are:

- A connection is required to create a stream, except for a file stream.
- A stream is required to create a pipeline.

Creating a Connection

To create a connection:

1. Click **Catalog** in the left pane.
2. From the **Create New Item** menu, select **Connection**.
3. Provide details for the following fields on the **Type Properties** page and click **Next**:
   - **Name** — name of the connection
   - **Description** — description of the connection
   - **Tags** — tags you want to use for the connection
   - **Connection Type** — type of connection: Coherence, Database, Druid, JNDI, or Kafka



* Name

Description

Tags: Enter tag

* Connection Type: Select a Connection type

4. Enter **Connection Details** on the next screen and click **Save.**

   When the connection type is **Coherence**:

   - **Host name** — the Coherence Extend Proxy Services TCP/IP Server Socket host
   - **Port** — the Coherence Extend Proxy Services TCP/IP Server Socket port

   When the connection type is **Database**:

   - **Connect using** — select the way you want to identify the database; `SID` or `Service name`
   - **Service name/SID** — the details of the service name or SID
   - **Host name** — the host name on which the database is running
   - **Port** — the port on which the database is running. Usually it is `1521`
   - **Username** — the user name with which you connect to the database
   - **Password** — the password you use to login to the database

   When the connection type is **Druid**, provide Zookeeper URL.

   When the connection type is **JNDI**:

   - **JNDI Provider** — select the JNDI service provider
   - **Server Url(s)** — the server url(s) for the JNDI connection; for example: host1:port1, host2:port2
   - **Username** — the user name for authenticating the JNDI connection
   - **Password** — the password for the JNDI connection

   When the connection type is **Kafka**, provide Zookeeper URL.

A connection with the specified details is created.

Cache Configuration for Coherence

Oracle Stream Analytics requires a special coherence cache configuration and the proxy schema, so that it can connect to the coherence.

To enrich stream data with external coherence cluster reference data, you must access external coherence cluster using extend client APIs. To access external cluster as client, you need to configure `cache-config` with `ExtendTcpCacheService` and `ExtendTcpInvocationService`.

Configure the Coherence Cluster

Make sure that you have Coherence for Java is installed.

To configure the external cluster as client:

1. Create an XML file named `cache-config.xml`.
2. Copy the following XML to the file:

```
<?xml version="1.0"?>

<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
   xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-
cache-config
   coherence-cache-config.xsd">
        <caching-scheme-mapping>
            <cache-mapping>
                <cache-name>
                    externalcache*
                </cache-name>
                <schema-name>
                    remote
                </schema-name>
            </cahce-mapping>
        </caching-scheme-mapping>

        <caching-schemes>
            <remote-cache-scheme>
                <scheme-name>
                    remote
                </scheme-name>
                <service-name>
                    ExtendTcpCacheService
                </service-name>
                <initiator-config>
                    <tcp-initiator>
                        <remote-addresses>
                            <socket-address>
                                <address>localhost    </address>
                                <port>9099</port>
                            </socket-address>
                        </remote-addresses>
                    </tcp-initiator>
                    <outgoing-message-handler>
                        <request-timeout>5s</request-timeout>
                    </outgoing-message-handler>
                </initiator-config>
            </remote-cache-scheme>

            <remote-invocation-scheme>
                <scheme-name>extend-invocation</scheme-name>
                <service-name>ExtendTcpInvocationService</service-name>
                <initiator-config>
```

```
                    <tcp-initiator>
                        <remote-addresses>
                            <socket-address>
                                <address>localhost</address>
                                <port>9099</port>
                            </socket-address>
                        </remote-addresses>
                    </tcp-initiator>
                    <outgoing-message-handler>
                        <request-timeout>5s</request-timeout>
                    </outgoing-message-handler>
                </initiator-config>
            </remote-invocation-scheme>
        </caching-schemes>
</cache-config>
```

3. Save and close the file.
4. Test the connection to the cluster.

   ```
   InvocationService service =
   (InvocationService) CacheFactory.getConfigurableCacheFactory().ensureSer
   vice("ExtendTcpInvocationService");
   ```

   `ensureService()` will throw exception if there is no coherence cluster available with the given host and port.
5. Create a coherence reference using a coherence connection.
6. Register the coherence as reference.

The following is the sample code to register the coherence as reference:

```
override def initialize():Unit = {
    repartition = true
    val externalEvent = EventType("externalorders",IntAttr("orderId"),
VarCharAttr("orderDesc", 20))
    val sExtSrcProps = Map(EXT_URL -> "",EXT_ENTITY_NAME ->
"externalcache")
    val jExtSrcProps = new java.util.HashMap[String,String](sExtSrcProps)
    val converter = ConverterFactory(ConverterType.COHERENCE,externalEvent)
    cc.registerEventType(externalEvent)

cc.registerRelation(externalEvent).onExternal(jExtSrcProps,ExtSourceType.CO
HERENCE,converter)
 }

def main(args: Array[String]) {
    cql = "istream(select R.orderId as orderId, R.orderStatus as
orderStatus, Ext.orderDesc as orderDesc from orders[now] as R,
externalorders as Ext where R.orderId = Ext.orderId)"
    name = "CoherenceCorrelation"
    processOrders(args)
    }
}
// EXT_URL is not used for coherence as reference , currently used for
```

```
webservice & database, so this will be set to EMPTY
//EXT_ENTITY_NAME is the cache name of the external coherence cluster
```

For the above example, coherence cache must have key as `orderId <Integer>` and value as `Map of values for orderId and orderDesc`. A sample cache similar to the following will populate:

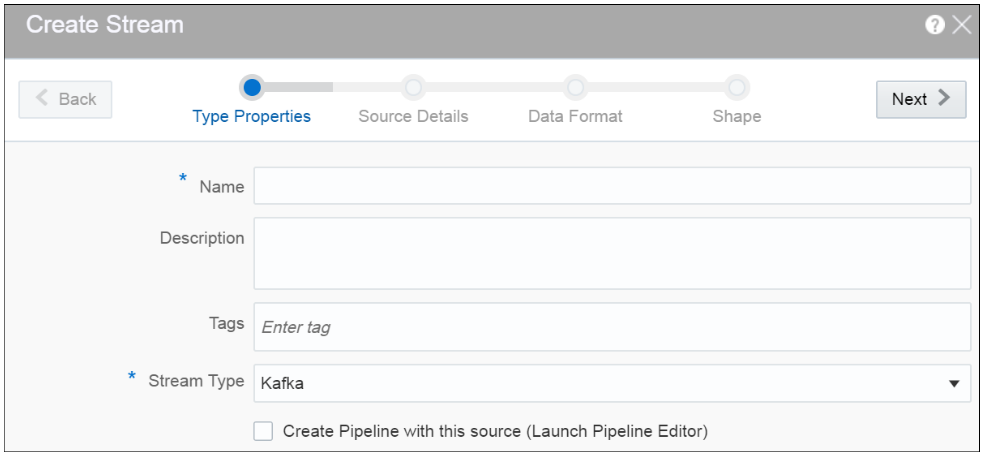
```
NamedCache cache = CacheFactory.getCache("externalcache");
Map<String,Object> order1 = new HashMap<String, Object>();
order1.put("orderId", new Integer(1));
order1.put("orderDesc", "HP Deskjet v2");
Map<String,Object> order2 = new HashMap<String, Object>();
order2.put("orderId", new Integer(2));
order2.put("orderDesc", "Oracle Database 12");
MapString,Object> order3 = new HashMap<String, Object>();
order3.put("orderId", new Integer(3));
order3.put("orderDesc", "Apple iPhone6s");
Map<String,Object> order4 = new HashMap<String, Object>();
order4.put("orderId", new Integer(4));
order4.put("orderDesc", "Logitech Mouse");
cache.put(1,order1);
cache.put(2,order2);
cache.put(3,order3);
cache.put(4,order4);
```

Creating a Stream

A stream is a source of events with a given content (shape).

To create a stream:

1. Navigate to **Catalog**.
2. Select **Stream** in the **Create New Item** menu.
3. Provide details for the following fields on the **Type Properties** page and click **Next**:
   - **Name** — name of the stream
   - **Description** — description of the stream
   - **Tags** — tags you want to use for the stream
   - **Stream Type** — select suitable stream type. Supported types are File, GoldenGate, JMS, and Kafka.



* Name

Description

Tags: Enter tag

* Stream Type: Kafka

Create Pipeline with this source (Launch Pipeline Editor)

4. Provide details for the following fields on the **Source Details** page and click **Next**:

   When the stream type is File:

   - **File Path or URL** — the location of the file that you want to upload
   - **Read whole content** — select this option if you want to read the whole content of the file
   - **Number of events per batch** — the number of events that you want to process per batch
   - **Loop** — select this option if you want to process the file in a loop
   - **Data Format** — the format of the data. The supported types are: CSV and JSON.

   When the stream type is GoldenGate:

   - **Connection** — the connection for the stream
   - **Topic name** — the topic name that receives events you want to analyze
   - **Data Format** — the format of the data. The supported types are: CSV, JSON, AVRO. AVRO is a data serialization system.

   When the stream type is JMS:

   - **Connection** — the connection for the stream
   - **Jndi name** — the Jndi that reads messages from topics, distributed topics, queues and distributed queues
   - **Client ID** — the client to be used for durable subscriber
   - **Message Selector** — the message selector to filter messages. If your messaging application needs to filter the messages it receives, you can use a JMS API message selector, which allows a message consumer to specify the messages it is interested in. Message selectors assign the work of filtering messages to the JMS provider rather than to the application.

     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’
     ```

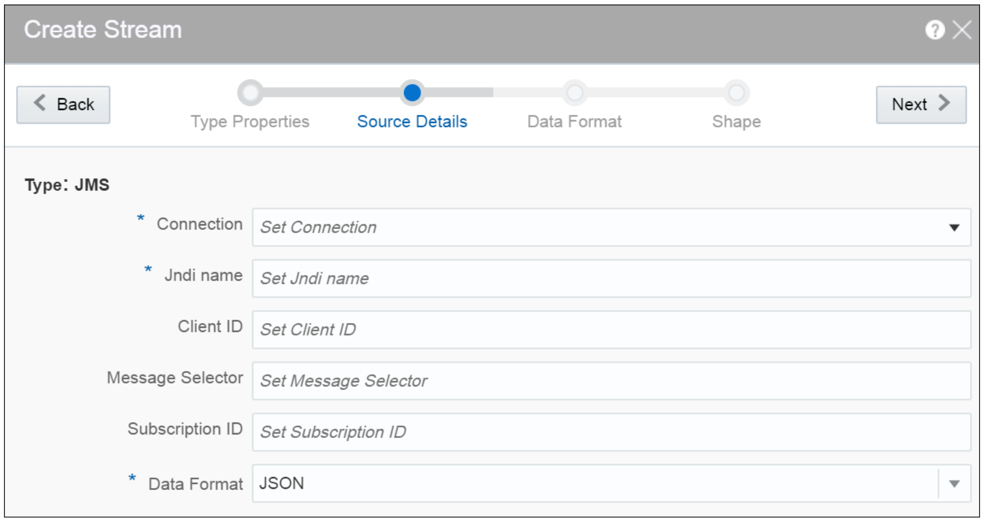
The `createConsumer` and `createDurableSubscriber` methods allow you to specify a message selector as an argument when you create a message consumer.

- **Subscription ID** — the subscription id for durable selector
- **Data Format** — the format of the data. The supported types are: CSV, JSON, AVRO, MapMessage. MapMessage is supported only for JNDI based streams.

  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.

When the stream type is Kafka:

- **Connection** — the connection for the stream
- **Topic name** — the topic name that receives events you want to analyze
- **Data Format** — the format of the data within the stream. The supported types are: CSV, JSON, AVRO.

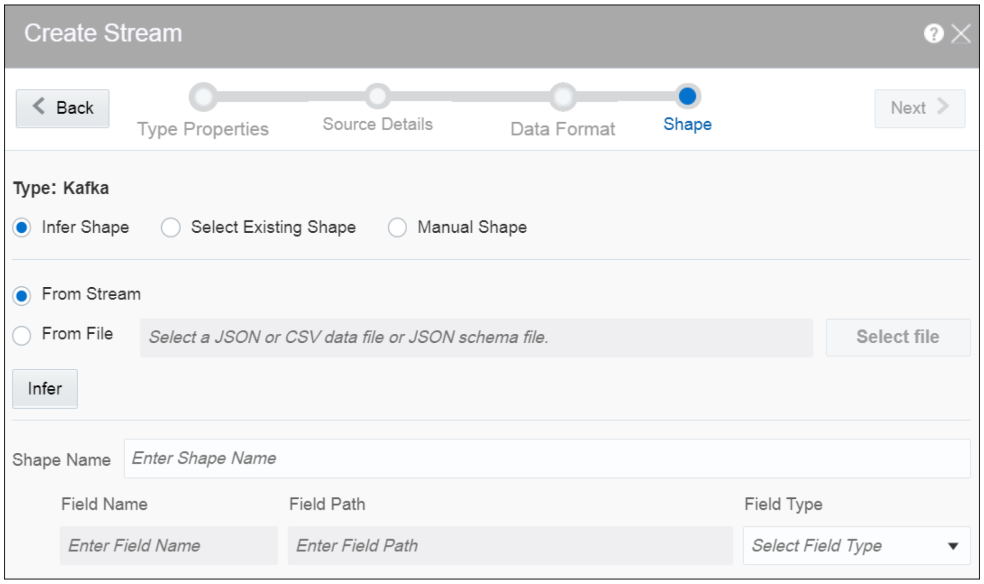


Type: JMS

* Connection	Set Connection
* Jndi name	Set Jndi name
Client ID	Set Client ID
Message Selector	Set Message Selector
Subscription ID	Set Subscription ID
* Data Format	JSON

5. Select one of the mechanisms to define the shape on the **Shape** page:
   - **Infer Shape** — detects the shape automatically from the input data stream.

     You can infer the shape from Kafka, JSON schema file, or CSV message/data file. You can also save the auto detected shape and use it later.
   - **Select Existing Shape** — lets you choose one of the existing shapes from the drop-down list.
   - **Manual Shape** — populates the existing fields and also allows you to add or remove columns from the shape. You can also update the datatype of the fields.



Type: Kafka

[✓ Infer Shape] Select Existing Shape Manual Shape

[✓ From Stream]

From File: Select a JSON or CSV data file or JSON schema file.

Shape Name	Enter Shape Name	
Field Name	Field Path	Field Type
Enter Field Name	Enter Field Path	Select Field Type

A stream is created with specified details.

CSV Data for Pre-defined Formats

When your data format is CSV, select a predefined format based on the variations of CSV data that differs due to the originating source of these CSV. The following table describes the CSV data for each of these predefined formats:

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

Capabilities of JMS Source

The capabilities of JMS Source are listed in the following table:

Capability	Description	Comments
Ability to connect to JMS Cluster	JMS consumer should be able to connect to JMS cluster and handle JMS server fail-over	
Message Format support	Map and TextMessage (JSON, CSV and AVRO)	Does not support xml and object
Message selector	JMS message selector to use to filter messages. Only messages that match the selector will produce events.	
Re-connection	Reconnect to JMS server or JMS cluster	
Read messages from queue/distributed queue		
Read messages from topic	Read messages from JMS topic. By default the subscriber is non-durable	
Support for Durable subscriber	A durable subscriber registers a durable subscription by specifying a unique identity that is retained by the JMS provider. If the consumer reconnects to JMS topic, it would read messages from where it last read.	
T3 Support	Weblogic JMS Protocol	

JMS Server Clean Up

When you create a JMS stream and select the durable subscription option (by providing client ID and subscription ID value), Oracle Stream Analytics creates the durable subscription (if not already present) when the pipeline using this stream is running. When you come out of the pipeline or unpublish the pipeline(or kill the running pipeline), the durable subscription 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 this pipeline anymore.

Creating a Reference

The reference defines a read-only source of reference data to enrich a stream. A stream containing a customer name could use a reference containing customer data to add the customer’s address to the stream by doing a lookup using the customer name.

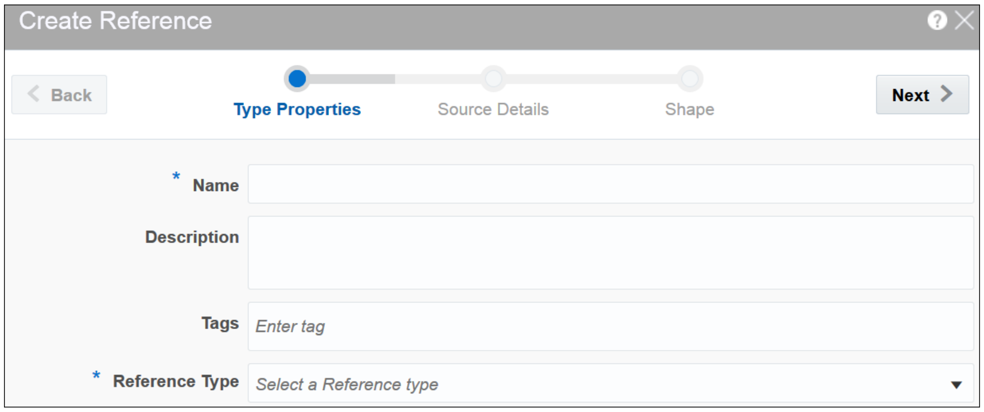
A database reference is a reference to specified table in the database. With cache enabled for database reference, when the values gets pulled from database, they are maintained in coherence cache for reference from where they can be served on next request. A database reference requires a database connection.

A coherence reference can be any external cache defined in coherence cluster that can have data from an external system.

To create a reference:

1. Navigate to **Catalog**.
2. Select **Reference** in the **Create New Item** menu.
3. Provide details for the following fields on the **Type Properties** page and click **Next**:

- **Name** — name of the reference
- **Description** — description of the reference
- **Tags** — tags you want to use for the reference
- **Reference Type** — the reference type of the reference. The supported reference types are: Coherence and Database.

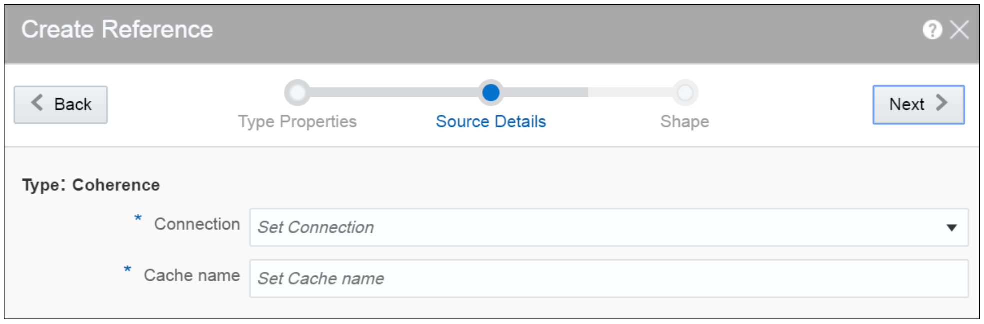


* Name	
Description	
Tags	Enter tag
* Reference Type	Select a Reference type

4. Provide details for the following fields on the **Source Details** page and click **Next**:

   When the reference type is **Coherence**, enter or select appropriate values for:

   - **Connection** — the connection for the coherence reference



Type: Coherence

* Connection	Set Connection
* Cache name	Set Cache name

   - **Cache name** — the name of the cache to enable caching for better performance at the cost of higher memory usage of the Spark applications. Caching is supported only for single equality join condition. When you update the cache, the application will receive the updated data very quickly.

   Coherence reference has data in key-value pairs. Key is object type and value is `Map<String,Object>`. `Map<String,Object>` is map of attribute names and values, attributes list should match with external event type. In this release, only external schema for key and value s supported.

   When the reference type is **Database Table**, enter or select appropriate values for:

   - **Connection** — the connection for the database reference
   - **Enable Caching** — select this option if you want to enable caching
   - **Expiry Delay** — the amount of time from last update that 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**:

   When the reference type is **Coherence**:

   - **Select Existing Shape** — select a shape that you want to use for the reference

     NOT_SUPPORTED:

     Ensure that you do not use any of the CQL reserved words as the column names. If you use the reserved keywords, you cannot deploy the pipeline.

   - **Manual Shape** — select this option if you want to define your own shape

   Note:

   When you load coherence data, ensure that you include precision and scale for number type. Only when these values are specified, the join works. For example,

   ```
   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);
   ```

   When the reference type is **Database Table**:

   - **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`
- `timestamp` (without timezone)
- `date time` (without timezone)

Note:

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

A reference is created with the specified details.

Limitations of Coherence as Reference

With coherence as reference, there are a few limitations:

- You cannot test the connection
- You need to specify the cache name manually
- Only equal operator is allowed while establishing a correlation with coherence reference
- You must use manual shape

Creating a Dashboard

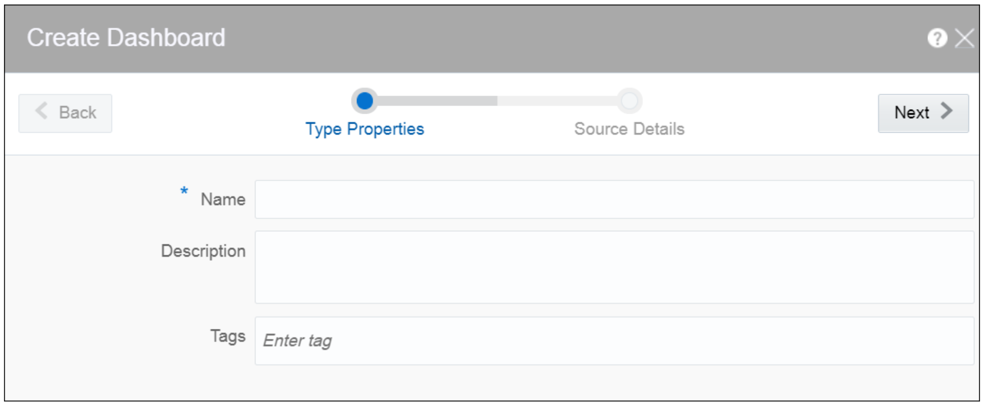
Dashboard is a visualization tool that helps you look at and analyze the data related to a pipeline based on various metrics like visualizations. A dashboard can have visualizations created out of cubes as well.

Dashboard is an analytics feature. You can create dashboards in Oracle Stream Analytics to have a quick view at the metrics.

To create a dashboard:

1. Go to the **Catalog**.
2. Select **Dashboard** in the **Create New Item** menu.

   The Create Dashboard screen appears.



3. Provide suitable details for the following fields:
   - **Name** — enter a name for the dashboard. this is a mandatory field.
   - **Description** — enter a suitable description for the dashboard. This is an optional field.
   - **Tags** — enter or select logical tags to easily identify the dashboard in the catalog. This is an optional field.

4. Click **Next**.
5. Enter a custom stylesheet for the dashboard. This is an optional step.
6. Click **Save**.

   You can see the dashboard in the Catalog.

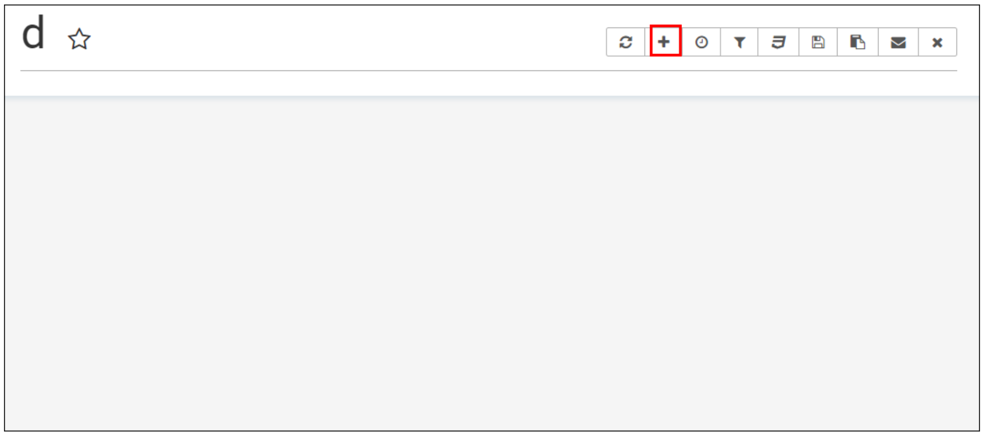
After you have created the dashboard, it is just an empty dashboard. You need to start adding visualizations to the dashboard.

Editing a Dashboard

To edit a dashboard:

1. Click the required dashboard in the catalog.

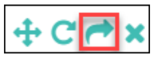
   The dashboard opens in the dashboard editor.

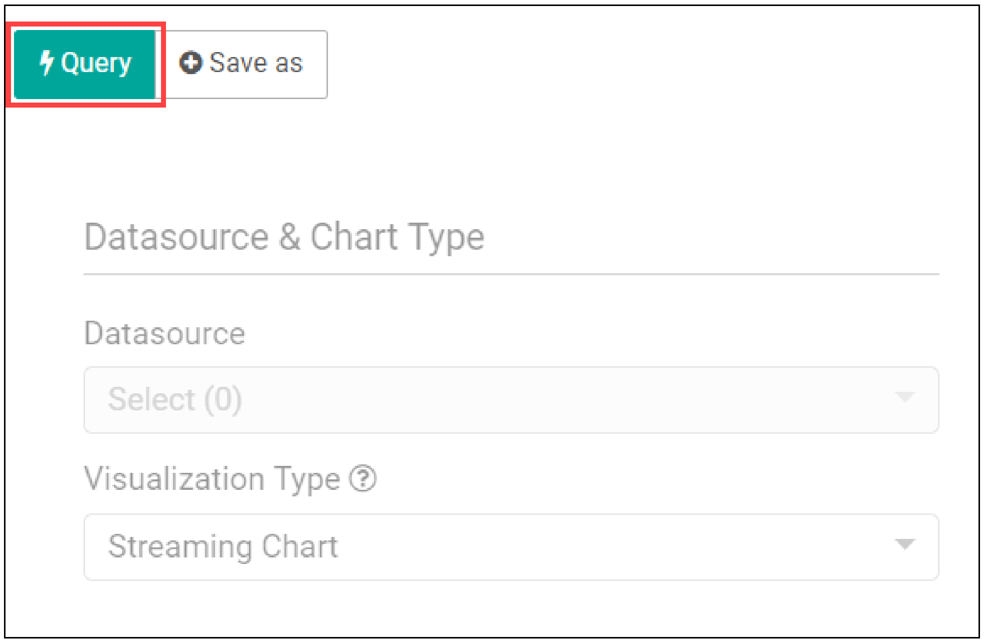
   

2. Click the **Add a new visualization** icon to see a list of existing visualizations. Visualizations from the pipelines and as well as from the cube explorations appear here. Go through the list, select one or more visualizations and add them to the dashboard.
3. Click the **Specify refresh interval** icon 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 just a client side setting and is not persisted with the Superset Version `0.17.0`.
4. Click the **Apply CSS to the dashboard** icon to select a CSS. You can also edit the CSS in the live editor.

   You can also see the active filter applied to the dashboard by clicking the **Active dashboard filters** icon. You can save the link to the dashboard or email the link to someone using the **Copy the link to the clipboard** and **Email the link** icons respectively.
5. Click the **Save** icon to save the changes you have made to the dashboard.
6. Hover over the added visualization, click the **Explore chart** icon to open the chart editor of the visualization.



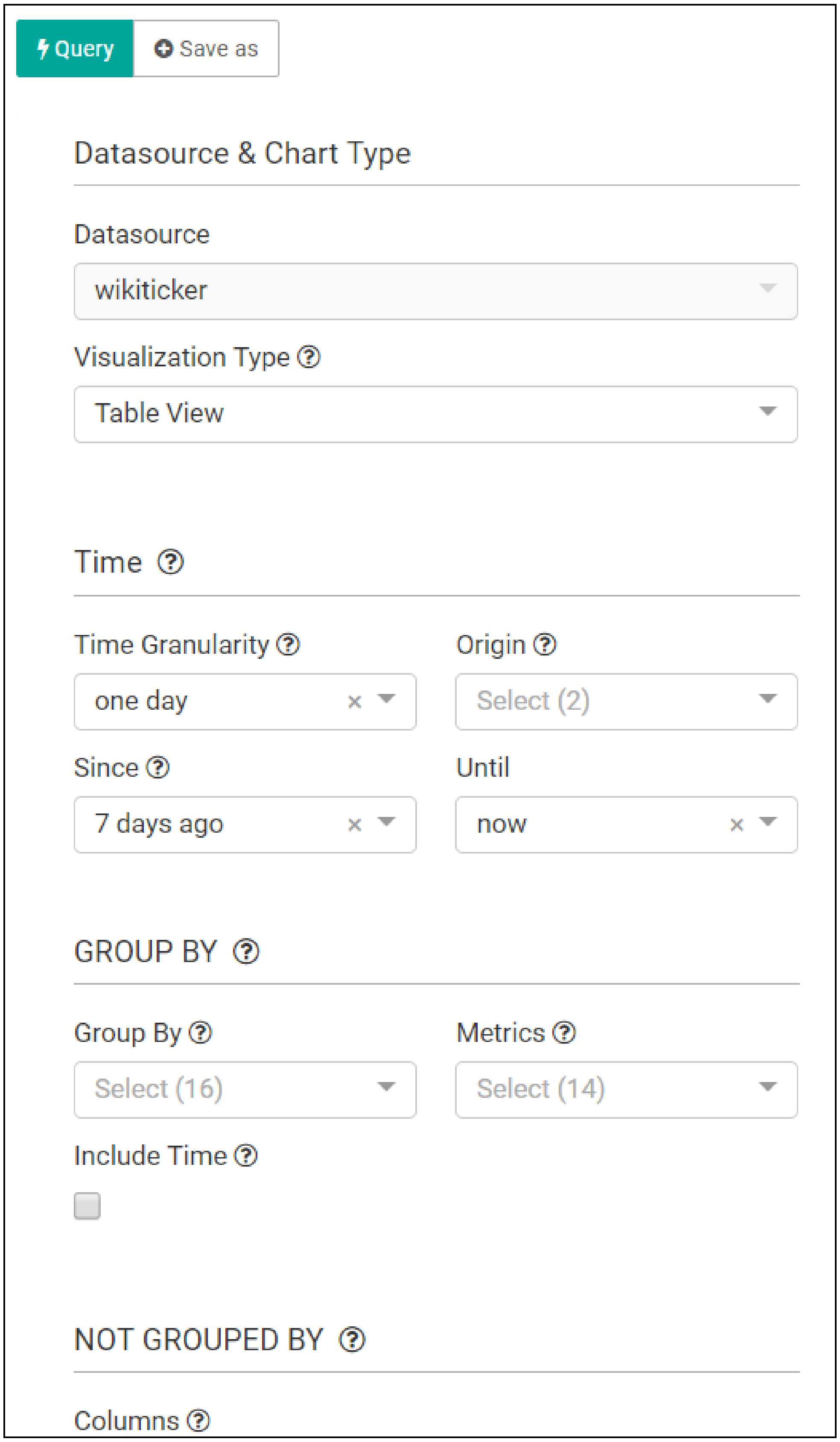
Datasource & Chart Type

Datasource: Select (0)

Visualization Type: Streaming Chart

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:



Datasource & Chart Type

Datasource: wikiticker

Visualization Type: Table View

Time

Time Granularity: one day

Origin: Select (2)

Since: 7 days ago

Until: now

GROUP BY

Group By: Select (16)

Metrics: Select (14)

Include Time: ☐

NOT GROUPED BY

Columns

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.

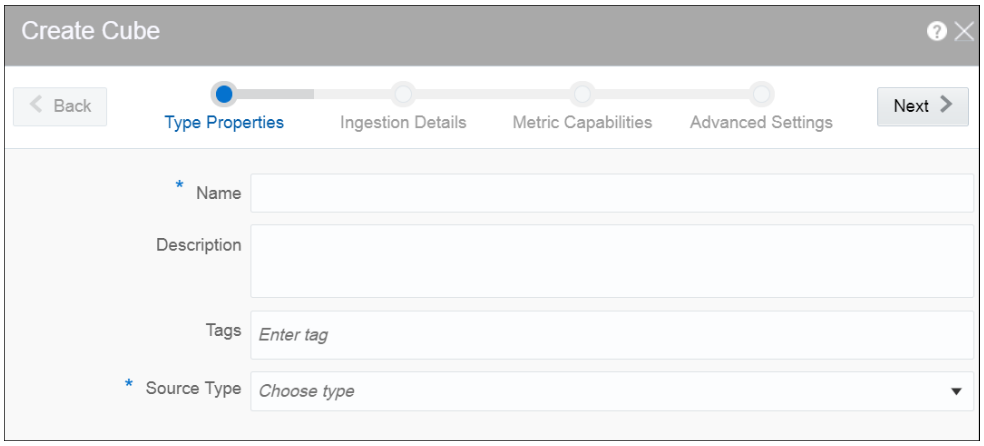
7. Click **Save as** to make the following changes to the dashboard:
   - Overwrite the visualization
   - Overwrite the current visualization with a different name
   - Add the visualization to an existing dashboard
   - Add the visualization to a new dashboard

Creating a Cube

Cube is a data structure that helps in quickly analyzing the data related to a business problem on multiple dimensions.

To create a cube:

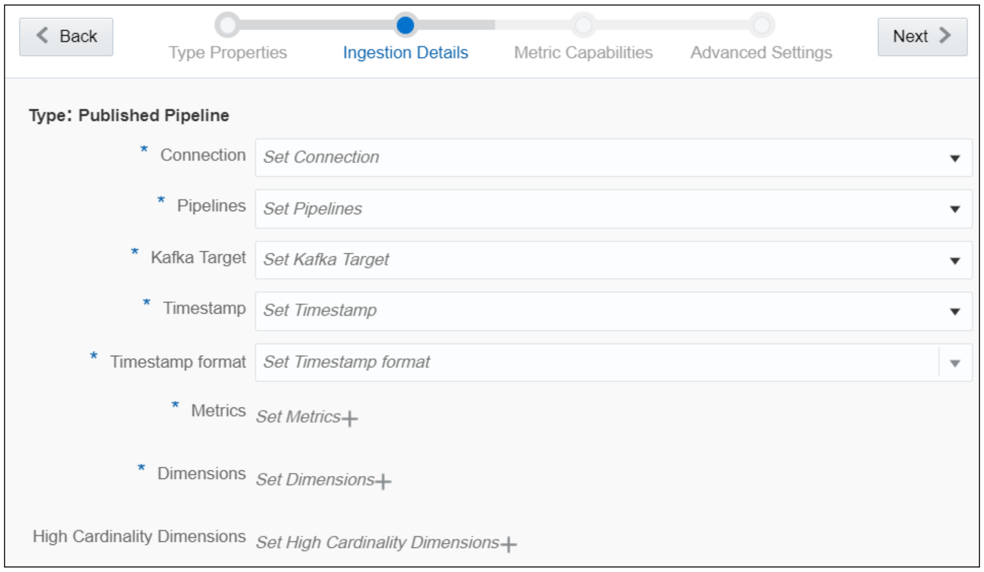
1. Go to the **Catalog**.
2. From the **Create New Item** menu, select **Cube**.
3. On the Create Cube — Type Properties screen, provide suitable details for the following fields:
   - **Name** — enter a name for the cube. This is a mandatory field.
     Make sure that the names you use for the underlying sources for the cube like Pipeline Name, Druid Connection, and Kafka Target use names that contain alphanumeric, hyphen, and underscore characters.
   - **Description** — enter a suitable description for the cube. This is an optional field.
   - **Tags** — enter or select logical tags for the cube. This is an optional field.
   - **Source Type** — select the source type from the drop-down list. Currently, Published Pipeline is the only supported type. This is a mandatory field.



* Name	
Description	
Tags	Enter tag
* Source Type	Choose type

4. Click **Next** and provide suitable details for the following fields on the Ingestion Details screen:
   - **Connection** — the connection for the cube. This is a mandatory field.
   - **Pipelines** — select a pipeline to be used as the base for the cube. This is a mandatory field.
   - **Kafka Target** — the Kafka target for the cube. This is a mandatory field.

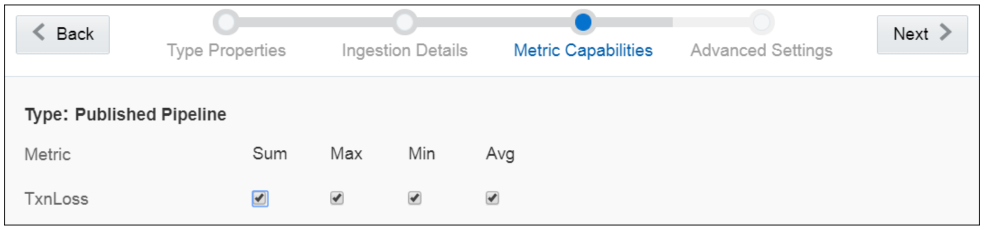
- **Timestamp** — select a column from the pipeline to be used as the timestamp. This is a mandatory field.
- **Timestamp format** — select or set a suitable format for the timestamp using Joda time format. This is a mandatory field. *auto* is the default value.
- **Metrics** — select metrics for creating measures
- **Dimensions** — select dimensions for group by
- **High Cardinality Dimensions** — high cardinality dimensions such as unique IDs. Hyperlog approximation will be used.



Type: Published Pipeline

* Connection	Set Connection
* Pipelines	Set Pipelines
* Kafka Target	Set Kafka Target
* Timestamp	Set Timestamp
* Timestamp format	Set Timestamp format
* Metrics	Set Metrics+
* Dimensions	Set Dimensions+
High Cardinality Dimensions	Set High Cardinality Dimensions+

5. Click **Next** and select the required values for the **Metric** on the Metric Capabilities screen.

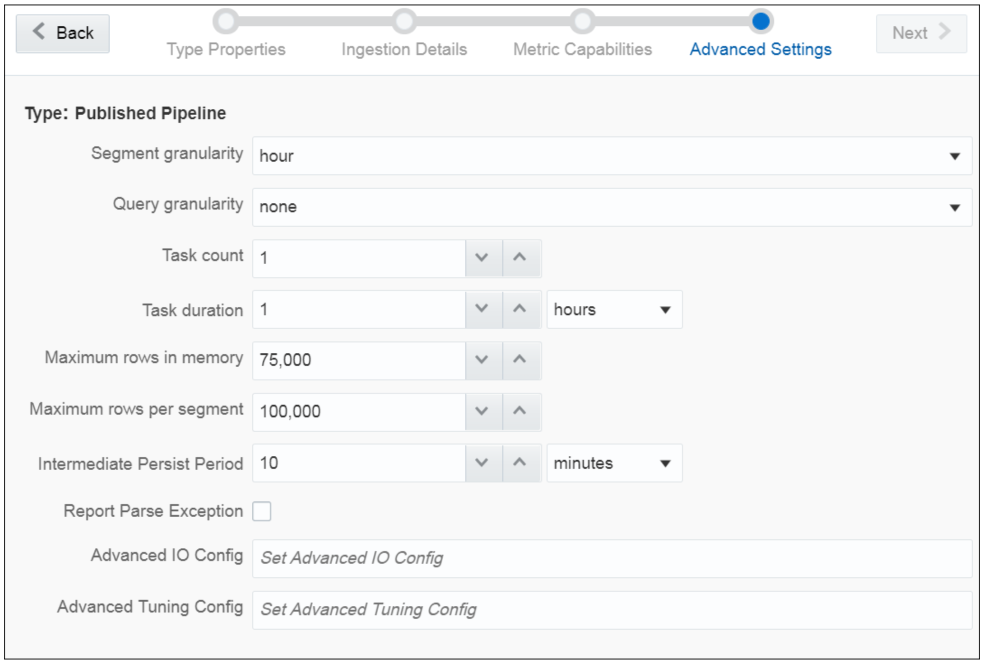


Type: Published Pipeline

Metric	Sum	Max	Min	Avg
TxnLoss	✓	✓	✓	✓

6. Click **Next** and make any changes, if required, on the Advanced Settings screen.
   - **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`.



Type: Published Pipeline

Segment granularity	hour
Query granularity	none
Task count	1
Task duration	1 hours
Maximum rows in memory	75,000
Maximum rows per segment	100,000
Intermediate Persist Period	10 minutes
Report Parse Exception	☐
Advanced IO Config	Set Advanced IO Config
Advanced Tuning Config	Set Advanced Tuning Config

7. Click **Save** to save the changes you have made.

You can see the cube you have created in the catalog.

Exploring a Cube

When you create druid based cube, you can explore data in it.

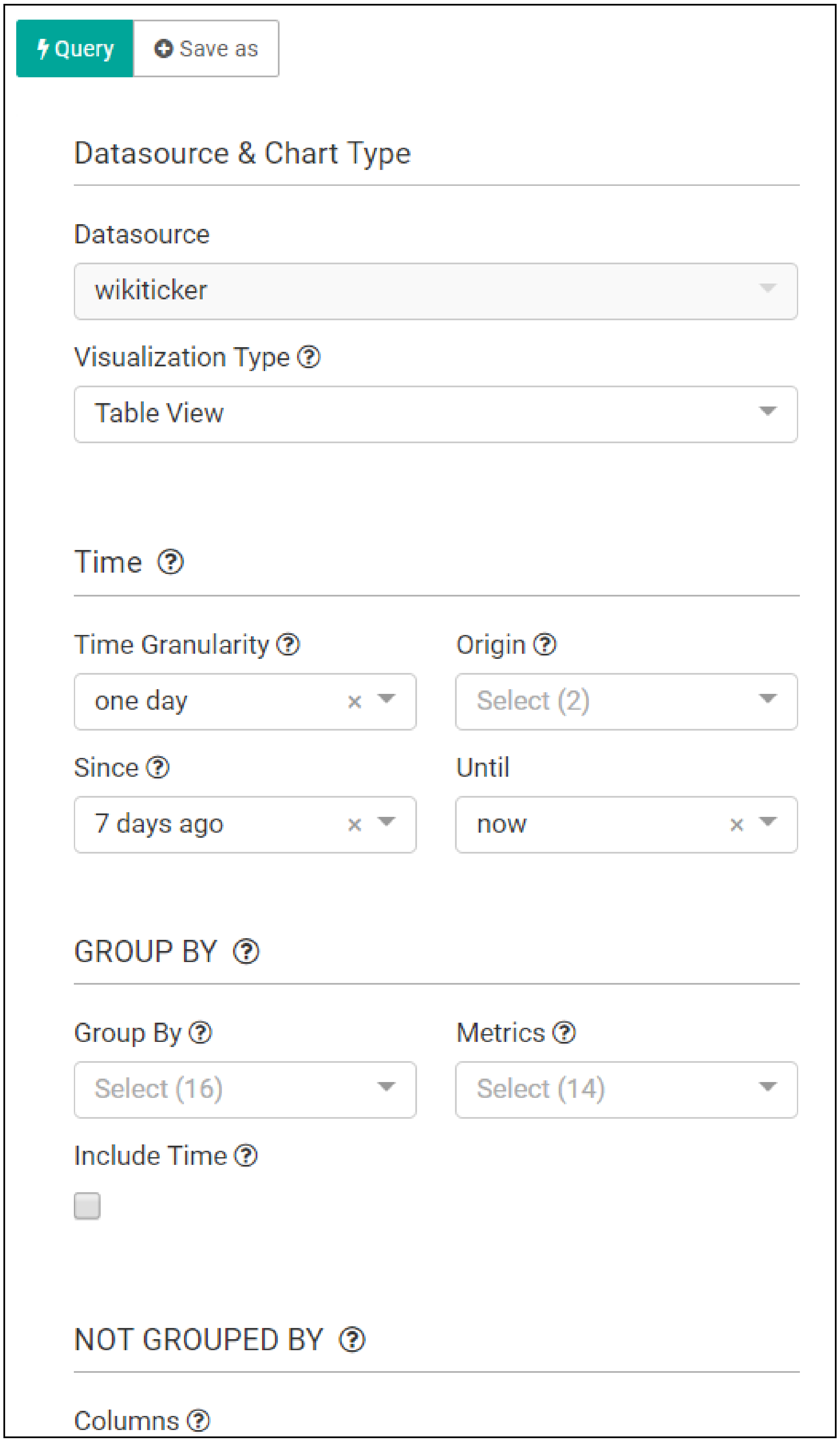
To explore a cube:

1. In the **Catalog**, click the cube that you want to explore.

   The Cube Exploration canvas appears.
2. Construct a query by setting the various parameters.
   - Visualization Type — the type of visualization to be used for displaying data. The supported visualizations are:

Distribution - Bar Chart	Separator
Sunburst	Pie Chart
World Cloud	Sankey
Time Series - Line Chart	Treemap
Directed force Layout	Time Series - Dual Axis Line Chart
Calendar Heatmap	World Map
Time Series - Bar Chart	Box Plot
Filter Box	Time Series - Percent Change
Bubble Chart	iFrame
Time Series - Stacked	Bullet Chart
Streaming Chart	Table View
Big Number with Trendline	Parallel Coordinates
Markup	Big Number
Heatmap	Pivot Table
Histogram	Horizon

   - Time — time related form attributes like time granularity, origin (starting point of time), and time range
   - Group By — parameters to aggregate the query data
   - Not Grouped By — parameter to query atomic rows
   - Options
   - Filters — columns that you can use in filters
   - Result Filters — columns that you can use in result filters



Query | Save as

Datasource & Chart Type

Datasource: wikiticker

Visualization Type: Table View

Time

Time Granularity: one day

Origin: Select (2)

Since: 7 days ago

Until: now

GROUP BY

Group By: Select (16)

Metrics: Select (14)

Include Time: ☐

NOT GROUPED BY

Columns

3. Click **Query** to run the query with the defined parameters.

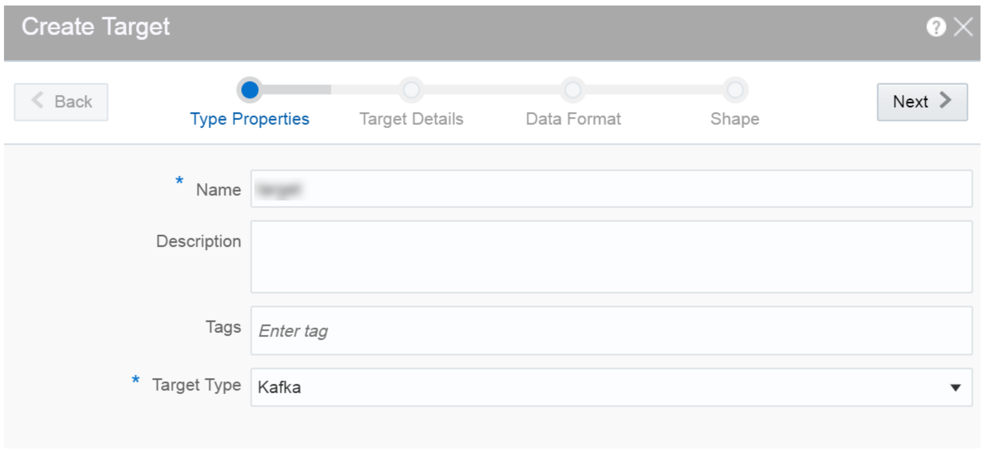
4. Click **Save As** to save the cube exploration. You can save it as a visualization, choose to add it to an existing dashboard, not to add to a dashboard, or to a new dashboard.

Creating a Target

The target defines a destination for output data coming from a pipeline.

To create a target:

1. Navigate to **Catalog**.
2. Select **Target** in the **Create New Item** menu.
3. Provide details for the following fields on the **Type Properties** page and click **Save** and **Next**:
   - **Name** — name of the target
   - **Description** — description of the target
   - **Tags** — tags you want to use for the target
   - **Target Type** — the transport type of the target. Supported types are JMS, Kafka and Rest. The target is a sink for the output event. Each type of target is a different sink system and therefore different configuration parameters are required for different types.



* Name

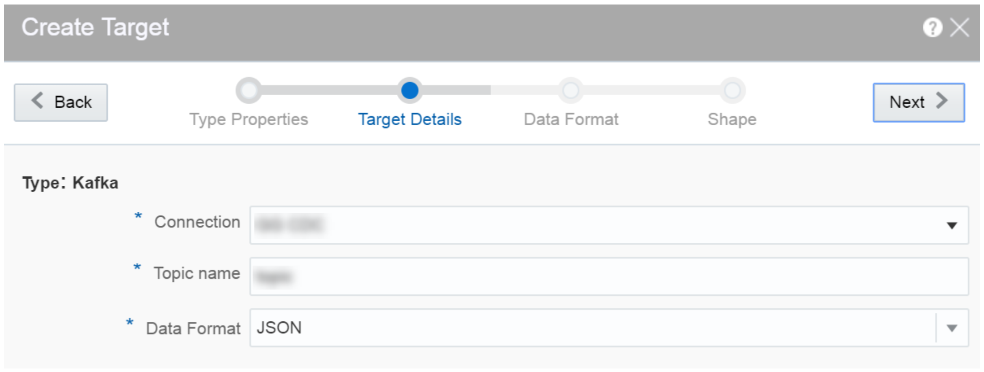
Description

Tags: Enter tag

* Target Type: Kafka

4. Provide details for the following fields on the **Target Details** page and click **Next**:

   When the target type is JMS:
   - **Connection** — the connection for the target
   - **Jndi name** — the topic or queue name defined in Jndi to be used in the target



Type: Kafka

* Connection

* Topic name

* Data Format JSON

- **Data Format** — select a suitable data format. This is a mandatory field. The supported data format types are: CSV and JSON.

When the target type is Kafka:

- **Connection** — the connection for the target
- **Topic Name** — the Kafka topic to be used in the target
- **Data Format** — select a suitable data format. This is a mandatory field. The supported data format types are: CSV and JSON.

When the target type is REST:

- **URL** — enter the REST service URL. This is a mandatory field.
- **Custom HTTP headers** — set the custom headers for HTTP. This is an optional field.
- **Batch processing** — select this option to send events in batches and not one by one. Enable this option for high throughput pipelines. This is an optional field.
- **Data Format** — select a suitable data format. This is a mandatory field.

Click **Test connection** to check if the connection has been established successfully.

Testing REST targets is a heuristic process. It uses proxy settings. The testing process uses GET request to ping the given URL and returns success if the server returns `OK (status code 200)`. The return content is of the type of `application/json`.

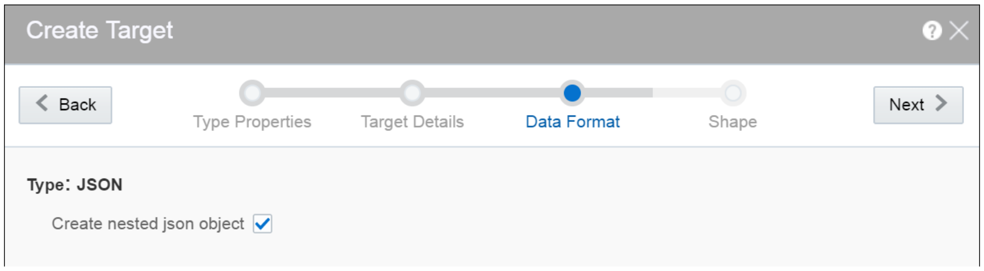
5. Provide details for the following fields on the **Data Format** page and click **Next**:

When the data format type is CSV:

- **CSV Predefined Format** — select a predefined CSV format. This supported formats are: Excel, InfomixUnload, InfomixUnloadCsv, MySQL, PostgreSQLCsv, PostgreSQLText.
- **Create the header row** — select this option if you want to create a header row in the target.

When the data format type is JSON:

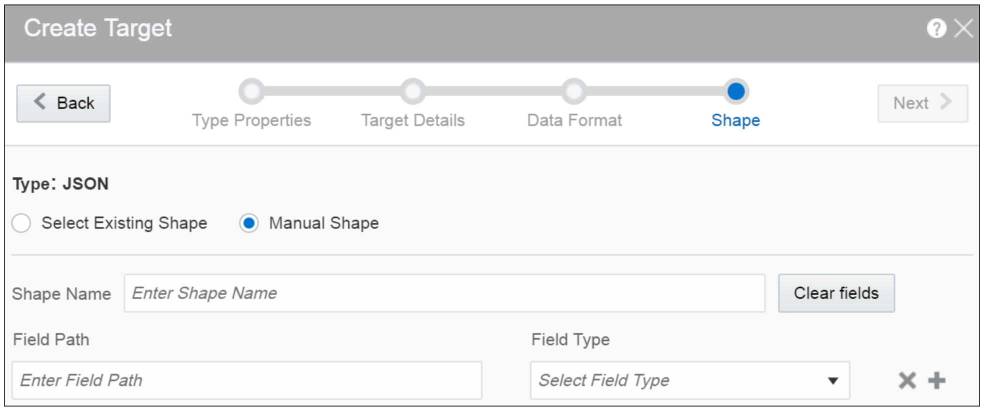
- **Create nested json object** — select this option if you want a nested json object to be created for the target



Type: JSON

Create nested json object ✓

6. Select one of the mechanisms to define the shape on the **Shape** page and click **Save**:
   - **Select Existing Shape** lets you choose one of the existing shapes from the drop-down list.
   - **Manual Shape** populates the existing fields and also allows you to add or remove columns from the shape. You can also update the datatype of the fields.



Type: JSON

Select Existing Shape [✓ Manual Shape]

Shape Name: Enter Shape Name

Field Path: Enter Field Path

Field Type: Select Field Type

A target is created with specified details.

Creating Target from Pipeline Editor

Alternatively, you can also create a target from the pipeline editor. When you click **Create** in the target stage, you are navigated to the **Create Target** dialog box. Provide all the required details and complete the target creation process. When you create a target from the pipeline editor, the shape gets pre-populated with the shape from the last stage.

Creating a Geo Fence

Geo fences are further classified into two categories: manual geo fence and database-based geo fence.

Create a Manual Geo Fence

To create a manual geo fence:

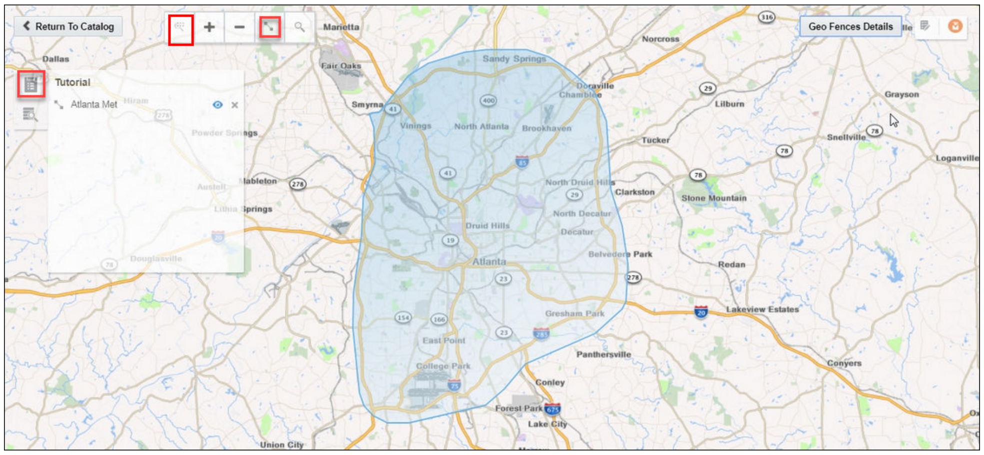
1. Navigate to the **Catalog** page.
2. Click **Create New Item** and select **Geo Fence** from the drop-down list.

   The Create Geo Fence dialog opens.

3. Enter a suitable name for the **Geo Fence**.
4. Select **Manually Created Geo Fence** as the **Type**.
5. Click **Save**.

   The Geo Fence Editor opens. In this editor you can create the geo fence according to your requirement.
6. Within the Geo Fence Editor, **Zoom In** or **Zoom Out** to navigate to the required area using the zoom icons in the toolbar located on the top-left side of the screen.

   You can also use the **Marquee Zoom** tool to zoom a specific area on the map. You can mark an area using the marquee zoom and that area in map is zoomed.
7. Click the **Polygon Tool** and mark the area around a region to create a geo fence.



8. Enter a name and description, and click **Save** to save your changes.

Update a Manual Geo Fence

To update a manual geo fence:

1. Navigate to the **Catalog** page.
2. Click the name of the geo fence you want to update.

   The Geo Fence Editor opens. You can edit/update the geo fence here.

Search Within a Manual Geo Fence

You can search the geo fence based on the country and a region or address. The search field allows you search within the available list of countries. When you click the search results tile in the left center of the geo fence and select any result, you are automatically zoomed in to that specific area.

Delete a Manual Geo Fence

To delete a manual geo fence:

1. Navigate to **Catalog** page.
2. Click **Actions**, then select **Delete Item** to delete the selected geo fence.

Create a Database-based Geo Fence

To create a database-based geo fence:

1. Navigate to **Catalog** page.
2. Click **Create New Item** and then select **Geo Fence** from the drop-down list.

   The Create Geo Fence dialog opens.
3. Enter a suitable name for the geo fence.
4. Select **Geo Fence** from Database as the Type.
5. Click **Next** and select **Connection**.
6. Click **Next**.

   All tables that have the field type as `SDO_GEOMETRY` appear in the drop-down list.
7. Select the required table to define the shape.
8. Click **Save**.

Note:

You cannot edit/update database-based geo fences.

Delete a Database-based Geo Fence

To delete a database-based geo fence:

1. Navigate to **Catalog** page.
2. Click **Actions** and then select **Delete Item** to delete the selected geo fence.

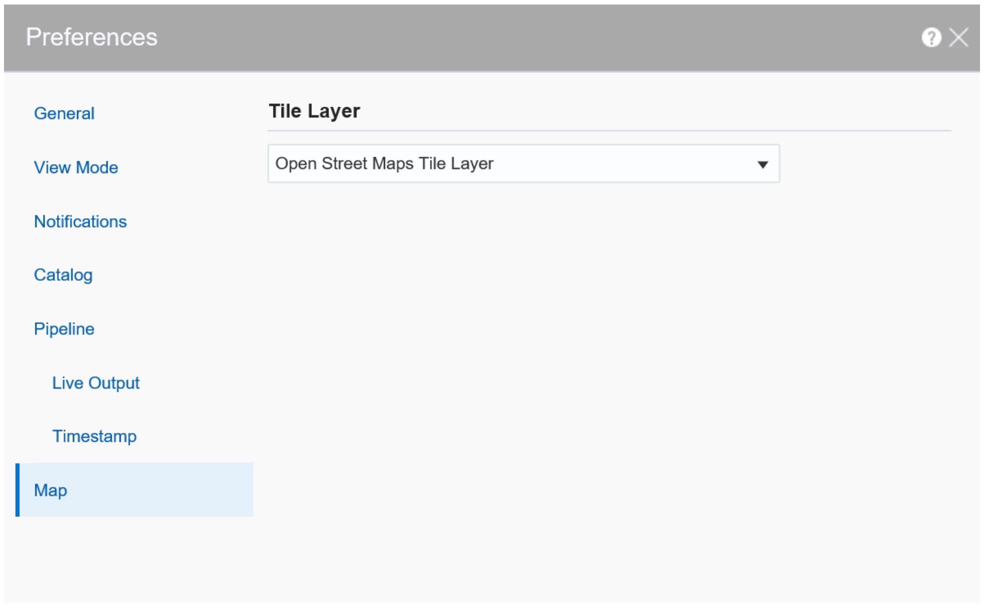
Display the Map Using Tile Layers

Tile layer is the base map that provides immediate geographic context. Tiles are stored in the map tile server. <ph ishcondition="Product_Family=Cloud" varref="streaming">Stream Analytics</ph><ph ishcondition="Product_Family=OnPremise" varref="osa">Oracle Stream Analytics</ph> supports two types of tile layers. Open Street Maps tile layer is a free map. And, Elocation tile layer is an Oracle tile layer. 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.

You can choose if you would like to see the map in Elocation tile layer or Open Street Maps tile layer. To set your preference:

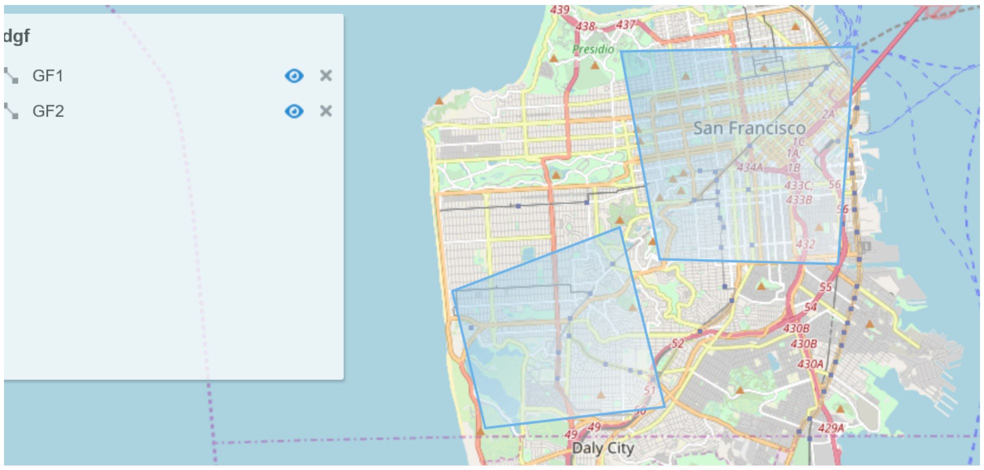
1. Click the user name in the top right corner of the screen.
2. Click **Preferences**. The Preferences page opens.
3. Click **Map**.
4. Under **Tile Layer**, choose **Open Street Maps Tile Layer** option from the drop-down list.



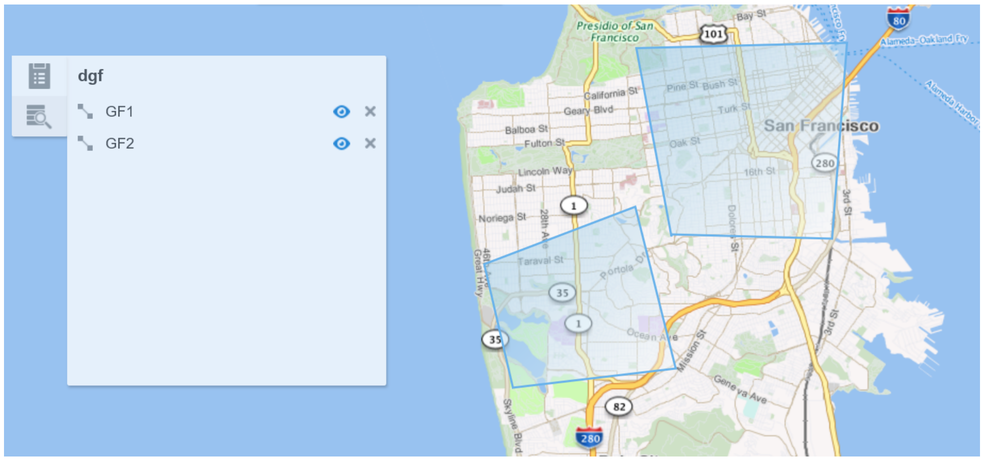
Tile Layer

Open Street Maps Tile Layer

5. Click **Save**. The map looks like this:



6. To display the map in Elocation tile layer, follow steps 1 to 3.
7. From the **Tile Layer** drop-down list, choose **Elocation Tile Layer**.
8. Click **Save**. The map looks like this:



Creating a Predictive Model

To create a predictive model:

1. In the **Create New Item** menu, select **Predictive Model (Beta)**.

   The Create Predictive Model page opens.

2. Under **Type Properties** do the following and then click **Next**:

   a. In the **Name** field, enter a meaningful name for your PMML model.

   b. In the **Predictive Model Type** drop-down list, select **PMML Model**.

   > **Note:**
   >
   > Only PMML Models up to version 4.1 are supported in this release.

3. Under **Predictive Model Details**, do the following and click **Save**:

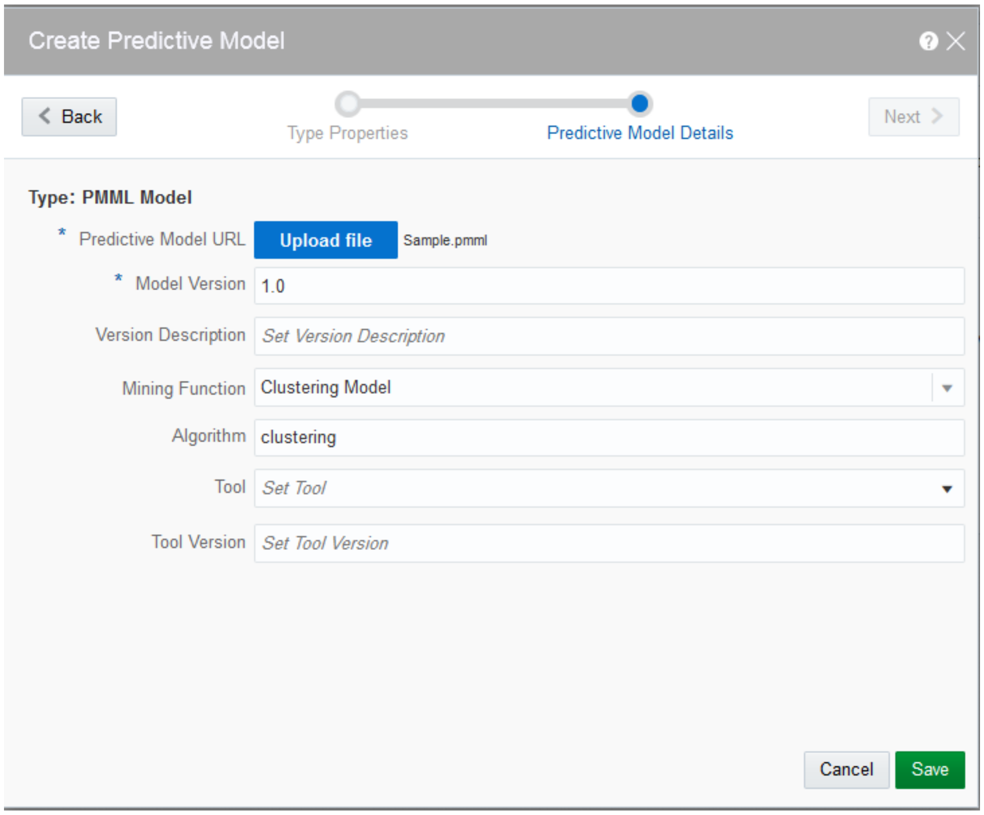
   a. For **Predictive Model URL**, upload your PMML file.

   b. In the **Model Version** field, enter the version of this artifact. For example, `1.0`.

   c. (Optional) In the **Version Description**, enter a meaningful description for your PMML file.

   d. In the **Algorithm** field, accept the default. The algorithm is derived from the PMML file you have uploaded.

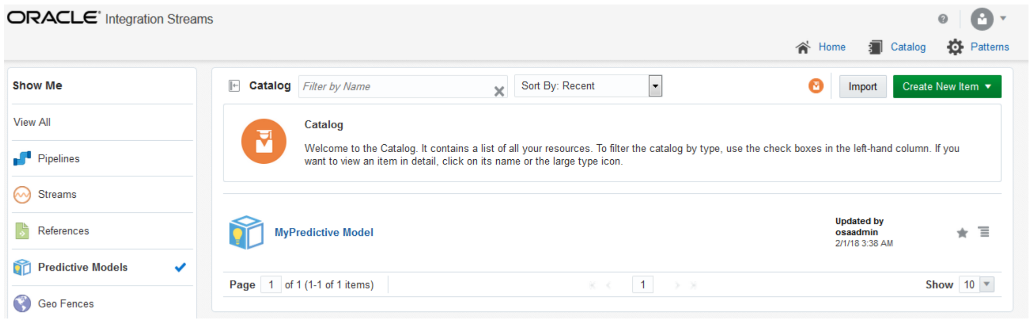
   e. (Optional) In the **Tool** drop-down list, select the tool with which you created your PMML file.



Type: PMML Model

* Predictive Model URL	Sample.pmml
* Model Version	1.0
Version Description	*Set Version Description*
Mining Function	Clustering Model
Algorithm	clustering
Tool	*Set Tool*
Tool Version	*Set Tool Version*

Your predictive model has been created. It is displayed in the **Catalog** if you have selected the **Predictive Models** option under **Show Me**.



Catalog

Welcome to the Catalog. It contains a list of all your resources. To filter the catalog by type, use the check boxes in the left-hand column. If you want to view an item in detail, click on its name or the large type icon.

MyPredictive Model — Updated by osaadmin 2/1/18 3:38 AM

Page 1 of 1 (1-1 of 1 items) Show 10

Limited Support for Predictive Models

The menu commands for creating Predictive Models and Scoring Stages are marked **Beta**, for example, **Predictive Model (Beta)**. The **Beta** label indicates that the functionality has been tested, but is not fully supported. The import and scoring of Predictive Models might contain undocumented limitations and you should use them as is.

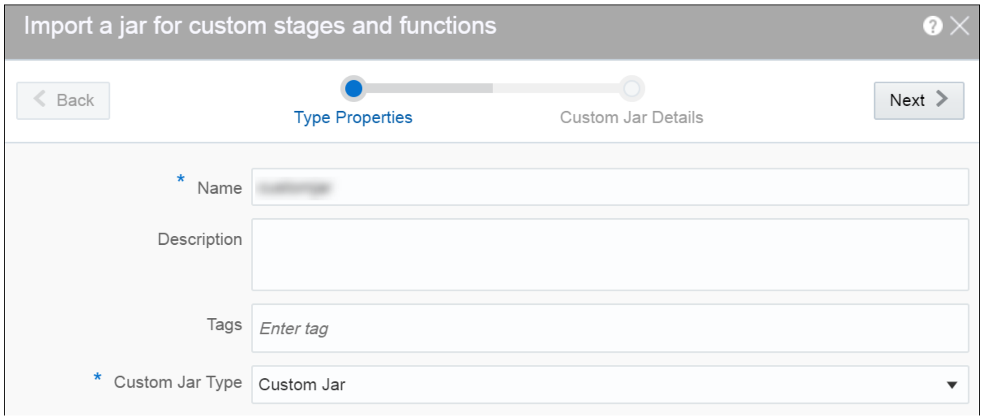
Creating a Custom Jar

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. In the **Create New Item** menu, select **Custom Jar**.

   The Import a jar for custom stages and functions wizard appears.
2. On the **Type Properties** page, enter/select suitable values and click Next:
   a. In the **Name** field, enter a meaningful name for the custom jar you are trying to import into the application.
   b. In the **Description** field, provide a suitable description.
   c. In the **Tags** field, select one or more of existing tags, or enter your own tags.
   d. In the **Custom Jar Type** drop-down list, select **Custom Jar**.



* Name	
Description	
Tags	Enter tag
* Custom Jar Type	Custom Jar

3. On the **Custom Jar Details** page, click **Upload file**, select the jar file that you want to import into the application, and then click **Save**.

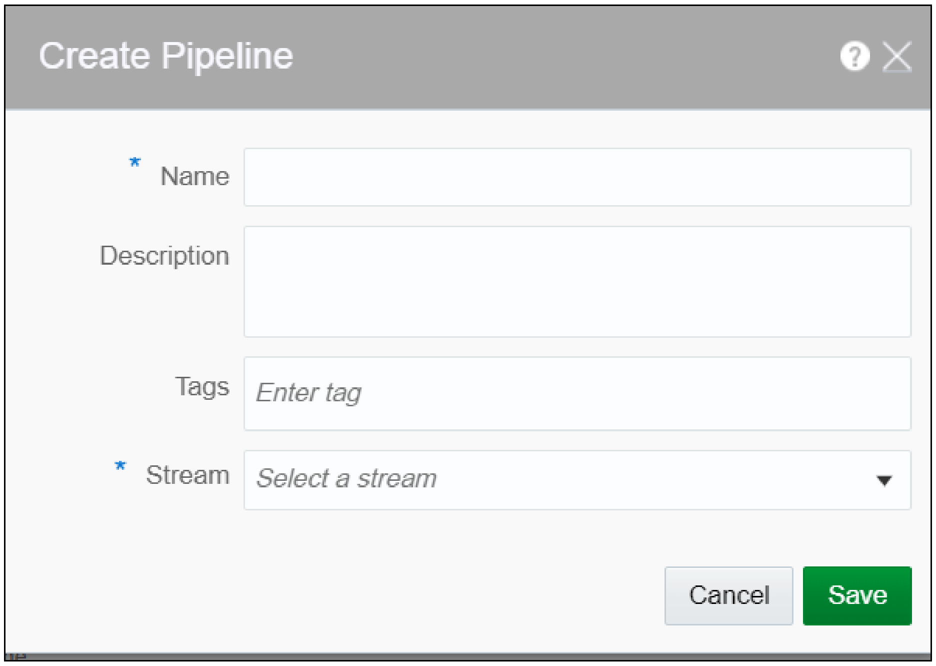
   Make sure that the jar file you select for uploading is a valid jar file and includes all the required dependencies.

Creating a Pipeline

A pipeline is a Spark application where you implement your business logic. It can have multiple stages such as a query stage, a pattern stage, a business rule stage, a query group stage, a custom stage and many more.

To create a pipeline:

1. Navigate to **Catalog**.
2. Select **Pipeline** in the **Create New Item** menu.
3. Provide details for the following fields and click **Save**:
   - **Name** — name of the pipeline
   - **Description** — description of the pipeline
   - **Tags** — tags you want to use for the pipeline
   - **Stream** — the stream you want to use for the pipeline



A pipeline is created with specified details.

Configuring a Pipeline

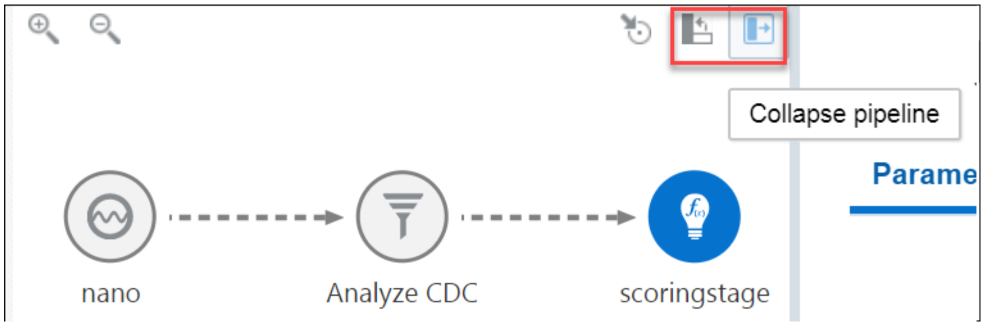
You can configure the pipeline to use various stages like query, pattern, rules, query group, scoring, and custom stage from custom jars.

Pipeline Editor

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

The pipelines in Oracle Stream Analytics can vary from being very simple to highly complex. Complex pipelines have various stages branching out from each/any stage of the pipeline. In other words, you can add any type of stage to any of the existing stage in the pipeline.

You can delete any stage that does not have any children without breaking the pipeline. You can expand/collapse a pipeline, switch the layout of the pipeline to vertical or horizontal, and zoom in or zoom out the pipeline. You can adjust the pipeline pane, editor pane, and the live output table pane using the resizing arrows.



The pipeline editor allows you to see the relationship and dependencies between various stages of the pipeline.

Working with Live Output Table

The streaming data in the pipeline appears in a live output table.

Hide/Unhide Columns

In the live output table, right-click columns and click **Hide** to hide that column 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 columns you select appear in the output table.

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.

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**.

Adding and Correlating Sources and References

You can correlate sources and references in a pipeline.

To add a correlating source or reference:

1. Open a pipeline in the **Pipeline Editor**.
2. Select the required query stage.
3. Click the **Sources** tab.
4. Click **Add a Source**.
5. Select a source (stream or reference) from the available list.
6. Click the **Window Area** in the source next to the clock icon and select appropriate values for **Range** and **Evaluation Frequency**.
7. Under **Correlation Conditions**, select **Match All** or **Match Any** as per your requirement. Then click **Add a Condition**.
8. Select the fields from the sources and the appropriate operator to correlate.

   Ensure that the fields you use on one correlation line are of compatible types. The fields that appear in the righ drop-down list depend on the field you select in the left drop-down list.
9. Repeat these steps for as many sources or references as you want to correlate.

Adding Filters

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

Adding Summaries

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.

Adding Group Bys

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.

   A group by is created on the selected column.

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 bys as well.

Using the Expression Builder

You can perform calculations on the data streaming in the pipeline using in-built functions of the Expression Builder.

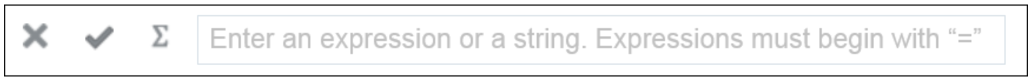
Oracle Stream Analytics supports various functions. For a list of supported functions, see Understanding Expression Builder Functions.

Note:

Currently, you can use expressions only within a query stage.

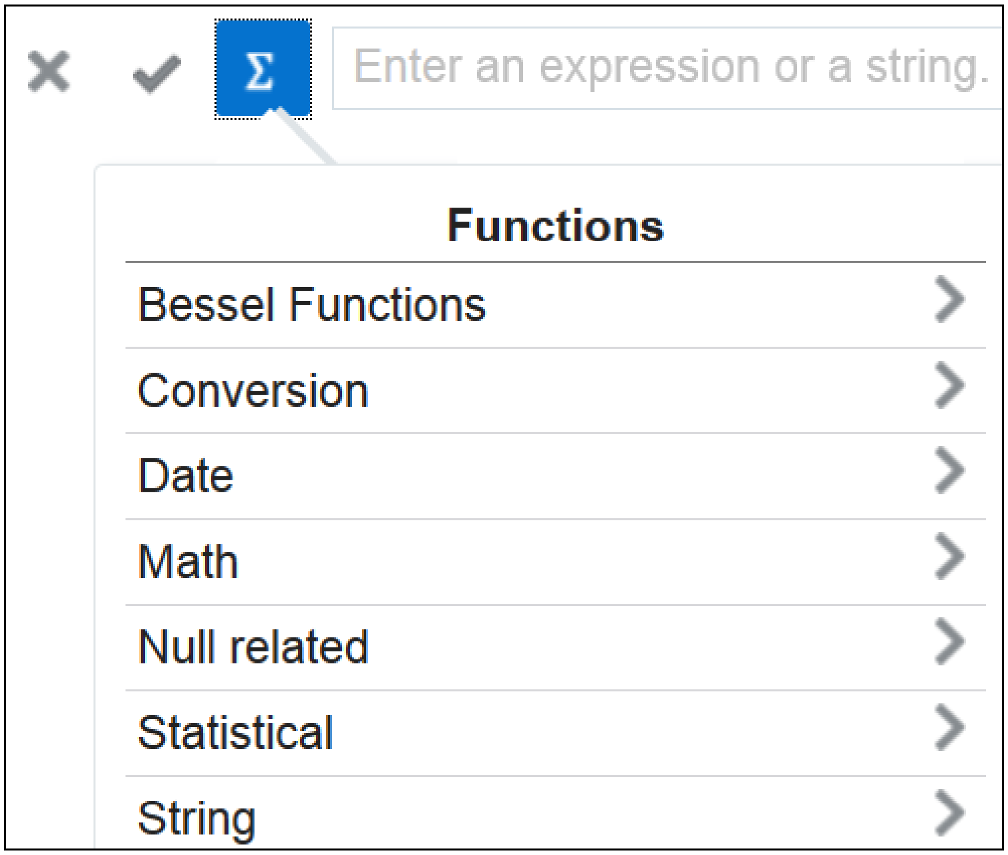
Adding a Constant Value Column

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.


Σ
Enter an expression or a string. Expressions must begin with "="

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.



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

Adding Visualizations

Visualizations are graphical representation of the streaming data in a pipeline. You can add visualizations on all stages in the pipeline except a target stage.

Select an appropriate visualization that suits your requirement.

Creating Visualization - Area Visualization

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.

Creating Visualization - Bar Visualization

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.

Creating Visualization - Bubble Visualization

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.

Creating Visualization - Geo Spatial Visualization

Geo Spatial visualization allows displaying location of an object on a geo fence and takes user to the area where events are occurring. User can configure visualization to specify latitude, longitude, identifier etc. Customization of visualization by specifying different pins like arrows with different colors based on certain condition is also allowed.

To add a geo spatial 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 **Geo Spatial**.
4. Enter/select values for the following fields:

   On the **Properties** tab:
   - **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.

- **Lat**: select the field that you want to use as the latitude. This is a mandatory field.
- **Long**: select the field that you want to use as the longitude. This is a mandatory field.
- **Key**: select the field that you want to use as the key. This is a mandatory field.

On the **Customizations** tab:

- Click the **Add** icon and select/enter appropriate values for **Field**, **Operator**, **Value**, and **Style**. This is an optional step.

5. Click **Create**.

The visualization is created and you can see the data populated in it.

Creating Visualization - Line Visualization

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.

Creating Visualization - Pie Visualization

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.

Creating Visualization - Scatter Visualization

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.

Creating Visualization - Stacked Bar Visualization

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.

Creating Visualization - 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.

Updating Visualizations

You can perform update operations like edit and delete on the visualizations after you add them.

You can open the visualization in a new window/tab using the **Maximize Visualizations** icon in the visualization canvas.

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.

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.

Adding a Pattern Stage

Patterns are templatized stages. You supply a few parameters for the template and a stage is generated based on the template.

For detailed information about the various type of patterns, see 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.

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.

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.

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

Adding a Query Group Stage

A query group stage allows you to use more than one query group to process your data - a stream or a table in memory. 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, such as a change data capture stream produced, to give just one example, 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.

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 Filters to understand the steps for creating filters.

   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 Summaries and Adding Group Bys for steps.
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 combined set of fields that get produced in the outgoing shape.
8. On the **Visualizations** tab, click **Add a Visualization** and add the required type of visualization. See Adding Visualizations for the procedure.

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.

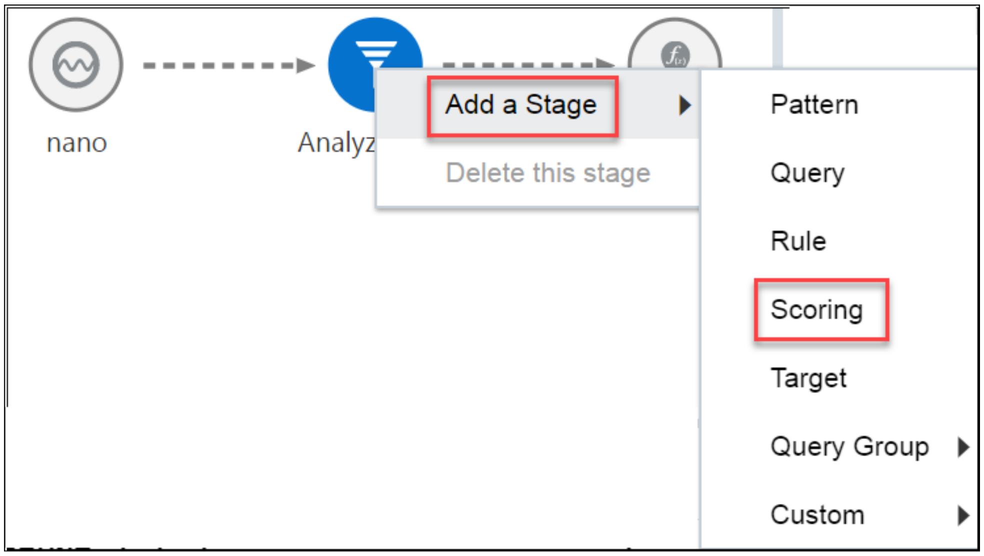
   You can see the table configuration that you have specified while creating the table stage in the **Table Configuration** tab.
7. On the **Input Filters** tab, click **Add a Filter**. See Adding Filters to understand the procedure.
8. On the **Groups** tab, click **Add a Group**. A group can consist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

   See Adding Summaries and Adding Group Bys for steps.
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 Visualizations for the procedure.

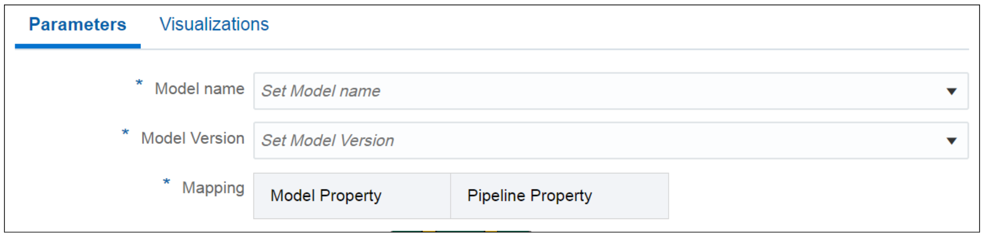
Adding a Scoring Stage

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 (Beta)**.



3. Enter a meaningful name and suitable description for the scoring stage and click **Save**.
4. In the stage editor, select appropriate values for the following:
   a. **Model name** — the predictive model that you want to use in the scoring stage
   b. **Model Version** — the version of the predictive model
   c. **Mapping** — the corresponding model fields that appropriately map to the stage fields



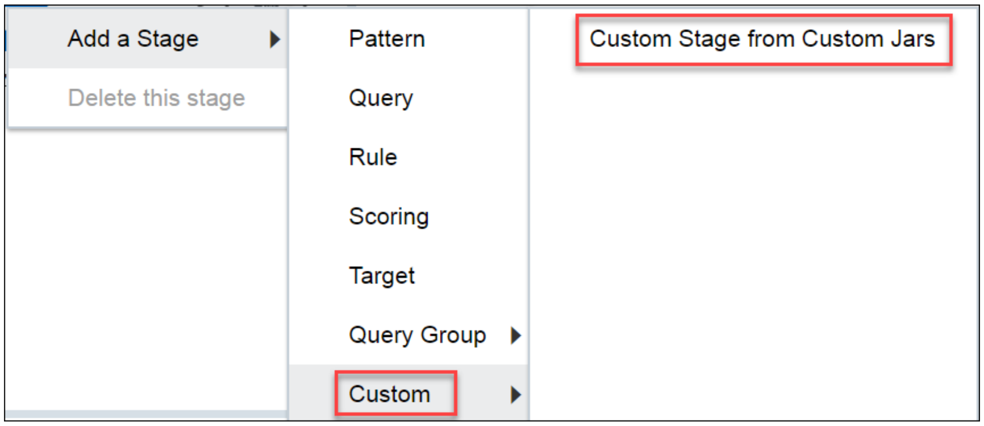
You can add multiple scoring stages based on your use case.

Adding a Custom Stage

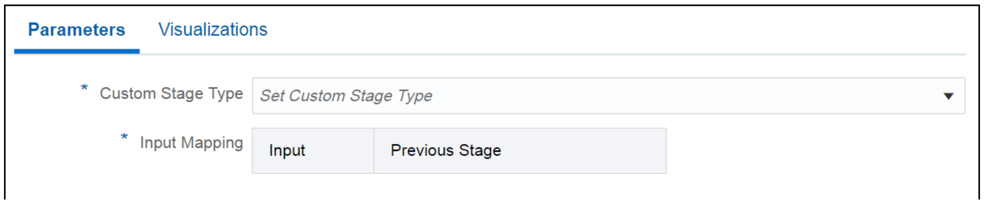
You can add filters in a pipeline to obtain more accurate streaming data.

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 meaningful name and suitable description for the scoring stage and click **Save**.
4. In the stage editor, select appropriate values for the following:
   a. **Custom Stage Type** — the custom stage that was previously installed though a custom jar
   b. **Input Mapping** — the corresponding column from the previous stage for every input parameter



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

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 scoring stage, click **Add a Stage**, and then select **Target**.
3. Enter a meaningful name and suitable description for the scoring stage and click **Save**.
4. In the stage editor, select a target that suits your requirement and start mapping the fields.
5. If the existing target does not suit your requirement of if there is no existing target, click **Create Target** to create a target.

Configuring a Target

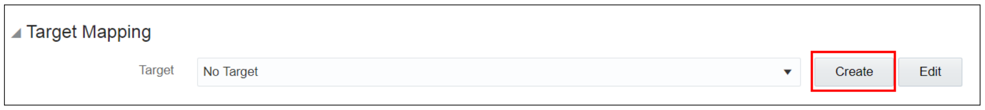
Target defines a destination for output data coming from a pipeline.

To configure a target:

1. Open a pipeline in the **Pipeline Editor**.

2. Select the target node in the pipeline.
3. Select a target for the pipeline from the drop-down list.
4. Map each of the **Target Property** and **Output Stream Property**.

You can also directly create the target from within the pipeline editor. See Creating a Target for the procedure. You can also edit an existing target.



The pipeline is configured with the specified target.

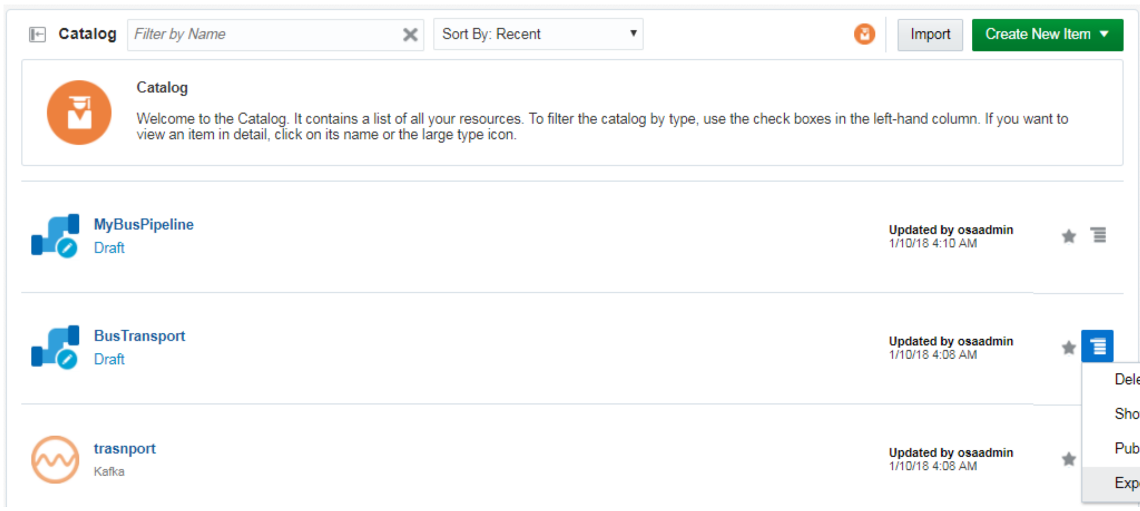
Exporting and Importing a Pipeline and Its Dependent Artifacts

The export and import feature lets you migrate your pipeline and its contents between Oracle Stream Analytics systems (such as development and production) in a matter of few clicks. 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**.

You can export and import pipelines and artifacts except for the following:

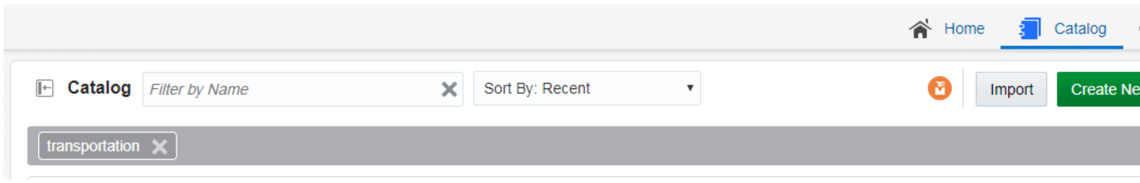
- Cubes
- Dashboards
- Custom Stages
- Visualizations
- File Streams
- Predictive Models

1. In your Oracle Stream Analytics instance, under **Catalog**, right-click the pipeline or artifact that you want to export to another instance of the Stream Analytics and then select **Export**.

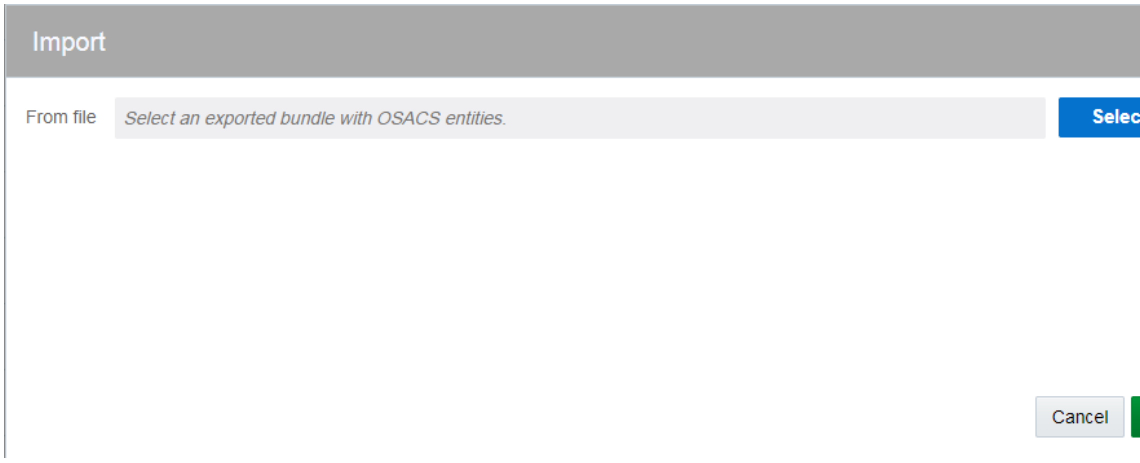


Your items are exported as a ZIP file.

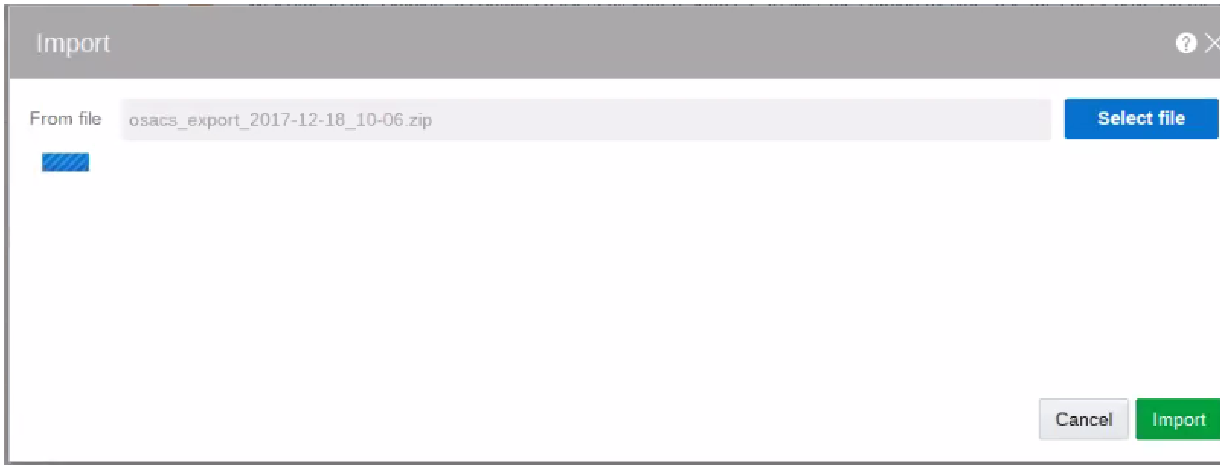
2. Go to the Oracle Stream Analytics instance to which you want to import the exported metadata.
3. On the toolbar, click **Import**.



4. In the **Import** dialog box, click **Select file** and then select the exported ZIP file.

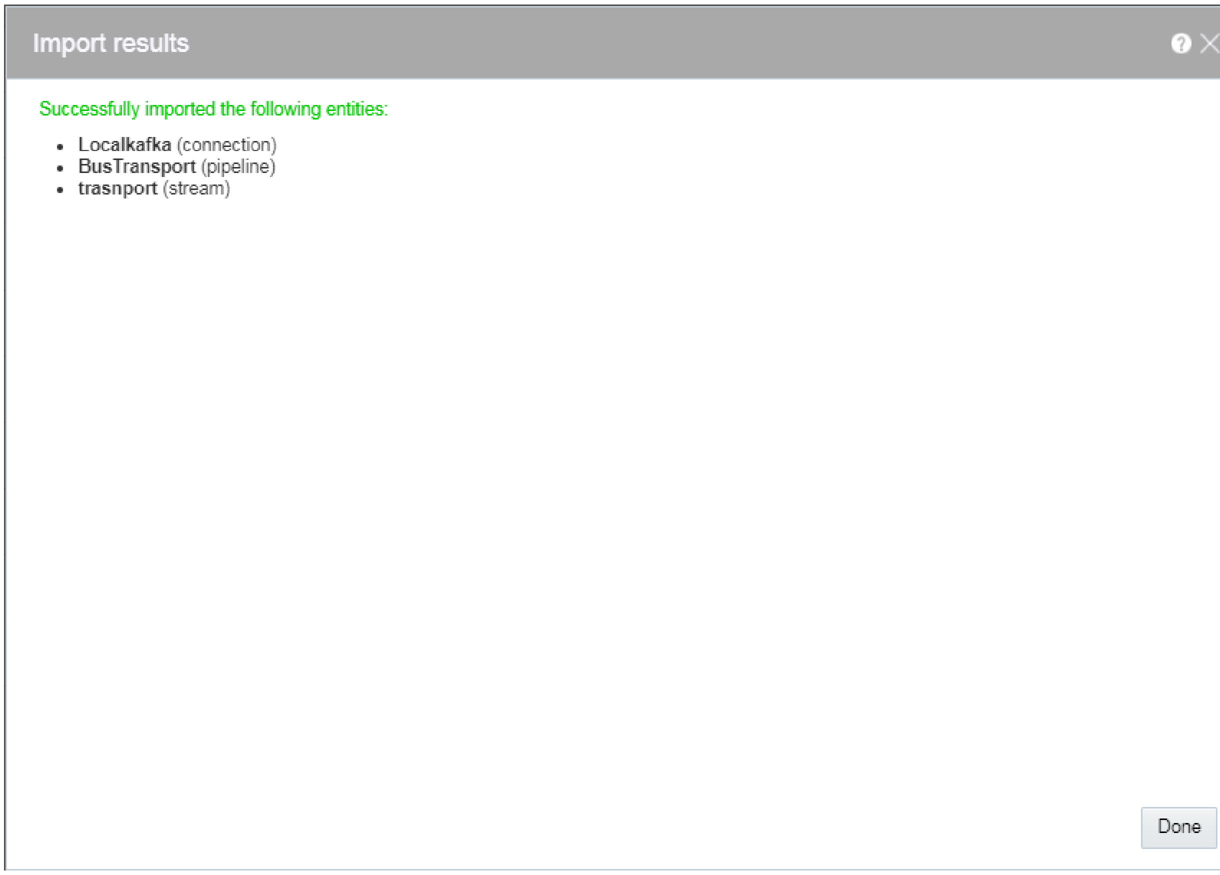


5. Click **Import**.



osacs_export_2017-12-18_10-06.zip

When the metadata is imported successfully, a message similar to the following appears:



Successfully imported the following entities:

- Localkafka (connection)
- BusTransport (pipeline)
- trasnport (stream)

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.

   > **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.

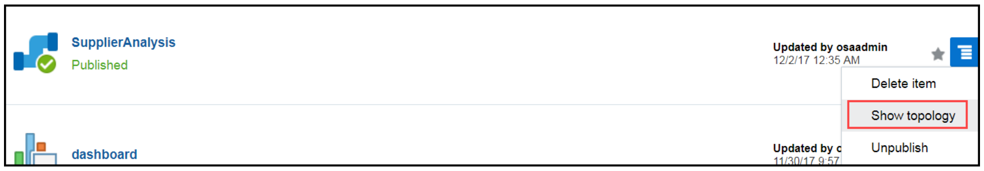
Using the Topology Viewer

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

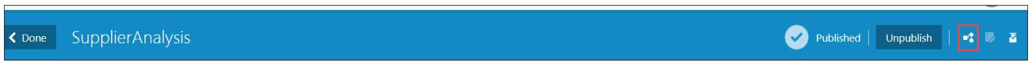
The topology viewer helps you in identifying the dependencies that a selected entity has on other entities. Understanding the dependencies helps you in being cautious while deleting or undeploying an entity. Oracle Stream Analytics supports two contexts for the topology — *Immediate Family* and *Extended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

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

- Select **Show topology** from the **Catalog Actions** menu 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.

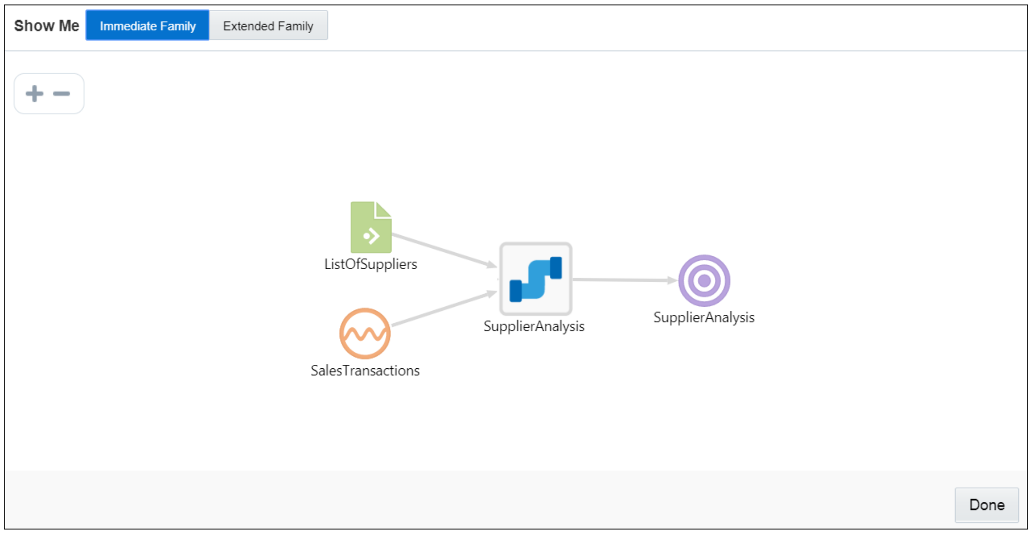
Note:

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.

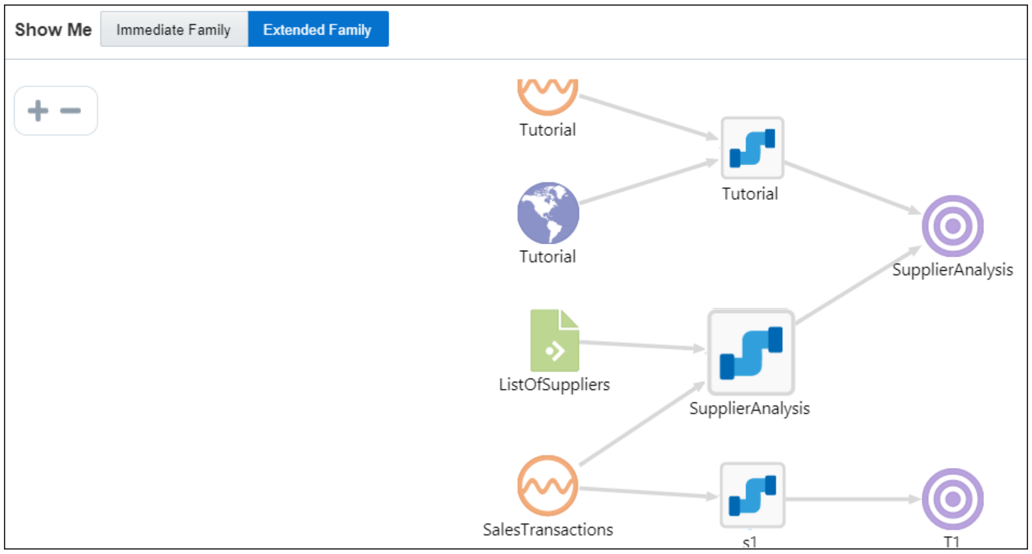
The following figure illustrates how a topology looks in the **Immediate Family**.



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**.



3

Working with Patterns

Patterns are a stage within a pipeline. When working from a pattern, you need to specify a few key fields to discover an interesting result. You can create pattern stages within the pipeline. Patterns are not stand-alone artifacts, they need to be embedded within a pipeline.

Topics:

About Oracle Stream Analytics Patterns

This topic applies only to Oracle user-managed services.

The visual representation of the event stream varies from one pattern type to another based on the key fields you choose.

Click **Patterns** on the Home page to see all the available patterns. Use the filters at left to view different categories of pattern. You can see full descriptions and learn more about each pattern by clicking the user assistant icon. Click again to hide the extra information.

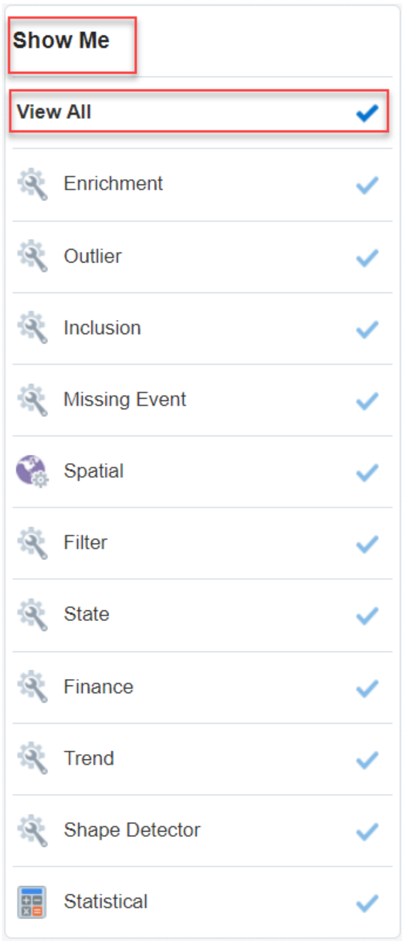
A *pattern* provides you with the results displayed in a live output stream based on common business scenarios.

Note:

While entering data in the fields for a specific pattern, ensure that the data you enter corresponds to the datatype of the field. If there is a mismatch between the entered data and the datatype, the pattern will not deploy and throw an error.

You can include or exclude patterns based on their categories using the **View All** link in the left panel under **Show Me**. When you click **View All**, a check mark appears next to it and all the patterns are displayed on the page.

When you want to display/view only a few/selective patterns, deselect **View All** and select the individual patterns. Only the selected patterns are shown in the catalog.



The following table lists the categories of patterns:

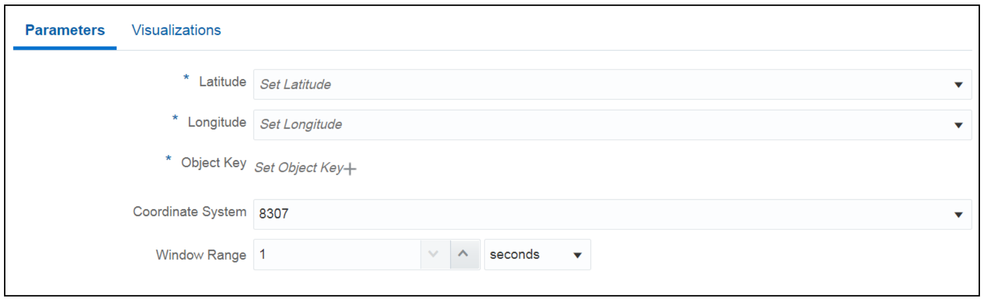
Category	Pattern
Enrichment	Reverse Geo Code: Near By Left Outer Join
Outlier	Fluctuation
Inclusion	Union Left Outer Join
Missing Event	'A' Not Followed by 'B' Detect Missing Event

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’
Finance	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
Shape Detector	Inverse W W
Statistical	Correlation Quantile

About the Spatial: Speed Pattern

Use this pattern to get the output average speed over the selected window range of a moving object.

For example, to analyze the average speed of a car.



Parameters	Visualizations
* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
Coordinate System	8307
Window Range	1 seconds

Provide suitable values for the following parameters:

- **Latitude**: the latitude of the location. Select a suitable value.
- **Longitude**: the longitude of the location. Select a suitable value.
- **Object Key**: select a suitable value for the object key.
- **Coordinate System**: the default value is 8307 and this is the only value supported.
- **Window Range**: a time range over which the speed is being calculated for an event. For example, if `window range=5 seconds` and object key is `phone no.`, then all the events with same `phone no.` received over last 5 seconds are used to calculate the average speed of that event.

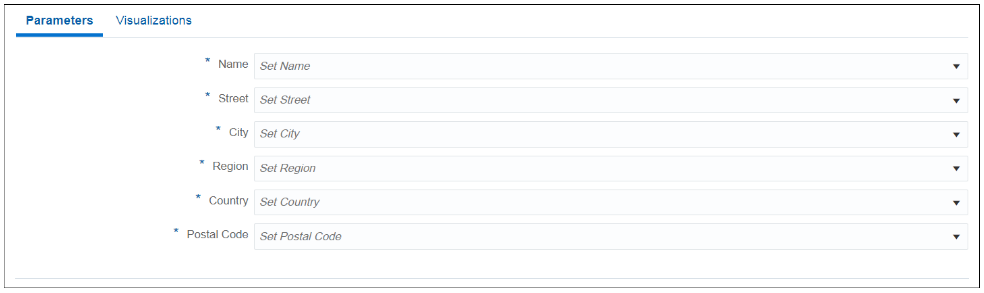
The outgoing shape contains **speed** as an added field along with the incoming fields. This is a numeric field, but the **speed** is measured in `meters per second`.

About the Geo Code Pattern

When analyzing data, you may encounter situations where you need to obtain the geographical code of a moving object based.

Use this pattern to get geographic coordinates (like latitude and longitude) for an address.

Make sure that you have set the proxy details in System Settings.



Parameters	Visualizations
* Name	Set Name
* Street	Set Street
* City	Set City
* Region	Set Region
* Country	Set Country
* Postal Code	Set Postal Code

Provide suitable values for the following parameters:

- **Name** — select a suitable value that can be uses as the place name. This is a mandatory parameter.
- **Street** — select a suitable value that can be used as the street name. This is a mandatory parameter.
- **City** — select a suitable value that can be used as the city name. This is a mandatory parameter.

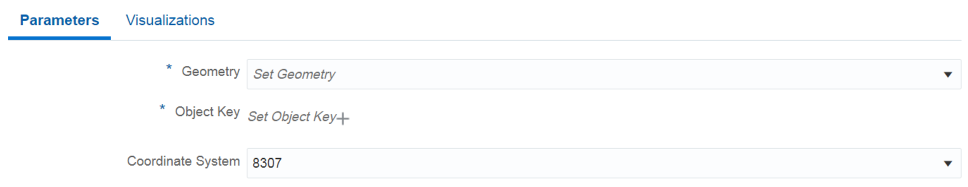
- **Region** — select a suitable value that can be used for the region. This is a mandatory parameter.
- **Country** — select a suitable value for the country. This is a mandatory parameter.
- **Postal Code** — select a suitable value for the postal code. This is a mandatory parameter.

The outgoing shape contains latitude and longitude as additional fields along with incoming fields.

About the Interaction: Single Stream Pattern

Use this 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.



* Geometry: Set Geometry
* Object Key: Set Object Key+
Coordinate System: 8307

Provide suitable values for the following parameters:

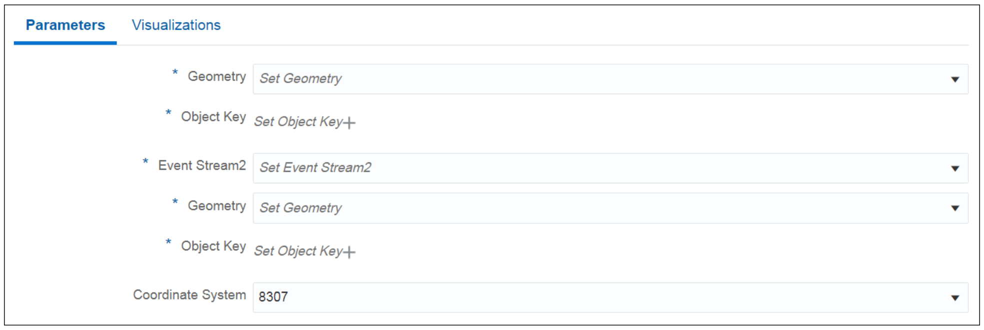
- **Geometry**: the field of type SDO_GEOMETRY data type. Only field of SDO_GEOMETRY data type should be chosen for such fields.
- **Object Key**: field used to uniquely identify object on the geo fence and is used for partitioning of data where supported.
- **Coordinate System**: the default value is 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.

About the Interaction: Two Stream Pattern

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*.

Use this pattern to get interaction of an object in one stream with objects in another stream.



* Geometry	Set Geometry
* Object Key	Set Object Key+
* Event Stream2	Set Event Stream2
* Geometry	Set Geometry
* Object Key	Set Object Key+
Coordinate System	8307

Provide suitable values for the following parameters:

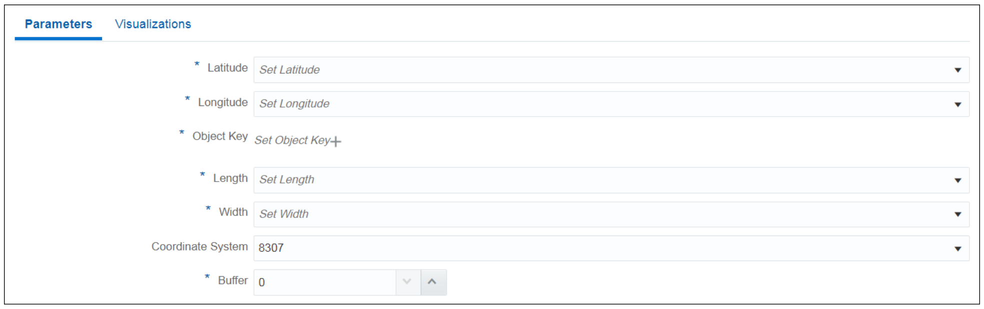
- **Geometry**: select a suitable value for geometry. This is a mandatory value.
- **Object Key**: select a suitable value for the object key. This is a mandatory value.
- **Event Stream 2**: select the second event stream. This is a mandatory value.
- **Geometry**: select a suitable value for geometry within the second stream. This is a mandatory value.
- **Object Key**: select a suitable value for the object key within the second stream. This is a mandatory value.
- **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.

About the Spatial: Point to Polygon Pattern

Use this pattern to get an object shape based on geographical coordinates, fixed length and breadth of and object.

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.



* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
* Length	Set Length
* Width	Set Width
Coordinate System	8307
* Buffer	0

Provide suitable values for the following parameters:

- **Latitude**: select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude**: select a suitable value for the longitude. This is a mandatory parameter.

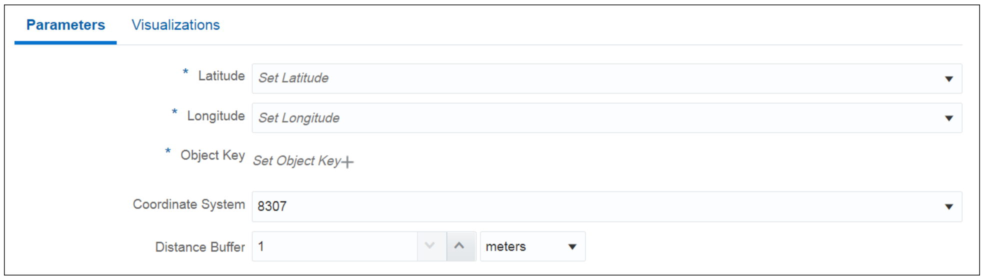
- **Object Key**: select a suitable value for the object key. This is a mandatory parameter.
- **Length**: select a suitable field to be used as the reference length of the polygon.
- **Width**: select a suitable field to be used as the reference width of the polygon.
- **Coordinate System**: the default 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`).

About the Proximity: Single Stream Pattern

Use this 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.



* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key
Coordinate System	8307
Distance Buffer	1 meters

Provide suitable values for the following parameters:

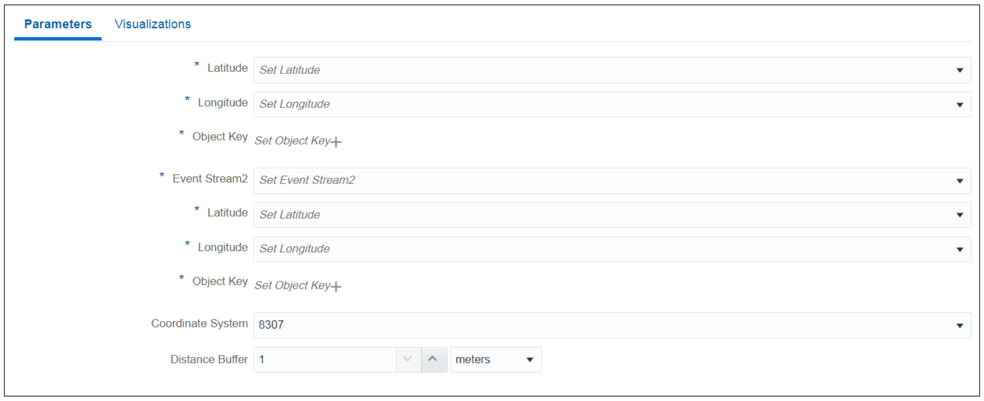
- **Latitude**: select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude**: select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key**: select a suitable value for the object key. This is a mandatory parameter.
- **Coordinate System**: the default value is 8307 and this is the only value supported.
- **Distance Buffer**: enter a proximity value for the distance buffer. This is the distance that two points can be apart and still be considered the same. 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.

About the Proximity: Two Stream Pattern

Use use this pattern to get the proximity between objects of two streams.

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'.



* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
* Event Stream2	Set Event Stream2
* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
Coordinate System	8307
Distance Buffer	1 meters

Provide suitable values for the following parameters:

- **Latitude**: select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude**: select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key**: select a suitable value for the object key. This is a mandatory value.
- **Event Stream 2**: select the second event stream. This is a mandatory value.
- **Latitude**: select a suitable value for the latitude in the second stream. This is a mandatory parameter.
- **Longitude**: select a suitable value for the longitude in the second stream. This is a mandatory parameter.
- **Object Key**: select a suitable value for the object key in the second stream. This is a mandatory parameter.
- **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:

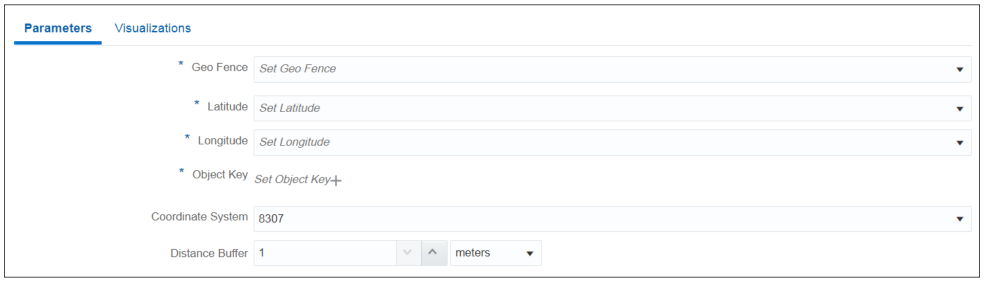
When a pipeline with this pattern has a database reference with cache enabled, the pattern does not display any output in the live output stream.

The outgoing shape displays **distance** as another column, which is the distance between two object under consideration along with the incoming shape.

About the Proximity: Stream with Geo Fence Pattern

Use this 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.



* Geo Fence	Set Geo Fence
* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
Coordinate System	8307
Distance Buffer	1 meters

Provide suitable values for the following parameters:

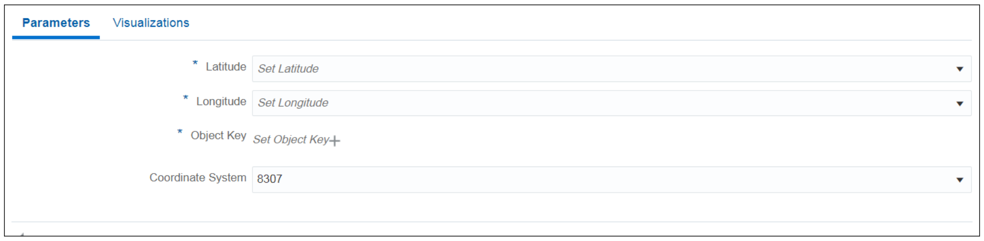
- **Geo Fence**: select a geo fence that you like to analyze.
- **Latitude**: select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude**: select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key**: select a suitable value for the object key. This is a mandatory parameter.
- **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.

About the Direction Pattern

Use this pattern to get the direction of a moving object.

For example, you can evaluate the direction of a moving truck.



* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
Coordinate System	8307

Provide suitable values for the following parameters:

- **Latitude**: select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude**: select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key**: select a suitable value for the object key. This is a mandatory parameter.
- **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.

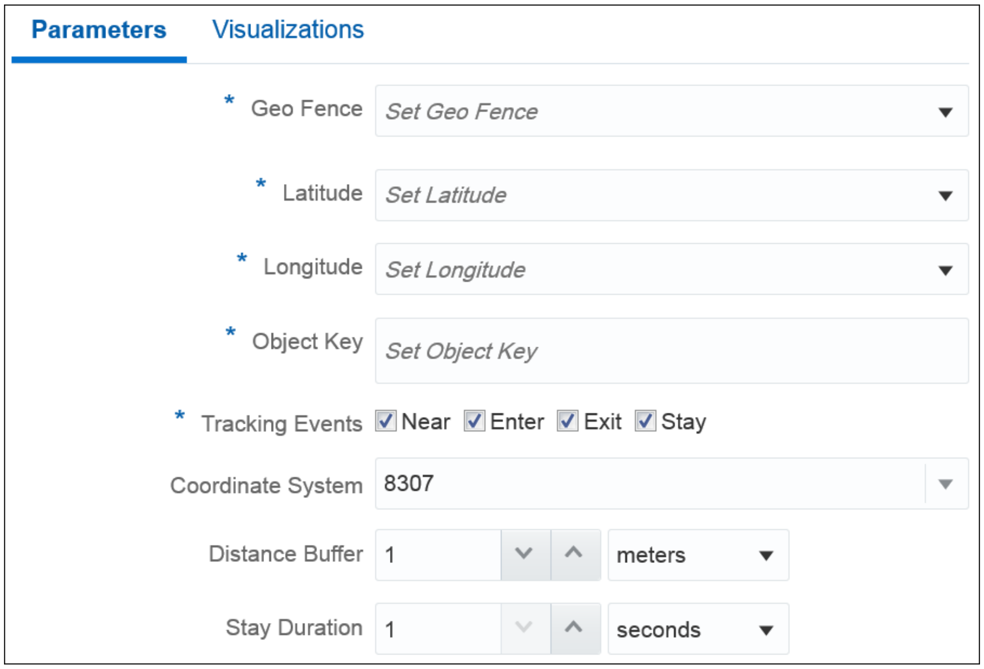
About the Geo Fence Pattern

Use this pattern when you want to track object relation with a virtual boundary called goe 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.

The geo-location can be:

- Near to Geo Fence
- Exiting Geo Fence
- Staying within Geo Fence for a specified duration
- Entering Geo Fence



* Geo Fence	Set Geo Fence
* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key
* Tracking Events	✓ Near ✓ Enter ✓ Exit ✓ Stay
Coordinate System	8307
Distance Buffer	1 meters
Stay Duration	1 seconds

Provide suitable values for the following parameters:

- Geo Fence
- Latitude
- Longitude

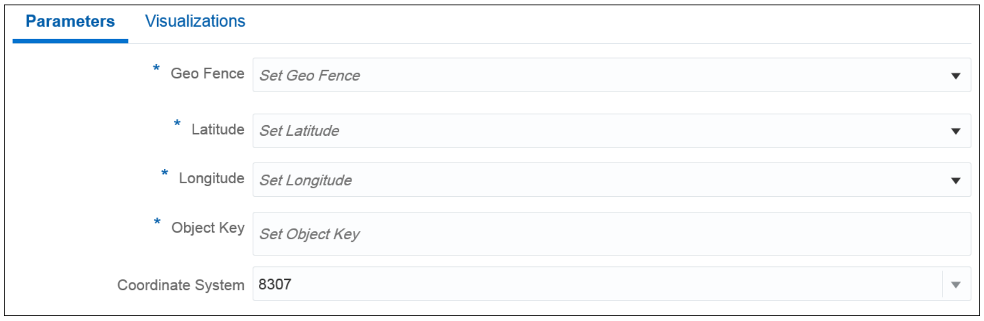
- Object Key
- Tracking Events
  - Near
  - Enter
  - Exit
  - Stay
- Coordinate system
- Distance Buffer — this parameter is enabled only if you select **Near** option in **Tracking Events**. This field acts as a buffer for filtering results. Only those events or objects which are within the specified 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 and this duration acts as 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.

About the Geo Filter Pattern

Use this pattern to when you want to track objects inside a virtual boundary.

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.



* Geo Fence	Set Geo Fence
* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key
Coordinate System	8307

Provide suitable values for the following parameters:

- **Geo Fence** — select one of the existing geo fences to analyze. This is a mandatory field.
- **Latitude** — select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude** — select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key** — select a suitable value for the object key. This field acts as a partitioning criteria and also used to uniquely identify objects. This is a mandatory parameter.

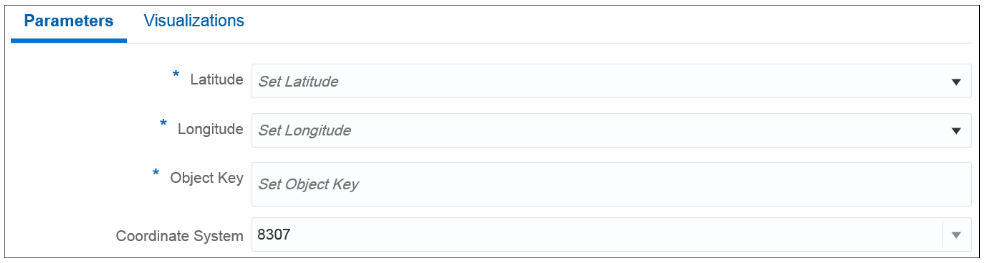
- **Coordinate System** — the default value is 8307 and this is the only value supported.

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.

About the Reverse Geo Code: Near By Pattern

Use this to obtain nearest place for the specified geographical coordinates.

Make sure that you have set the proxy details in System Settings.



* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key
Coordinate System	8307

Provide suitable values for the following parameters:

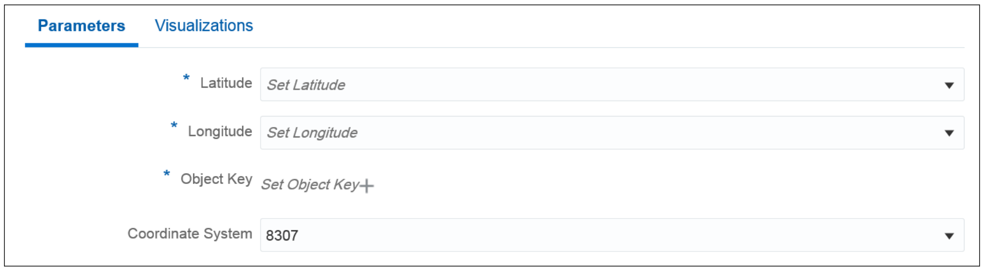
- **Latitude** — select a suitable value for the latitude. This is a mandatory parameter.
- **Longitude** — select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key** — select a suitable value for the object key. This is a mandatory parameter.
- **Coordinate system** — 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.

About the Reverse Geo Code: Near By Place Pattern

Use this pattern to obtain the near by location with granular information like city, country, street etc. for the specified latitude and longitude.

Make sure that you have set the proxy details in System Settings.



* Latitude	Set Latitude
* Longitude	Set Longitude
* Object Key	Set Object Key+
Coordinate System	8307

Provide suitable values for the following parameters:

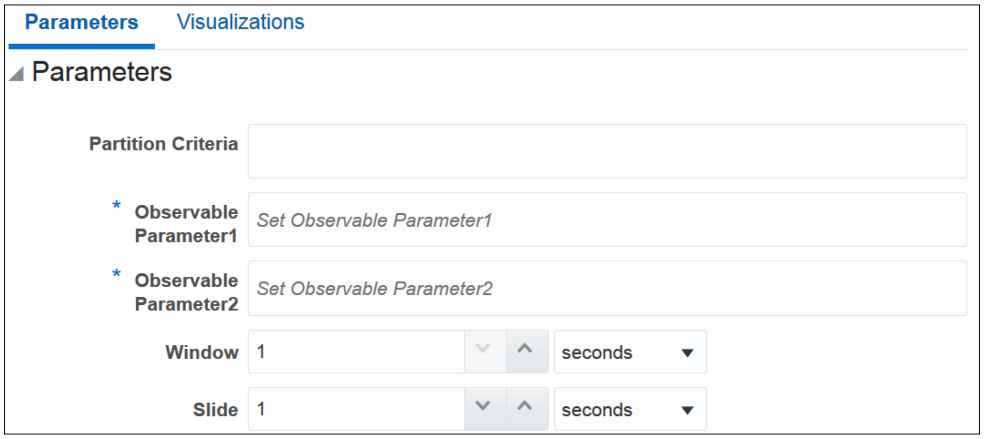
- **Latitude** — select a suitable value for the latitude. This is a mandatory parameter.

- **Longitude** — select a suitable value for the longitude. This is a mandatory parameter.
- **Object Key** — select a suitable value for the object key. This is a mandatory parameter.
- **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`.

About the Correlation Pattern

Use this pattern if you need to identify correlation between two numeric parameters.



Partition Criteria	
* Observable Parameter1	Set Observable Parameter1
* Observable Parameter2	Set Observable Parameter2
Window	1 seconds
Slide	1 seconds

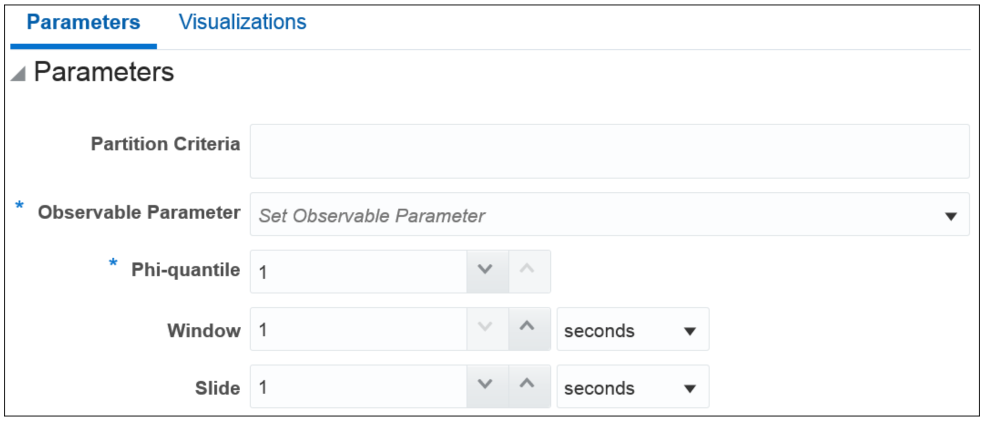
Provide suitable values for the following parameters:

- **Partition Criteria**: the field by which you want to partition.
- **Observable Parameter 1**: the first value used to identify the correlation.
- **Observable Parameter 2**: the second value used to identify the correlation.
- **Window**: a rolling time period, the duration within which the correlation is identified.
- **Slide**: the frequency at which you want to refresh the data.

The outgoing shape is same as the incoming shape.

About the Quantile Pattern

Use this pattern if you need to calculate the value of quantile function.



Partition Criteria

* Observable Parameter: Set Observable Parameter

* Phi-quantile: 1

Window: 1 seconds

Slide: 1 seconds

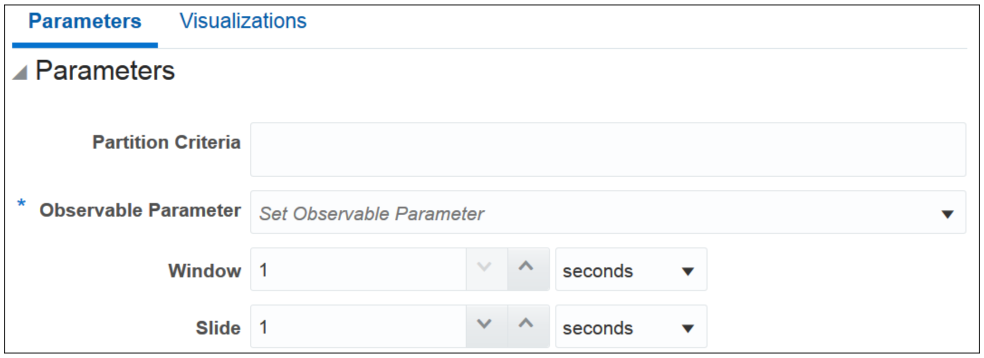
Provide suitable values for the following parameters:

- **Partition Criteria**: the field based on which you want to partition.
- **Observable Parameter**: the observable to calculate the quantile.
- **Phi-quantile**: the value is used to calculate the quantile of the selected event stream.
- **Window**: a rolling time period, within which the events will be collected and quantile is calculated.
- **Slide**: how frequent newly updated output will be pushed downstream and into the browser.

The outgoing shape is the same as the incoming shape.

About the Standard Deviation Pattern

Use this pattern to calculate the standard deviation of the selected values with the expected values.



Partition Criteria

* Observable Parameter: Set Observable Parameter

Window: 1 seconds

Slide: 1 seconds

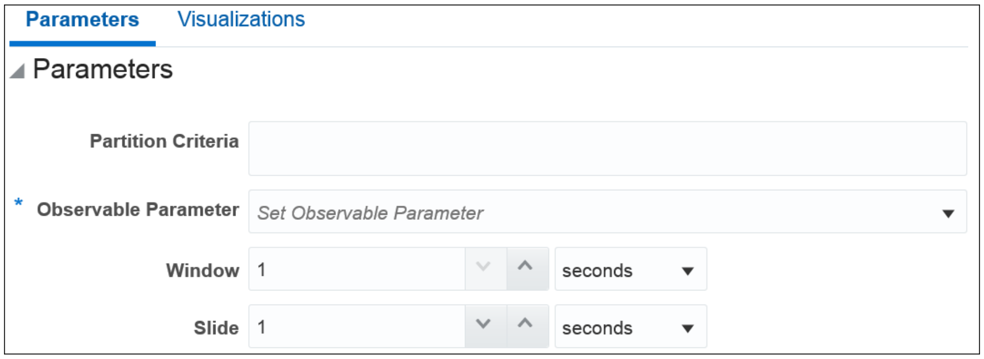
Provide suitable values for the following parameters:

- **Partition Criteria**: the field based on which you want to partition.
- **Observable Parameter** : the value used to identify the standard deviation.
- **Window**: a rolling time period, the duration within which the standard deviation is identified.
- **Slide**: the frequency at which you want to refresh the data.

The outgoing shape is the same as the incoming shape.

About the Median Pattern

Use this pattern to calculate the median of an event stream with respect to a specific parameter.



Partition Criteria

* Observable Parameter: Set Observable Parameter

Window: 1 seconds

Slide: 1 seconds

Provide suitable values for the following parameters:

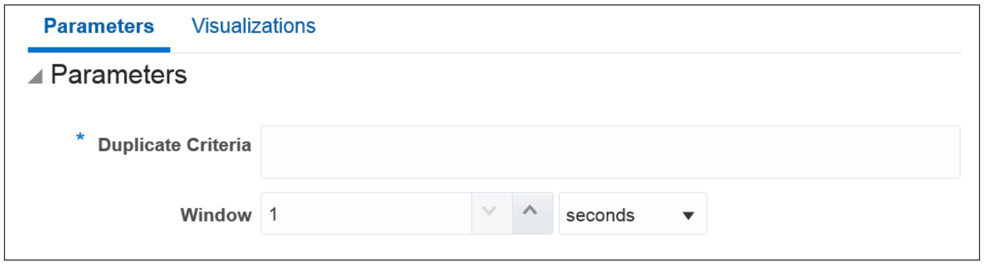
- **Partition Criteria**: the field based on which you want to partition.
- **Observable Parameter**: the observable to calculate the median.
- **Window**: a rolling time period, within which the events will be collected and median is calculated.
- **Slide**: how frequent newly updated output will be pushed downstream and into the browser.

The outgoing shape is same as the incoming shape.

About the Detect Duplicates Pattern

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.

Use this pattern to understand how many duplicate events your stream has. For example, when you suspect that your aggregates are offset, you may want to check your stream for duplicate events.



* Duplicate Criteria

Window: 1 seconds

Provide suitable values for the following parameters:

- **Duplicate Criteria**: a list of fields, whose values will be compared between events to look for identical values. If all the configured fields have identical values, the incoming event will be considered a duplicate and an outgoing event will be fired.

- **Window**: a time period, within which the duplicates will be searched for. For example, if you set the window to 10 seconds, a duplicate event that arrives 9 seconds after the first one will be trigger an outgoing event, while a duplicate event that arrives 11 seconds after the first one will not do so.

Outgoing Shape

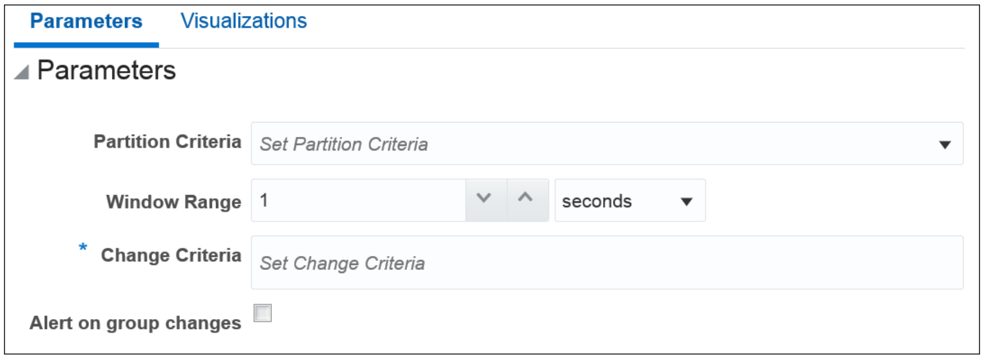
The outgoing shape is the same as the incoming shape with one extra field: `Number_of_Duplicates`. This extra field will carry the number of duplicate events that have been discovered. All the other fields will have values of the last duplicate event.

About the Change Detector Pattern

The Change Detector pattern looks for changes in the values of your event fields and report the changes once they occur within a specified range window. For example, and events arrives with value `value1` for field `field1`. If 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.

Use it when you need to be aware of changes in a normally stable value. 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.



Partition Criteria	Set Partition Criteria
Window Range	1 seconds
* Change Criteria	Set Change Criteria
Alert on group changes	☐

Provide suitable values for the following parameters:

- **Partition Criteria**: a field to partition your stream by. 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 range**: 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**: a list of fields, whose values will be compared between events to look for changes. If the fields contain no changes, no alerts will be generated.

- **Alert on group changes**: this parameter is responsible for the 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. 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 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:

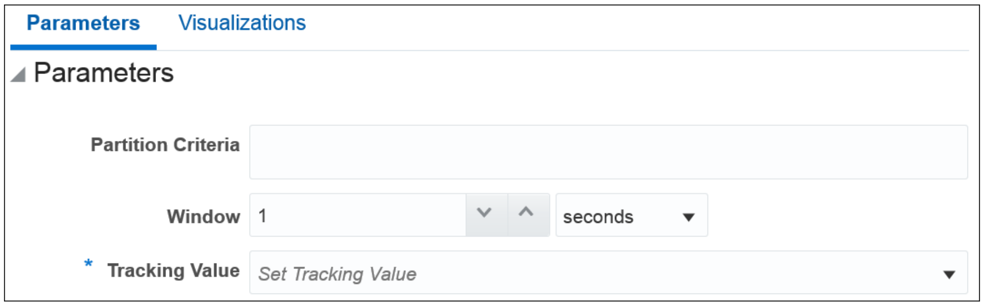
- `sensor_id`
- `temperature`
- `orig_temperature`
- `pressure`
- `orig_pressure`
- `location`
- `orig_location`

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.

About the W Pattern

The W pattern, also known as a double bottom chart pattern, is used in the technical analysis of financial trading markets.

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.



Partition Criteria

Window 1 seconds

* Tracking Value Set Tracking Value

Provide suitable values for the following parameters:

- **Partition Criteria**: a field to partition your stream by. For example, a ticker symbol.
- **Window**: a time period, within which the values of the designated field are analyzed for the W shape.
- **Tracking value**: a field, whose values are 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:

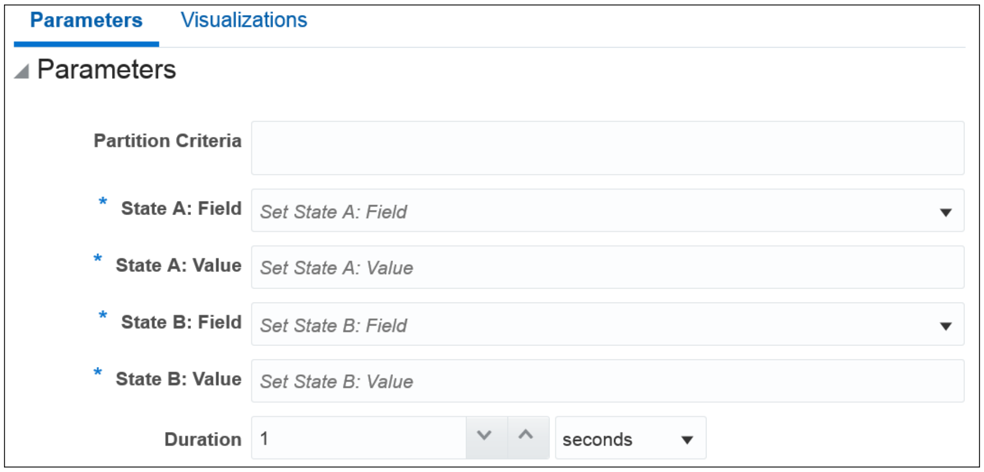
- `firstW`
- `firstValleyW`
- `headW`
- `secondValleyW`
- `lastW`

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.

About the ‘A’ Followed by ‘B’ Pattern

The 'A' Followed by 'B' pattern looks for particular events following one another and will 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 you flow. For example, if an order status `BOOKED` is followed by an order status `SHIPPED` (skipping status `PAID`), you need to raise an alert.



Partition Criteria	
* State A: Field	Set State A: Field
* State A: Value	Set State A: Value
* State B: Field	Set State B: Field
* State B: Value	Set State B: Value
Duration	1 seconds

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 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`

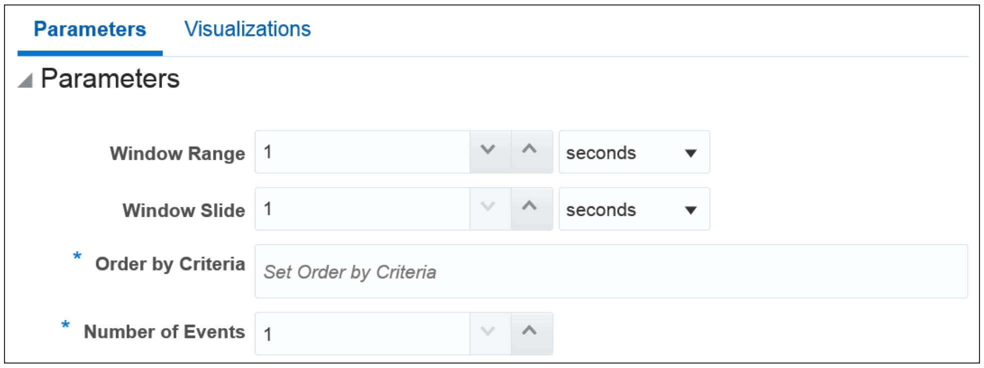
You will get the following outgoing shape:

- `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)

About the Top N Pattern

The Top N pattern will output N events with highest values from a collection of events arriving within a specified time window sorted not in the default order of arrival but the way you specify.

Use it to get the highest values of fields in your stream within a specified time window. For example, use it to get N highest values of pressure sensor readings.



Window Range	1	seconds
Window Slide	1	seconds
* Order by Criteria	Set Order by Criteria	
* Number of Events	1	

Provide suitable values for the following parameters:

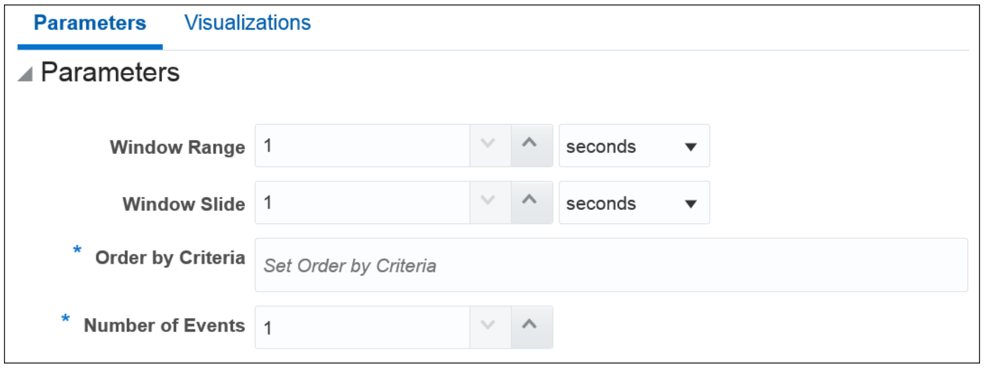
- **Window Range**: a rolling time period, within which the events will be collected and ordered per your ordering criteria.
- **Window Slide**: how frequent newly updated output will be pushed downstream and into the browser.
- **Order by Criteria**: a list of fields to use to order the collection of events.
- **Number of Events**: a number of top value events to output.

The outgoing shape is the same as the incoming shape.

About the Bottom N Pattern

The Bottom N pattern will output N events with lowest values from a collection of events arriving within a specified time window sorted not in the default order of arrival but the way you specify.

Use it to get the lowest values of fields in your stream within a specified time window. For example, use it to get N lowest values of pressure sensor readings.



Window Range	1	seconds
Window Slide	1	seconds
* Order by Criteria	Set Order by Criteria	
* Number of Events	1	

Provide suitable values for the following parameters:

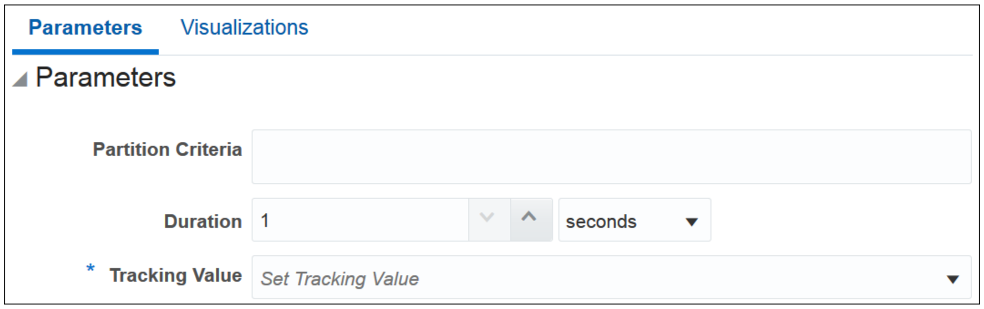
- **Window Range**: a rolling time period, within which the events will be collected and ordered per your ordering criteria.
- **Window Slide**: how frequent newly updated output will be pushed downstream and into the browser.
- **Order by Criteria**: a list of fields to use to order the collection of events.
- **Number of Events**: a number of bottom value events to output.

The outgoing shape is the same as the incoming shape.

About the Up Trend Pattern

The Up Trend pattern detects a situation when a numeric value goes invariably up 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.



Partition Criteria

Duration 1 seconds

* Tracking Value *Set Tracking Value*

Provide suitable values for the following parameters:

- **Partition Criteria**: a field by which to partition your stream. 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**: a time period, within which the values of the designated field are analyzed for the upward trend.
- **Tracking value**: a field, whose values are analyzed for the 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`

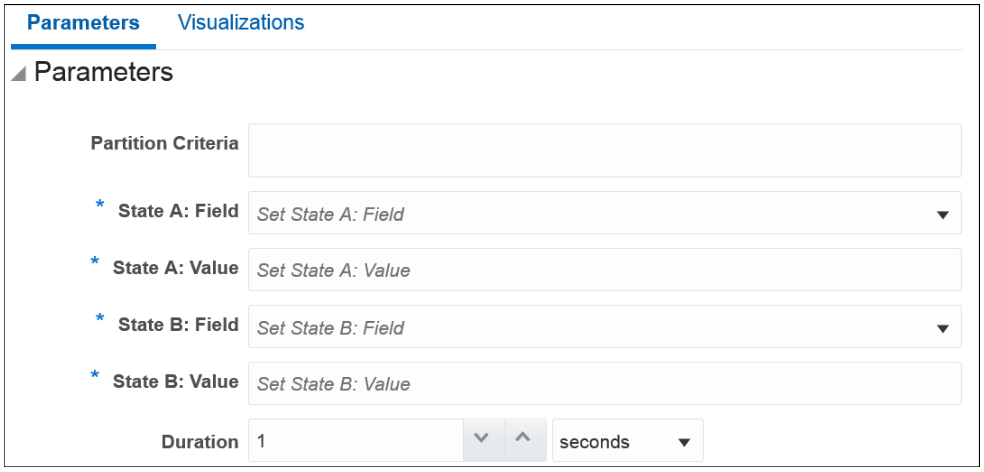
Normally, you would use `sensor_id` to partition your data and say you want to look for the upward trend in temperature. So, select `sensor_id` in the partition criteria parameter and temperature in the tracking value parameter. Use a duration that fits your use case. In this scenario, you will have 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)

About the ‘A’ Not Followed by ‘B’ Pattern

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.



Parameters

Partition Criteria	
* State A: Field	Set State A: Field
* State A: Value	Set State A: Value
* State B: Field	Set State B: Field
* State B: Value	Set State B: Value
Duration	1 seconds

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.

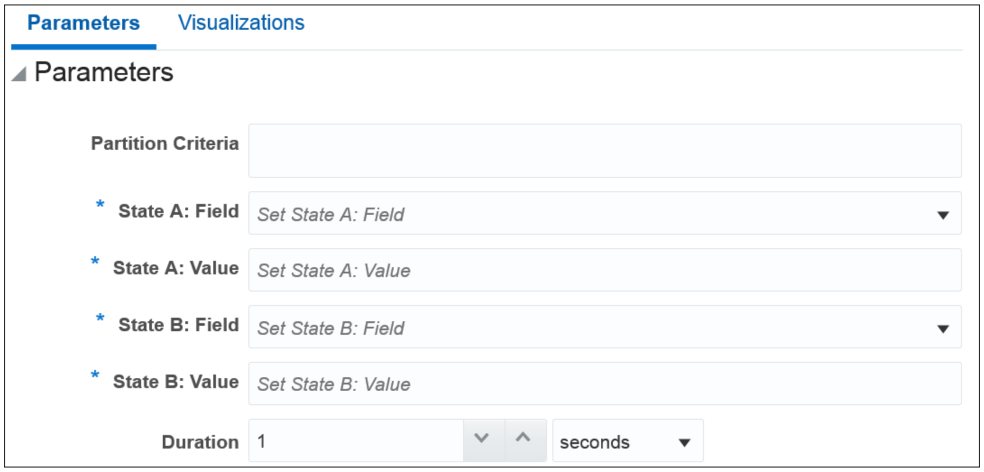
Outgoing Shape

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.

About the ‘B’ Not Preceded by ‘A’ Pattern

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.



Partition Criteria	
* State A: Field	Set State A: Field
* State A: Value	Set State A: Value
* State B: Field	Set State B: Field
* State B: Value	Set State B: Value
Duration	1 seconds

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.

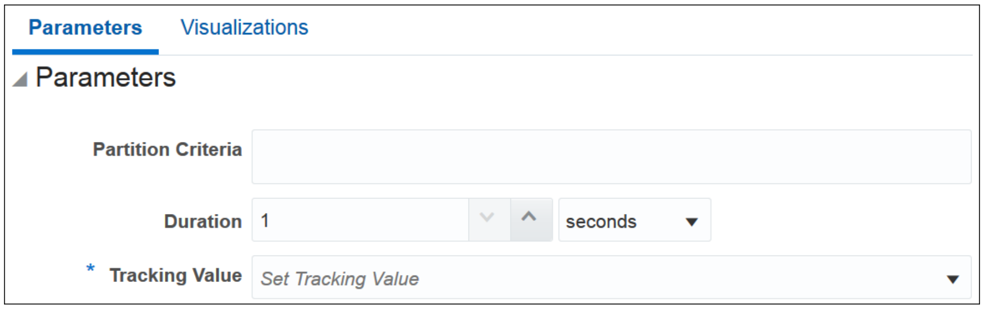
Outgoing Shape

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.

About the Down Trend Pattern

The Down Trend pattern detects a situation when a numeric value goes invariably down over a period of time.

Use this pattern if you need to detect situations of a constant reduction in one of your numeric values. For example, detect a constant drop in pressure from one of your sensors.



Partition Criteria	
Duration	1 seconds
* Tracking Value	Set Tracking Value

Provide suitable values for the following parameters:

- **Partition Criteria**: a field to partition your stream by. 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**: a time period, within which the values of the designated field are analyzed for the downward trend.
- **Tracking value**: a field, whose values are 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:

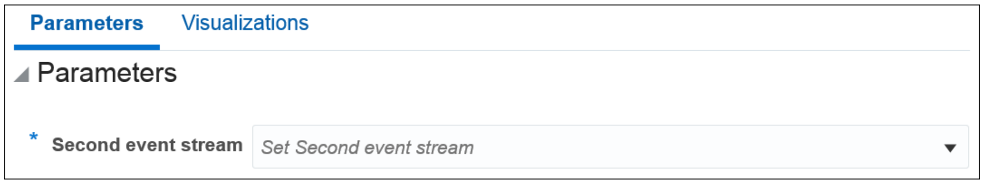
- `sensor_id`
- `startValue` (this is the value of temperature that starts the trend)
- `endValue` (this is the value of temperature that ends the trend)
- `temperature` (the value of the last event)
- `pressure` (the value of the last event)
- `location` (the value of the last event)

The pattern is visually represented based on the data you have entered/selected.

About the Union Pattern

The Union pattern merges two streams with identical shapes into one.

Use this pattern if you have two streams with identical shapes that you want to merge into one, for example when you have two similar sensors sending data into two different streams, and you want to process the streams simultaneously, in one pipeline.



* Second event stream: Set Second event stream

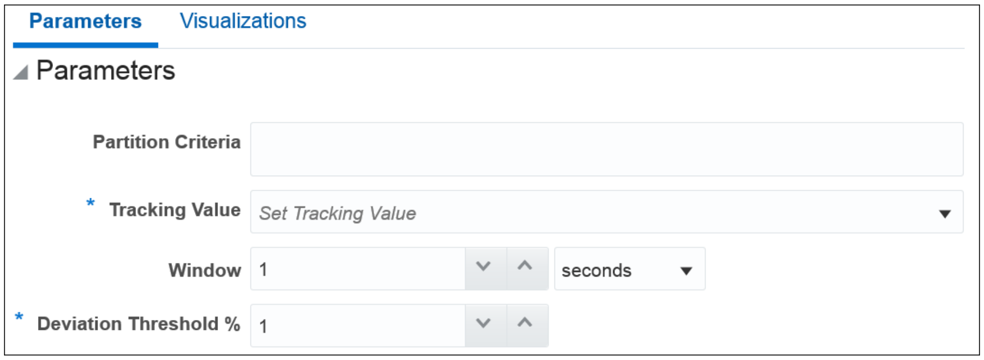
Provide suitable values for the following parameters:

- **Second event stream**: 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.

About the Fluctuation Pattern

Use this 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.



Partition Criteria	
* Tracking Value	Set Tracking Value
Window	1 seconds
* Deviation Threshold %	1

Provide suitable values for the following parameters:

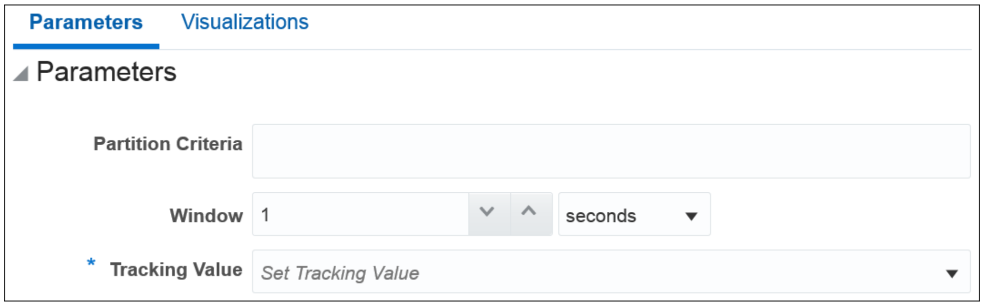
- **Partition Criteria**: the field based on which you want to partition.
- **Tracking Value**: the value is used to track the event data and create a pattern in the live output stream.
- **Window**: a rolling time period, the frequency at which you want to refresh the data.
- **Deviation Threshold %**: value indicates 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.

About the Inverse W Pattern

The Inverse W pattern, also known as a double top chart pattern, is used in the technical analysis of financial trading markets.

Use this pattern when you want to see the financial data in a graphical form.



Partition Criteria	
Window	1 seconds
* Tracking Value	Set Tracking Value

Provide suitable values for the following parameters:

- **Partition Criteria**: a field to partition your stream by. For example, a ticker symbol.
- **Window**: a time period, within which the values of the designated field are analyzed for the inverse W shape.
- **Tracking value**: a field, whose values are 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:

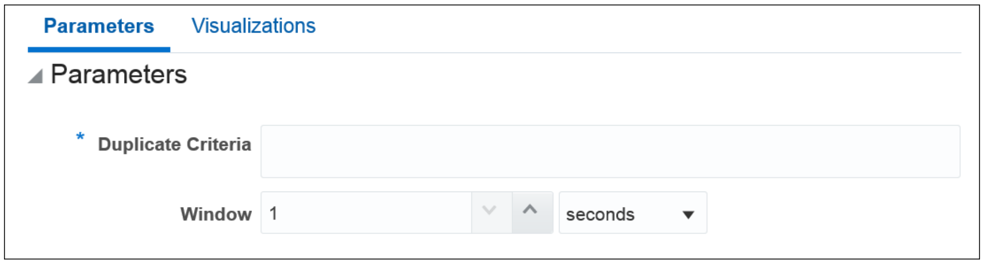
- `firstW`
- `firstPeakW`
- `headInverseW`
- `secondpeakW`
- `lastW`

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.

About the Eliminate Duplicates Pattern

The Eliminate Duplicates pattern looks for duplicate events in your stream within a specified time window and removes 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. It is up to you to specify what fields are analyzed for duplicate values. You can configure the pattern to compare just one field or the whole event.

Use it to get rid of noise in your stream. If you know that your stream contains duplicates that might offset your aggregates, such as counts, use the Eliminate Duplicates pattern to cleanse your data.



* Duplicate Criteria	
Window	1 seconds

Provide suitable values for the following parameters:

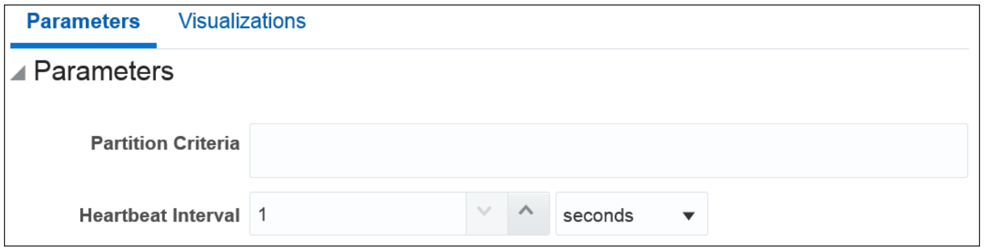
- **Duplicate Criteria**: a list of fields, whose values will be compared between events to look for identical values. If all the configured fields have identical values, the second, third, and subsequent events will be dropped.
- **Window**: a time period, within which the duplicates will be discarded. 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.

About the Detect Missing Event Pattern

The Detect Missing Event pattern discovers simple situations when an expected event is missing.

Use this pattern if you need to detect missing events in your feed. For example, you have a feed when multiple sensors send their readings every 5 seconds. Use this pattern to detect sensors that have stopped sending their readings, which may indicate that the sensor is broken or there is no connection to the sensor.



Partition Criteria

Heartbeat Interval 1 seconds

Provide suitable values for the following parameters:

- **Partition Criteria**: a field to partition your stream by. 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.
- **Heartbeat interval**: a time period, within which missing events are detected. If there is no event from a sensor within the heartbeat interval after the last event, an alert is triggered.

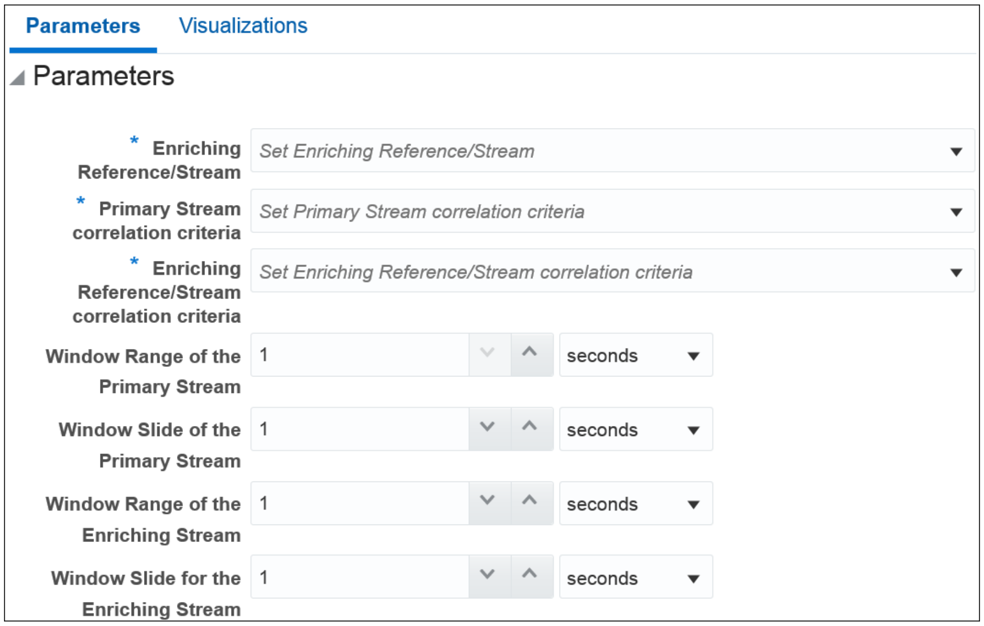
Outgoing Shape

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.

About the Left Outer Join Pattern

The Left Outer join pattern joins your flow with another stream or a reference using the left outer join semantics.

Use this 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.



Parameters

* Enriching Reference/Stream	Set Enriching Reference/Stream	
* Primary Stream correlation criteria	Set Primary Stream correlation criteria	
* Enriching Reference/Stream correlation criteria	Set Enriching Reference/Stream correlation criteria	
Window Range of the Primary Stream	1	seconds
Window Slide of the Primary Stream	1	seconds
Window Range of the Enriching Stream	1	seconds
Window Slide for the Enriching Stream	1	seconds

Provide suitable values for the following parameters:

- **Enriching Reference/Stream:** the stream or reference you want to join to your flow.
- **Correlation Criteria**: fields on which the stream / reference will be joined.
- **Window Range of the Primary Stream**: a rolling time window used to make a collection of events in your primary flow to be joined with the enriching stream / reference.
- **Window Slide of the Primary Stream**: how often the data will be pushed downstream and to the UI.
- **Window Range of the Enriching Stream**: a rolling time window used 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**: how often the data will be pushed downstream and to the UI. Disabled, if a reference is used.

The outgoing shape is a sum of two incoming shapes.

Creating a Pipeline Using a Pattern

Instead of creating a pattern stage from within a pipeline, you can also create a pipeline for a pattern directly.

To create a pipeline using a pattern:

1. Click **Patterns** in the left tree on the Home page.

   The Patterns page appears.
2. Scroll through the list of available patterns and select the required pattern.
3. Click **Use this pattern** within the selected pattern tile.

   The Create pipeline using <Pattern> dialog box appears.
4. Fill in the details for the metadata in the **Pipeline** section.

5. Enter details for the **Pattern Stage**.
6. Click **Save**.

   The pipeline editor opens where you can specify the parameters required for the pattern. The pipeline also appears in the Catalog.

4 Understanding Expression Builder Functions

Expression Builder is an editor that allows you to build expressions using various existing functions. The expressions help you in achieving the required results for your pipelines.

Topics:

What are 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

What are 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

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

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

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

What are 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
`timestamp()`	Returns the current output time
`year(date)`	Returns year of the date

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

Format	Example Values
`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
`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

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 of the week in the time represented by this date object. Returns a big integer indicating the day of the week represented by this date.

Examples

If Sunday=0, Monday=1 and so on, then:

Function	Result
`day(12/06/17 09:15:22 AM)`	3
`day(2017:11:23 11:20:25 PM)`	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

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

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

second(date)

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

Example

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

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

What are 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. The return value is of the datatype `double`.

What are 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
`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`.
`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`.

What are 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
`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

Function Name	Description
`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
`toRadians(value1)`	Returns the measurement of the angle in radians
`ulp(value1)`	Returns the size of an ulp of the argument

maximum(value1, value2)

Returns the maximum 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
`maximum(1999220,1997220)`	1999220

Function	Result
`maximum(135.45, 135.50)`	135.50

Note:

If the user provides two different data types as input arguments, then Stream Analytics does implicit conversion to convert one of the argument to the other argument's type.

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.

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
`round(3.5)`	4

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

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

What are 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

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

What are 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.

What are 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

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.

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

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

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

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

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

Troubleshooting Oracle Stream Analytics

Oracle Stream Analytics provides the capability to analyze and monitor stream of events in real time. User can create pipelines to model the stream processing solutions. Typically Oracle Stream Analytics pipelines run continuously for long duration until killed explicitly or fails. In this guide, we will explain steps to troubleshoot various issues which you can encounter during pipeline creation, execution or deployment time.

Pipeline Monitoring and Debug Metrics

For every running Oracle Stream Analytics pipeline, there is a corresponding spark application deployed on 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 not working as per expectation.

To access monitoring and debug metrics:

1. Open Spark Master UI.

   You can obtain Spark Master URL from the System Settings. The Spark Master page displays a list of running applications. Each entry in the list of application contains details about application such as application ID, name, owner, current status etc.

2. Open Application Master Page.

   Click the link with the caption `ApplicationMaster` to open the Application Details page. Ensure that you click the link for the application corresponding to your pipeline. For more details see Determine the Spark Application Name Corresponding to a Pipeline.

3. Click the Pipeline tab. This is Pipeline Summary Page having details of all stages inside an Oracle Stream Analytics Pipeline.

   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:** 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 which is being done in each of the pipeline stage. click on any Stage ID entry in the Pipeline Stage table to open **Pipeline Stage Details Page.**
- **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 transformation 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.

4. The Stage Summary page provides details about all the transformations in a stage. This page has following information:
   - **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 Transformations:** This section displays a table having details about all transformations being performed in each pipeline 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.
   - **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 that OfferByAge is a RULE stage in Oracle Stream Analytics pipeline. To compute this stage, Oracle Stream Analytics is computing a CQLDStream transformation. 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.

5. Click on **CQLDStream** to open **CQL Engine Summary** page corresponding to the transformation.

   - **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.

6. The CQL Engine Summary page has details about all the transformations in a stage. This page has following information:

   - **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 killed.

7. This page contain details about execution and HA statistics for the query corresponding to a Oracle Stream Analytics pipeline stage. Following are the details displayed on this page:
    - **Query ID:** System generated identifier for query
    - **Query Text:** Query String
    - **Num 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 construcsts, number of input kafka partitions and number of executors assigned to application.
    - **Execution Statistics:** 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. 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**.
9. This page contains few additional information about each execution operator apart from the CQL Engine Query Details page provides all essential metrics for each operator.

Common Issues and Remedies

This section provides a comprehensive list of issues or questions categorized by the catalog item. While working on a particular catalog item, if you face any difficulties then refer to the list of common issues or questions corresponding to that catalog item.

Topics

- Pipeline
- Stream
- Reference

- Geofence
- Connection
- Target
- Dashboard
- Cube
- Logs

Pipeline

Common issues encountered with pipelines are listed in this section.

Ensure that Pipeline is Deployed Successfully

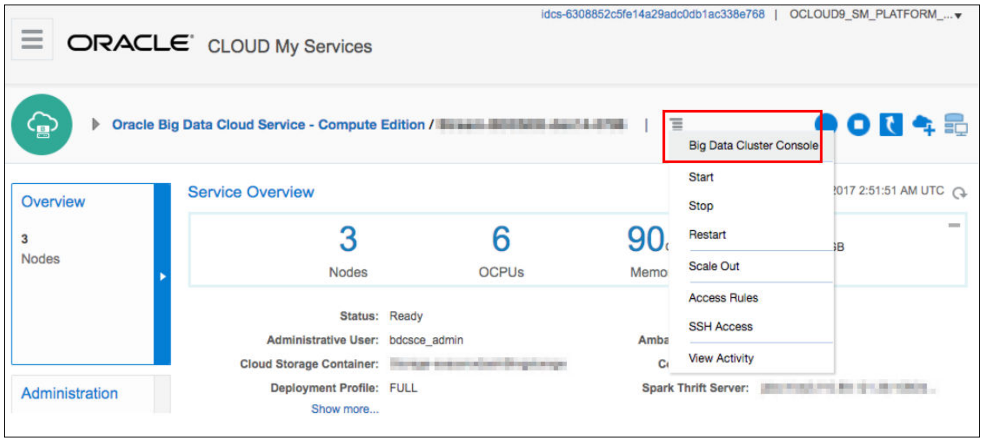
You can deploy pipelines to any Spark Cluster (version 1.6).

Follow the steps in the below sections to verify that the pipeline is deployed and running successfully on Spark cluster.

Verify Pipeline Deployment on Oracle Big Data Cloud Service - Compute Edition based Spark Cluster

Perform these steps if you are subscribes to Oracle Big Data Cloud Service.

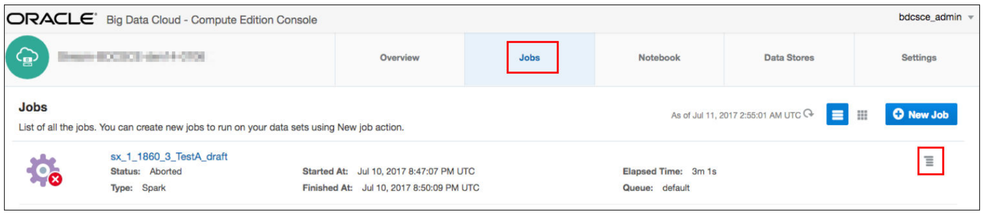
1. Go to PSM user interface and open the home page for Oracle Big Data Cloud Service (BDCSCE) instance.
2. Click on the hamburger menu next to instance name and then click **Big Data Cluster Console**.



Status: Ready
Administrative User: bdcsce_admin
Cloud Storage Container: [redacted]
Deployment Profile: FULL
Spark Thrift Server: [redacted]

3. Enter the login credentials and open the Big Data Cluster Console home page.
4. Navigate to **Jobs** tab.

   You can see a list of jobs. Each job corresponds to a spark pipeline running on your BDCSCE cluster.



[redacted]

Jobs

List of all the jobs. You can create new jobs to run on your data sets using New job action.

As of Jul 11, 2017 2:55:01 AM UTC

sx_1_1860_3_TestA_draft

Status: Aborted	Started At: Jul 10, 2017 8:47:07 PM UTC	Elapsed Time: 3m 1s
Type: Spark	Finished At: Jul 10, 2017 8:50:09 PM UTC	Queue: default

5. Find the entry corresponding to your pipeline and check the status. For more information, see Determine the Spark Application Name Corresponding to a Pipeline.

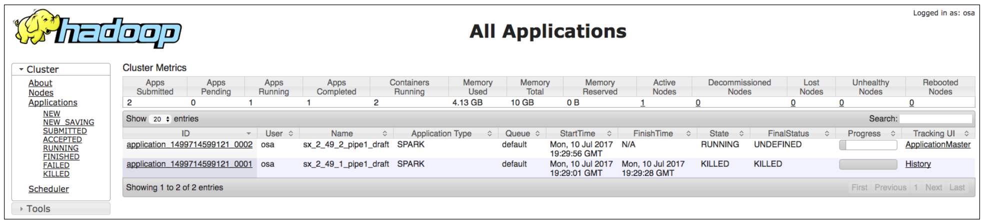
   If you see the status as `Running`, then the pipeline is currently deployed and running successfully.

6. Click the hamburger menu corresponding to the required job to fetch logs and click Logs to get container wise logs.

   You can download these files for further debugging.

Verify Pipeline Deployment on Apache Spark Installation based Spark Cluster

1. Open Spark Master user interface.



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
2	0	1	1	2	4.13 GB	10 GB	0 B	1	0	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1499714599121_0002	osa	sx_2_49_2_pipe1_draft	SPARK	default	Mon, 10 Jul 2017 19:29:56 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster
application_1499714599121_0001	osa	sx_2_49_1_pipe1_draft	SPARK	default	Mon, 10 Jul 2017 19:29:01 GMT	Mon, 10 Jul 2017 19:29:28 GMT	KILLED	KILLED		History

Showing 1 to 2 of 2 entries First Previous 1 Next Last

2. Find the entry corresponding to your pipeline and check the status. For more information, see Determine the Spark Application Name Corresponding to a Pipeline.

   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

You must have 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. Listen to the Kafka topic where the input stream for the pipeline is received.

   Since this topic is created using Kafka APIs, you cannot consume this topic with REST APIs.

   a. Listen to the Kafka topic hosted on Oracle Event Hub Cloud Service. You must use Apache Kafka utilities or any other relevant tool to listed to the topic.

      Follow these steps to listen to Kafka topic:

i. Determine the Zookeeper Address. — Go to Oracle Event Hub Cloud Service Platform home page. Find the IP Address of Zookeeper.

ii. Use following command to listen the Kafka topic:

```
./kafka-console-consumer.sh --zookeeper IPAddress:2181 --topic
nano
```

b. 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:

i. Determine the Zookeeper Address from Apache Kafka Installation based Cluster.

ii. Use the following command to listen the Kafka topic:

```
./kafka-console-consumer.sh --zookeeper IPAddress:2181 --topic
nano
```

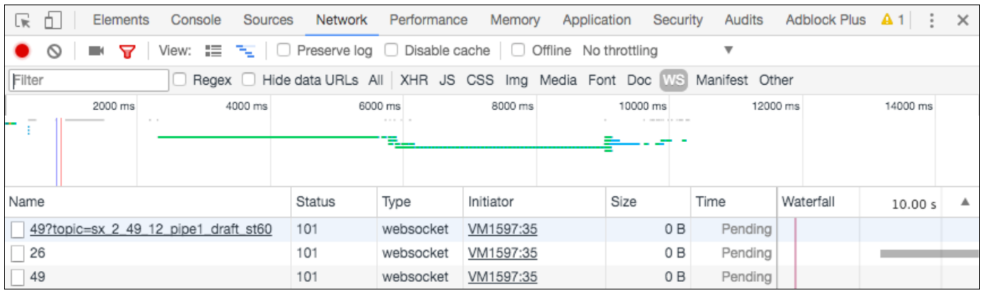
Determine the Spark Application Name Corresponding to a Pipeline

You can perform the following steps 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.



Name	Status	Type	Initiator	Size	Time	Waterfall	10.00 s
49?topic=sx_2_49_12_pipe1_draft_st60	101	websocket	VM1597:35	0 B	Pending		
26	101	websocket	VM1597:35	0 B	Pending		
49	101	websocket	VM1597:35	0 B	Pending		

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

Perform the following steps to ensure caching is leveraged in the pipeline:

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 Details page.
5. In the CQL Engine Summary Details 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.

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 stage which is currently selected. At any moment of time, there will be one stage which will be selected. Try switching to different stage. If you observe output in live table for another stage, the problem can be associated with the stage only. To debug further, go to step 5.

   If there is no output in any stage, then move to step 2.
2. Make sure that pipeline is still running on Spark Cluster. For more details, see Ensure that Pipeline is Deployed Successfully.
3. If the spark application for your pipeline is killed or aborted, then it suggests that 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. Make sure that input stream is pushing events continuously in pipeline.

b. If input stream is pushing events, make sure that the each of the pipeline stages is processing events and emitting outputs.

c. If both of the above steps are verified successfully, then make sure that application is able to push output events of each stage to its corresponding monitor topic. To achieve this:

   i. Determine the monitor topic for the stage where the output of stage is being pushed into using instructions provided in 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 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 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

The topic name for pipeline stage output will be of format **PipelineID_StageID**. A sample topic name is `sx_TestPipeline_5226CDCB_2F11_4A46_8987_4BFCE6FED617_K60bGBKF_draft_st040800FD_A5B3_4AC2_90D5_ED85EB528F41`. In this string, the pipeline name is `sx_TestPipeline_5226CDCB_2F11_4A46_8987_4BFCE6FED617_K60bGBKF_draft` and the stage id is `st040800FD_A5B3_4AC2_90D5_ED85EB528F41`.

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 Determine the Spark Application Name Corresponding to a Pipeline 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.

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 pipeline has been successfully deployed to Spark Cluster. For more information, see Ensure that Pipeline is Deployed Successfully. Also make sure 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

Perform the below steps to verify whether 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 long time, then drill down into stage details.

Stream

Common issues encountered with streams are listed in this section.

Cannot Find MapMessage Format

In this release, only CSV and JSON formats are supported for FILE stream. For Kafka streams, only CSV, JSON and AVRO formats are supported.

Cannot Find AVRO Format

In this release, only CSV and JSON formats are supported for streams (FILE stream type). Hence, you cannot find the AVRO format.

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.

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. Check if lenient parameter under Source Type Properties for the Stream. 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, Please obtain the application logs for further debugging.

Connection

Common issues encountered with connections are listed in this section.

Ensure that Connection is Active

Kafka Connection

If your connection is a Kafka Connection, then use the following steps:

1. Go to Catalog and select the Kafka connection which you want to test.
2. Click Next to go to "Connection Details" tab.
3. Click Test Connection. If the test is successful, it indicates that the connection is active.

Database Connection

If your connection is a database Connection, then use the following steps:

1. Go to Catalog and select the database connection which you want to test.
2. Click Next to go to "Connection Details" tab.
3. Click Test Connection. If the test is successful, it indicates that the connection is active.
4. If the test fails, you'll see an error message to indicate that connection can't be made to database. To resolve the issue:
   - `OSA-01260 Failed to connect to the database. IO Error: The Network Adapter could not establish the connection` — indicates that that the DB host is not reachable from OSA design time
   - `OSA-01260 Failed to connect to the database. ORA-01017: invalid username/password; logon denied` — indicates that the credentials are incorrect and you can't access database using these credentials

Coherence Connection

If your connection is a coherence Connection, then use the following steps:

OSA doesn't provide capability to Test connection for a coherence cluster. Refer to Oracle Coherence documentation to find utilities and tools to achieve that.

Druid Connection

If your connection is a druid Connection, then use the following steps:

1. Go to Catalog and select the database connection which you want to test.
2. Click Next to go to "Connection Details" tab.
3. Click Test Connection. If the test is successful, it indicates that the connection is active.
4. If you see error message like `OSA-01460 Failed to connect to the druid services at zooKeeper server.`, then the druid zookeeper is not reachable. Make sure that druid services and zookeeper cluster are up and running.

JNDI Connection

If your connection is a JNDI Connection, then use the following steps:

1. Go to Catalog and select the database connection which you want to test.
2. Click Next to go to "Connection Details" tab.
3. Click Test Connection. If the test is successful, it indicates that the connection is active.
4. If you see error message like `OSA-01707 Communication with server failed. Ensure that server is up and server url(s) are specified correctly.` then either server is down or server url(s) is incorrectly specified. Server url can only be specified in format host1:port1,host2:port2 and so on.
5. If you see error message like `OSA-01706 JNDI connection failed. User: weblogic, failed to be authenticated` then ensure username and password is specified correctly.

Reference

Common issues encountered with references are listed in this section.

Ensure that a Reference is Cached

If a reference is of type COHERENCE, then caching is not supported as the source itself is a cache.

If a reference is of type DATABASE, then use following steps to check if the reference is cached:

1. Go to Catalog and open the required Reference.
2. Verify that the Enable Caching property in Source Type Parameters section is set to true. If it is TRUE, caching is enabled. If not, edit the reference and enable the property.

Target

Common issues encountered with targets are listed in this section.

Cannot Find AVRO and MapMessage Formats for REST Target

In this release, only CSV and JSON formats are supported for REST targets.

Cannot Find MapMessage Format for Kafka Target

In this release, only CSV, JSON and AVRO formats are supported for Kafka targets.

Cannot See Any Events in REST Endpoint for REST Target

If pipeline is in draft mode, then OSA doesn't push events to REST endpoint. Only a published application pushes the events to REST target.

Cannot See Any Events in the Target for Kafka Topic

If a pipeline is in draft mode, then OSA doesn't push events to Kafka Topic. Only a published application pushes the events to output Kafka target.

Geofence

Common issues encountered with geofences are listed in this section.

Ensure that Name and Description for Geo Fence is Set Correctly

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 geo fence.
2. Click Edit for Source Type Properties and then Next.
3. Ensure that mapping for Name and Description is defined in Shape section.
4. Once these mappings are defined, you can see name and description for geo fence.

Ensure that DB-based Geofence is Working

To ensure that a database-based geofence is working:

1. Go to Catalog and open the required database-based geo fence.
2. Ensure that the connection used in geo fence is active by clicking test button in database connection wizard.
3. Ensure that table used in geo fence is still valid and exists in DB.
4. Go to the geo fence page and verify that the issue is resolved.

Cube

Common issues encountered with cubes are listed in this section.

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.

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 its 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).

Dashboard

Common issues encountered with dashboards are listed in this section.

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.

Dashboard Layout Reset after You Resized/moved the Visualizations

Use the following steps to troubleshoot:

1. This might happen, if user forgets to save the dashboard after movement/resizing of visualizations.
2. Make sure to click Save after changing the layout.

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, make sure that the pipeline is deployed and running on cluster. Once you have the live output table displaying data, it should show up on streaming visualization.

Troubleshoot 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:

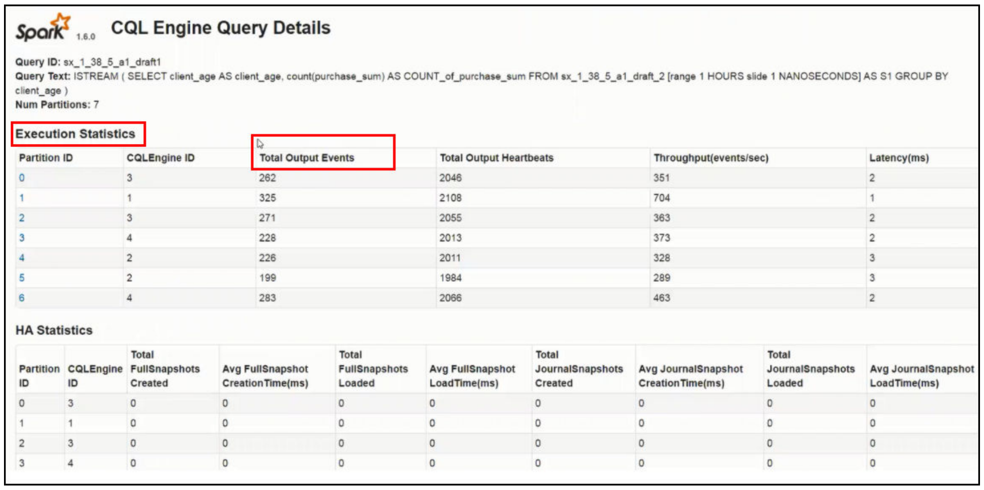
1. Ensure that Pipeline is Deployed Successfully
2. Ensure that the Input Stream is Supplying Continuous Stream of Events to the Pipeline
3. Ensure that CQL Queries for Each Query Stage Emit Output
4. Ensure that the Output of Stage is Available

Ensure that CQL Queries for Each Query Stage Emit Output

Check if the CQL queries are emitting output events to monitor CQL Queries using CQL Engine Metrics.

Follow these steps to check the output events:

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.



CQL Engine Query Details

Query ID: sx_1_38_5_a1_draft1
Query Text: ISTREAM (SELECT client_age AS client_age, count(purchase_sum) AS COUNT_of_purchase_sum FROM sx_1_38_5_a1_draft_2 [range 1 HOURS slide 1 NANOSECONDS] AS S1 GROUP BY client_age)
Num Partitions: 7

Execution Statistics

Partition ID	CQLEngine ID	Total Output Events	Total Output Heartbeats	Throughput(events/sec)	Latency(ms)
0	3	262	2046	351	2
1	1	325	2108	704	1
2	3	271	2055	363	2
3	4	228	2013	373	2
4	2	226	2011	328	3
5	2	199	1984	289	3
6	4	283	2066	463	2

HA Statistics

Partition ID	CQLEngine ID	Total FullSnapshots Created	Avg FullSnapshot CreationTime(ms)	Total FullSnapshots Loaded	Avg FullSnapshot LoadTime(ms)	Total JournalSnapshots Created	Avg JournalSnapshot CreationTime(ms)	Total JournalSnapshots Loaded	Avg JournalSnapshot LoadTime(ms)
0	3	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
2	3	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0

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

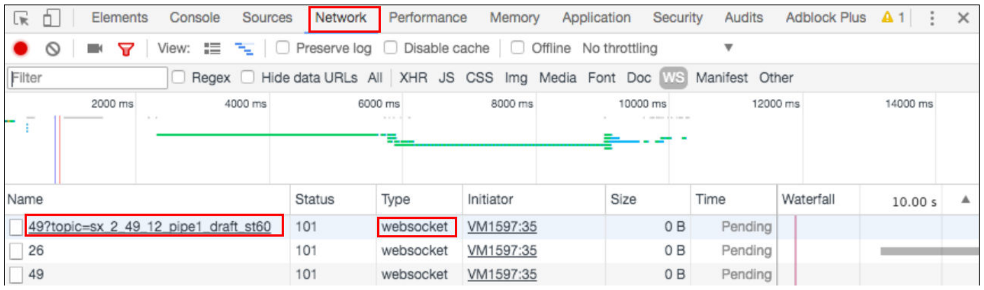
One of the essential things required to troubleshoot a pipeline is to ensure that the output of stage is available in monitor topic.

Follow these steps to check if the output stream is available in the monitor topic:

1. Ensure that you stay in the pipeline Editor and don't click **Done**. Else, the pipeline will be 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.



Name	Status	Type	Initiator	Size	Time	Waterfall 10.00 s
49?topic=sx_2_49_12_pipe1_draft_st60	101	websocket	VM1597:35	0 B	Pending	
26	101	websocket	VM1597:35	0 B	Pending	
49	101	websocket	VM1597:35	0 B	Pending	

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 Oracle Event Hub Cloud Service. You must use Apache Kafka utilities or any other relevant tool to listed to the topic.

      Follow these steps to listen to Kafka topic:

i. Determine the Zookeeper Address. — Go to Oracle Event Hub Cloud Service Platform home page. Find the IP Address of Zookeeper.

ii. Use following command to listen the Kafka topic:

```
./kafka-console-consumer.sh --zookeeper IPAddress:2181 --topic
sx_2_49_12_pipe1_draft_st60
```

b. 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:

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
```

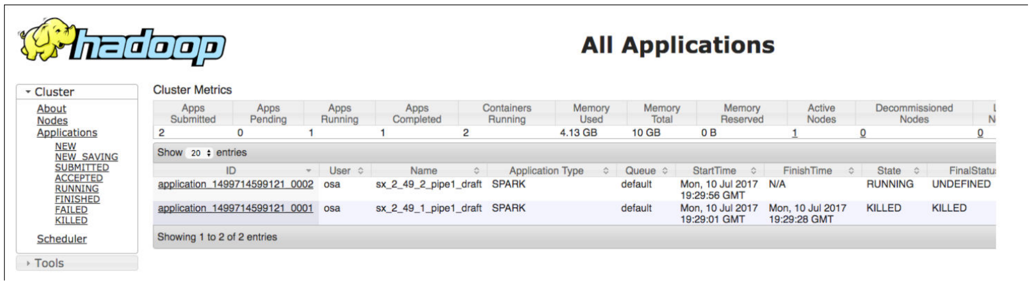
Access CQL Engine Metrics

When a pipeline with a query or pattern stage is deployed to a Spark cluster, you can perform the complex event processing using a set of CQL Engine Metrics running inside the spark cluster.

Use CQL queries which can do aggregate, correlate, filter, and pattern matching over a stream of events. Spark provides an out-of-the-box pipeline UI (commonly running on `<host>:4040`) that can help users to monitor a running Spark Streaming pipeline. As CQL queries also run as part of Spark Streaming pipeline, the Spark pipeline UI is extended to include monitoring capabilities of CQL queries.

To access CQL Engine metrics:

1. Create a pipeline with at least one query or pattern stage.
2. Navigate to Spark Master User Interface.



Cluster Metrics

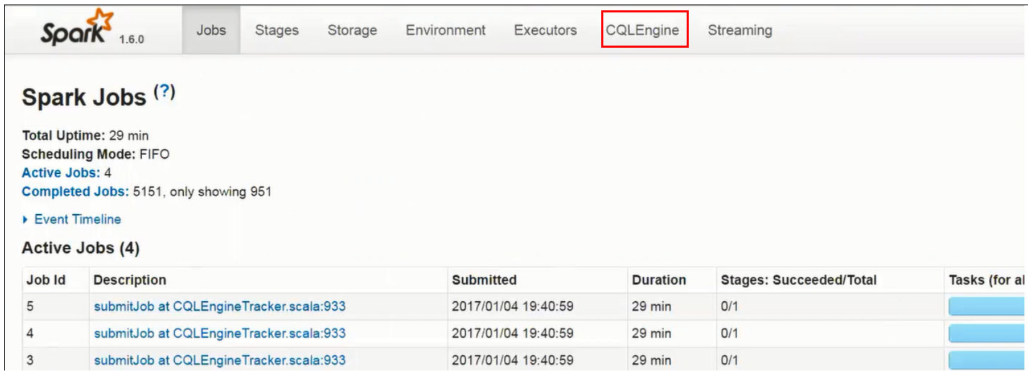
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes
2	0	1	1	2	4.13 GB	10 GB	0 B	1	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatu
application_1499714599121_0002	osa	sx_2_49_2_pipe1_draft	SPARK	default	Mon, 10 Jul 2017 19:29:56 GMT	N/A	RUNNING	UNDEFINED
application_1499714599121_0001	osa	sx_2_49_1_pipe1_draft	SPARK	default	Mon, 10 Jul 2017 19:29:01 GMT	Mon, 10 Jul 2017 19:29:28 GMT	KILLED	KILLED

Showing 1 to 2 of 2 entries

3. Click the **CQL Engine** tab.


Spark 1.6.0 Jobs Stages Storage Environment Executors CQLEngine Streaming

Spark Jobs (?)

Total Uptime: 29 min
Scheduling Mode: FIFO
Active Jobs: 4
Completed Jobs: 5151, only showing 951

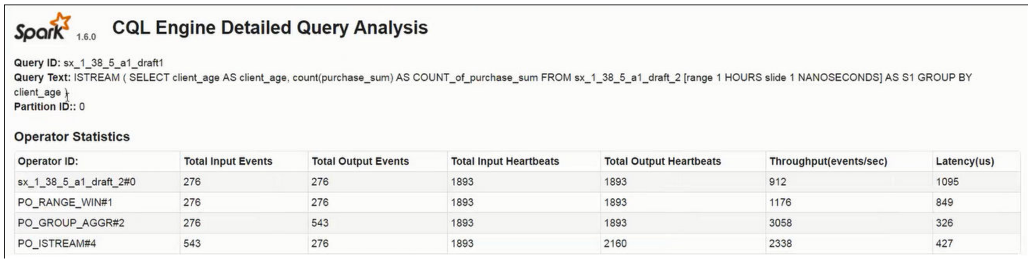
▸ Event Timeline

Active Jobs (4)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for al
5	submitJob at CQLEngineTracker.scala:933	2017/01/04 19:40:59	29 min	0/1	
4	submitJob at CQLEngineTracker.scala:933	2017/01/04 19:40:59	29 min	0/1	
3	submitJob at CQLEngineTracker.scala:933	2017/01/04 19:40:59	29 min	0/1	

You can see the details of all queries running inside a Spark CQL pipeline. This page also shows various streams/relations and external relations registered as part of the pipeline.

4. Click any query to see the details of that query. the query details page shows partition-wise details about a particular running query.
5. Click the specific partition link to determine further details about query plan and operator level details. This page shows the operator level details of a query processing a particular partition.


Spark 1.6.0 CQL Engine Detailed Query Analysis

Query ID: sx_1_38_5_a1_draft1
Query Text: ISTREAM (SELECT client_age AS client_age, count(purchase_sum) AS COUNT_of_purchase_sum FROM sx_1_38_5_a1_draft_2 [range 1 HOURS slide 1 NANOSECONDS] AS S1 GROUP BY client_age)
Partition ID:: 0

Operator Statistics

Operator ID:	Total Input Events	Total Output Events	Total Input Heartbeats	Total Output Heartbeats	Throughput(events/sec)	Latency(us)
sx_1_38_5_a1_draft_2#0	276	276	1893	1893	912	1095
PO_RANGE_WIN#1	276	276	1893	1893	1176	849
PO_GROUP_AGGR#2	276	543	1893	1893	3058	326
PO_ISTREAM#4	543	276	1893	2160	2338	427

Troubleshoot Pipeline Deployment

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.