TIBCO BusinessEvents® Enterprise Edition

Configuration Guide

Version 6.3.1 | September 2024

Contents

Before You Begin

To maintain uniformity, the following terms have been used in the TIBCO BusinessEvents Studio UI and the product documentation:

- TIBCO ActiveSpaces software version 2.x is referred to as *Legacy ActiveSpaces*.
- TIBCO ActiveSpaces software version 4.6.1 and later are referred to as *ActiveSpaces*.

For details about the supported versions, see the *Readme.txt* file available at the TIBCO BusinessEvents® Enterprise Edition Product Documentation page.

Rule Management Server Prerequisite

In addition to Legacy ActiveSpaces as cluster and cache provider, you can also configure TIBCO BusinessEvents Rule Management Server (RMS) with the following combinations:

Cluster	Cache	Store
Apache Ignite	Apache Ignite	None/Shared Nothing/RDBMS/Store Providers (TIBCO ActiveSpaces and Cassandra)
TIBCO FTL	Apache Ignite	None/Shared Nothing/RDBMS/Store Providers (TIBCO ActiveSpaces and Cassandra)
TIBCO FTL	No cache	TIBCO ActiveSpaces

By default, Apache Ignite is used as the cluster and cache provider.

For more information about configuring these for your RMS project, see *TIBCO BusinessEvents Configuration Guide*.

Third-Party Software Documentation References

For complete details about the third-party software used in the project, see its documentation.

Note: When you obtain third-party software or services, it is your responsibility to ensure you understand the license terms associated with such third-party software or services and comply with such terms.

Third-Party Software Documentation

Software	Used as	Documentation Reference URL
TIBCO ActiveSpaces 4.6.1 and above	Store provider	TIBCO ActiveSpaces documentation
TIBCO ActiveSpaces 2.x	Cluster and Cache provider	TIBCO ActiveSpaces documentation
Apache Kafka	Channel	Apache Kafka documentation
Confluent Schema Registry	Schema Registry	Confluent documentation
TIBCO Messaging - Schema Repository for Apache Kafka	Schema Registry	TIBCO Messaging - Schema Repository for Apache Kafka documentation
Apache Pulsar	Channel	Apache Pulsar documentation
Apache Cassandra	Store provider	Apache Cassandra documentation
GridGain	Data Center Replication	GridGain documentation
TIBCO FTL	Cluster provider	TIBCO FTL documentation

Software	Used as	Documentation Reference URL
Apache Ignite	Cluster and Cache provider	Apache Ignite documentation
InfluxDB	Metrics store provider	InfluxDB documentation
Grafana	Application metrics visualization	Grafana documentation
Ignite CDC	Data Center Replication	Apache Ignite documentation
Control Plane	Metrics store provider	TIBCO® Platform Documentation
Apache Maven	Native Maven projects	Apache Maven Documentation

Cluster Deployment Descriptor (CDD)

The Cluster Deployment Descriptor (CDD) is an XML file to configure a project for deployment. You can use the TIBCO BusinessEvents Studio CDD editor to configure these settings and properties.

One EAR file and one CDD file define all the settings for all the engines and agents you want to deploy for a single application. Because the deploy-time configuration settings for all processing units are in the CDD file, you do not have to rebuild the EAR file to make changes to deploy-time settings.

When you deploy a processing unit, you specify these items:

- An EAR file
- A CDD file that you have configured for that EAR
- A processing unit (engine) that is configured in the specified CDD file.

Note: For deployment using TIBCO Administrator, the CDD file you specify can be in the file system or in the EAR file. To specify a CDD located in an EAR file, provide its project path and name. The CDD file does not accept global variables as values.

You can configure multiple CDD files for a project for different purposes such as testing a design, trying out different object and cluster management options, dividing the work differently between agents and processing units (engines), and so on. However, it is a good approach to use the same CDD file to deploy all processing units for an application in a specific environment.

For details about the procedure to add a CDD, see Adding a CDD File to the Project.

CDD Settings at Runtime

It is important to understand the effect of design-time settings in the runtime environment. The tab and section within a tab where you set values in the CDD can affect the scope of those values, and how they can be overridden.

Note: You can disable a channel for a specific Process Unit or Agent Class by adding CDD property `be.channel.deactivate` on Process Unit or Agent Class tab and set it's value to a comma, separated lists of channels.

Using Properties at Different Levels

The scope of a property depends on the property sheet you add it to. Not all properties are valid at all levels. Use your judgment.

For example, properties that include the agent class name, such as `Agent.`*`AgentClassName`*`.threadcount`, can be used at different levels. Here is the scope of each level for these *`AgentClassName`* properties:

Cluster level

Applies to all *`AgentClassName`* agents in the cluster.

Processing unit level

Applies to any *`AgentClassName`* agent deployed in the specified processing unit.

Agent class level

Applies to any *`AgentClassName`* agent, used in any processing unit.

(Not all agent-level properties include the agent class in the property name.)

Only one value for a property is used when a processing unit is deployed.

Order of precedence at runtime can affect decisions made at design time. For more information, see "Order of Precedence at Run time" in *TIBCO BusinessEvents Administration*.

For details about the available configuration properties, see Configuration Properties Reference.

Global Variables Setup in CDD (for Command Line Startup)

Global variables are added to a project and their value is set using the TIBCO BusinessEvents Studio Global Variables editor.

You can also set global variable values in the CDD file. This can sometimes be useful when you start the agent at the command line.

Note: Global variables set in TIBCO Administrator override those set in the CDD.

Add properties using this format:

```
tibco.clientVar.GVName = value
```

The *GVName* must exactly match the name set in the TIBCO BusinessEvents Studio Global Variables editor.

Note: If global variables are defined in the TIBCO BusinessEvents project using groups, specify the group path using forward slashes. For example, if a variable *JMSuri* is located under a group called `URIs`, specify the variable as `tibco.clientVar.URIs/JMSuri`.

Add such properties at the appropriate level in the CDD, depending on the desired scope: Cluster, Process Unit, or Agent Class.

Adding a CDD File to the Project

The configuration settings for an application are defined in the CDD file. You can use the wizard to specify the default cluster and object management(OM) mode of your project.

If you choose an OM type in the wizard, the template for that OM type is used and all the default settings are loaded. For example, if you choose Cache, then a cache agent with default values is created for you. For example, if you change from Cache to Store and save, the cache agent and its configuration are lost. For all OM types, however, the General settings are configured in the same way. Similarly, you can choose the Cluster Provider Type to get a template for the cluster of your choice.

Note: Names in the CDD must conform to the `NCName` datatype. See the following page for more details:

REC-xml-names

Some Japanese characters, such as half-width Katakana, have issues when they are used in XML names. See the following document for more details:

japanese-xml

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the folder where you want to store the CDD and select **New > Cluster Deployment Descriptor**. The New Cluster Configuration Wizard opens up.
2. In the **File name** field, type a name for the CDD and click **Next**. You can change the name in the editor as required.

   Note: When you are using TIBCO Administrator for deployment, by default, TIBCO Administrator looks for a processing unit named default and a CDD file named default.

3. In the Object Manager Selection page, select a cluster type for the project.
    - Unclustered
    - Clustered

    Based the cluster type, the values of the **Object Management Type** field are listed.
4. Select the **Object Management Type** for the deployment and click **Finish**.

    The available options are described in the following table:

Cluster Type	Object Management Type

Unclustered	**`In Memory`**
	Sets `In Memory` as the **Object Management Mode** (in CDD).
	`Store`
	Sets `Store` as the **Object Management Mode**. The default **Store Provider** is Apache Cassandra; however, you can change it later in CDD, if required

Clustered	**`Cache`**
	Sets `Legacy ActiveSpaces` as the **Cluster Provider** and `Cache` as the **Object Management Mode**. The default cache provider is Legacy ActiveSpaces.
	`Store`
	Sets `TIBCO FTL` as the **Cluster Provider** and `Store` as the **Object Management Mode**. The default store provider is Apache Cassandra; however, you can change it, if required.

The CDD editor opens up.

Tip: Check the Object Management node on the left and ensure that you have selected the correct Object Management type.

You can also right-click the Object Management element and select **Change to *OM Type***.

5. In the CDD editor, on the **Cluster** tab, select **General** and specify the following:
   - The cluster name, although it already contains the file name that you have provided in the new CDD wizard, but you can update it, if needed. If Legacy ActiveSpaces is the cluster provider, any spaces in the name are converted to underscore characters internally.

     Note: Do not use the names `$cluster` or `default` as they are reserved names.
   - If required, you can update the author name and add comments as suitable. (Version and date are not editable.)
6. Save the new CDD file.

What to do next

Configure the CDD file settings on different tabs.

- For details about different cluster, object management, and metrics options available in CDD, see Cluster Configurations for Your Project.
- For configuring database concepts, see *TIBCO BusinessEvents Data Modelling Developer Guide*.

  Note: The database concepts are applicable only for JDBC stores.

Cluster Configurations for Your Project

Using the CDD file, you can configure the cluster provider, cache provider, store provider, and metrics store provider for your project. You can also choose to keep it in unclustered mode and configure *In Memory* object management.

In Memory Object Management

To use your project with In Memory object management (that is, without any cache), you can opt for `Unclustered` as the cluster management mode.

For detailed configurations, see Setting Up In Memory Object Management.

Cluster, Cache, and Store Configurations

The CDD provides hierarchical options to select different providers, which helps you to use different combination of these providers.

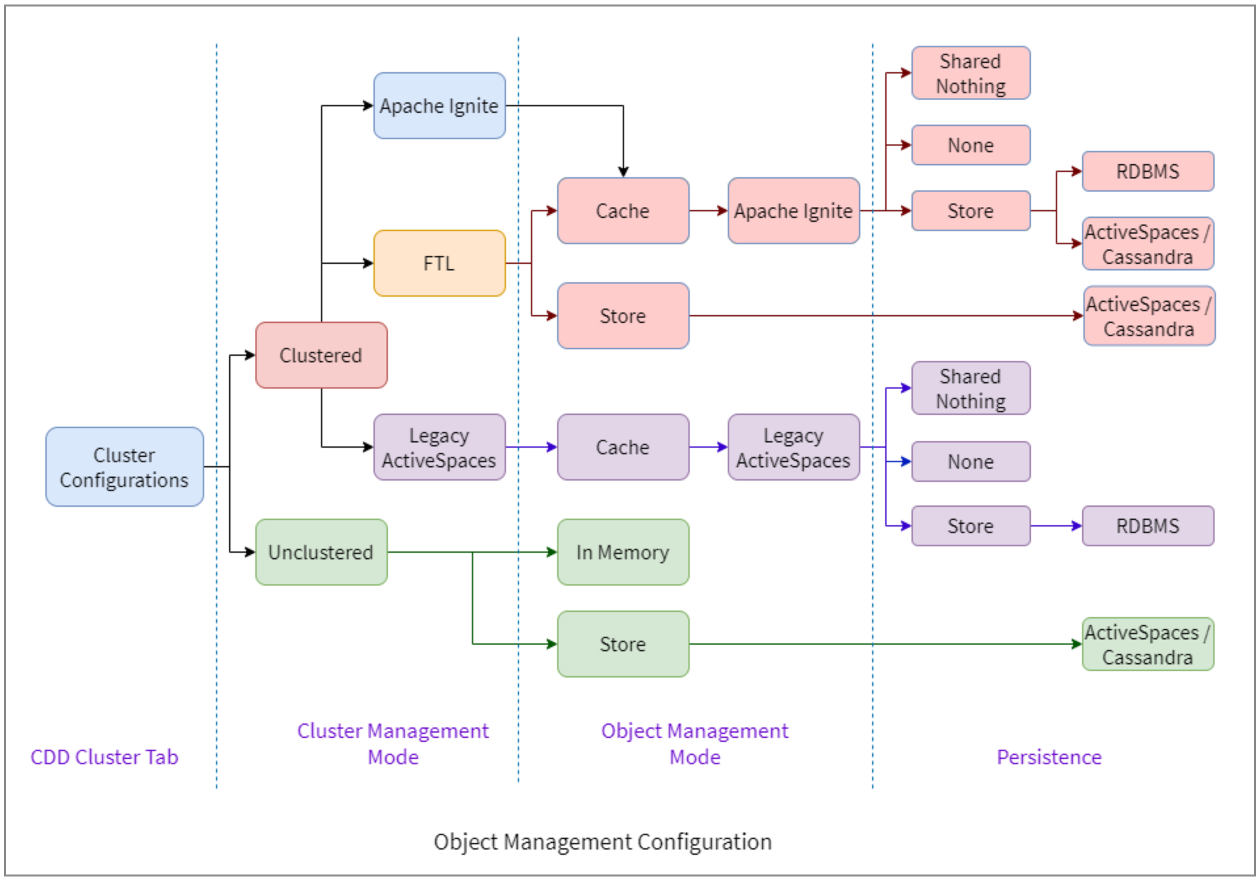
TIBCO BusinessEvents supports the following cluster, cache, and store providers out of the box.

- Cluster Provider
    - Legacy ActiveSpaces
    - TIBCO FTL
    - Apache Ignite
- Cache Provider
    - Legacy ActiveSpaces
    - Apache Ignite
- Store Provider
    - Apache Cassandra
    - ActiveSpaces

Project Configuration Options

The following figure displays the options and combinations that you can configure for your project using the supported providers. The figure lists the options available under each parent option. For example, if you select `Unclustered` as the cluster type, then you can select the Object Management type as either `Store` or `In Memory`. Next, if you select `Store` as the Object Management type, you can select either `Apache Cassandra` or `ActiveSpaces` as the store provider.

Figure 1: Object Management Configuration Options



For further details, see the following topics:

- Adding a CDD File to the Project
- Collections, Agent Classes, and Processing Units

Metrics Store

To view your TIBCO BusinessEvents applications and system metrics, you can select from multiple metrics store providers. These metrics store configurations are independent of the selection of the cluster or cache configurations. Thus, you can get the application metrics for the In Memory application in the same way as you do for the Cache OM application.

The inference agent uses the Application Metrics configurations in the CDD to connect to the configured metrics store providers. The agent then publishes configured entity set updates to the metric store. In the CDD file, you can add entities (Concept and Events) for which you want to publish updates. You can also use rule functions to filter out the entities for which you do not want updates. You can then connect to this metrics store in your visualization software to create your own visualizations or dashboards based on the data published by your TIBCO BusinessEvents application.

TIBCO BusinessEvents supports the following metrics store and visualization software:

- **Metrics Store:** InfluxDB and Control Plane
- **Visualization Software:** Grafana

See the following topics for configurations required for these metrics providers:

- Configuring InfluxDB as a Metrics Store
- Configuring Control Plane as a Metrics Store

Comparison across Various Configurations

Cache OM with Store versus Store OM Configuration

In TIBCO BusinessEvents, you can choose the object management (OM) strategy that you want for your application persistence. The following table provides a feature comparison between two persistence strategies:

- Cache OM with store
- Store OM

Features	Cache OM with Store	Store OM
Expiry	When TTL expires for a concept, it is evicted from the entity cache as well as the object table after a specified delay (if legacy entity lookup strategy is used)	Data is removed directly from the store after the TTL expires.
Query Support	Query execution applies at the cache level.	Query execution applies at the store level, so the feature is limited to what the end store supports.
Data Recovery	Data recovery is applicable to cache configuration only.	Data recovery is not applicable to store configuration.

Clustered (with Cache and/or Store) versus Unclustered (with Store) Configuration

You can use the clustered (with cache and/or store) or the unclustered configuration with store. The difference between using these configurations is that the clustered configurations support the following features:

- Fault tolerance support
- Distributed locking

These features are not supported by the unclustered with store configuration.

Setting Up In Memory Object Management

If you do not want to use cluster and/or persistence for your project, you can use In Memory object management

For details about In Memory object management, see "In Memory Object Management" in *TIBCO BusinessEvents Architects Guide*.

Procedure

1. In TIBCO BusinessEvents Studio, perform one of the following steps for your project:
    - In the **New CDD** wizard, add a new CDD file and select **Cluster Type** as `Unclustered` and **Object Management Type** as `In Memory`. For details, see Adding a CDD File to the Project
    - If modifying an existing CDD file, on the **Cluster** tab, select the **Cluster Management** and change the **Cluster Management Mode** to `Unclustered`.

      By default, the object management is set to In Memory.
2. To verify, select **Object Management** and see that **Object Management Mode** is set to `In Memory`.
3. Save the CDD file.

What to do next

- You can set up the Application Metrics configurations to display data in a visualization software for analysis. For more information, see Application Metrics Configurations
- Setup the agents and processing units for your project, see Collections, Agent Classes, and Processing Units.

Object Management Configurations

To configure object management for your application, select the store provider, cache provider, configure a cache agent quorum, and number of back-up copies.

If you have a backing store, you can also configure a limited cache and specify the cache size for each entity.

Configuring a Cluster Provider

- Configuring TIBCO FTL as the Cluster Provider
- Configuring Apache Ignite as the Cluster Provider
- Setting Up Legacy ActiveSpaces as Cluster and Cache Provider

Configuring the Backing Store

- For backing store-related project configuration details, see Persistence Options.
- For details about setting up the backing store itself, see *TIBCO BusinessEvents Administration*.

Configuring Management of Domain (Entity) Object Instances

If you choose Cache OM or Store OM, you must also configure how to manage the domain (that is, entity) object instances.

For example, you can determine whether the instances are flushed from the Rete network after each RTC (as is generally recommended) or are kept in the cache.

If you set up a backing store, you can specify additional settings. For example, you can define a subset of ontology object instances to be stored in the backing store, and you can control which (if any) object instances are loaded into the cache from the backing store at system startup.

Configuring a Limited (or Unlimited) Cache

You can use the options available to configure a cache of a limited size, or use the described procedure to set options for an unlimited cache.

Also see Limited and Unlimited Cache Size in *TIBCO BusinessEvents Architects Guide*.

Note: Use of limited cache is supported only when a backing store is used. The backing store retains entries in excess of the limit. Without the use of a backing store data inconsistency could result:

- Entries for an object in the object table (an internally used cache) and in the object cache itself could expire independently of each other.
- Domain object settings for limited cache apply at the object level. Related concepts could have different settings. For example, a container concept could have a limited cache setting and its container concept an unlimited cache setting. Each could be evicted at different times.

Configuring Apache Ignite as the Cluster Provider

TIBCO BusinessEvents provides Apache Ignite as a built-in and open-source clustering option. Apache Ignite is a well-suited option for applications with smaller cluster sizes.

Before you begin

Ensure that a CDD file is added to your project, see Adding a CDD File to the Project.

Procedure

1. In the TIBCO BusinessEvents Studio, open the CDD file for editing.
2. On the **Cluster** tab, select **Cluster Management** and `Apache Ignite` as the **Cluster Provider**.
3. Configure the following cluster properties.

Apache Ignite Cluster Properties

Property	Description
Discovery URL	Specify a set of IP addresses and ports for node discovery. For fault tolerant discoveries, specify multiple IP addresses with port ranges as comma separated values. **Example:** IP1:47500..47510[,IP2:PORTS]
Listen Port	Port to listen on for discovery of nodes in a cluster.
Communication Port	Specify the port or range of ports used by the Ignite Communication SPI.
Socket Write Timeout	Specify the socket write timeout for the TCP connection.
Connect Timeout	Specify the connection timeout used for establishing the connection with remote nodes.

Apache Ignite Cluster Properties(Continued)

Property	Description
Join Timeout	Specify the join timeout. If a non-shared IP finder is used and a node fails to connect to any address from the IP finder, the node keeps trying to join within this timeout. If all addresses are unresponsive after the timeout, an exception is generated and node start up fails.
Network Timeout	Specify the maximum network timeout for network operations.
Failure Detection Timeout	Specify the failure detection timeout. The failure detection timeout is used to determine how long the communication or discovery SPIs must wait before considering that a remote connection has failed.
Enable Hostaware	Select the checkbox to enable the host-aware replication. By default, the checkbox is selected. When host-aware replication is enabled, if the cache nodes are deployed on multiple machines to satisfy replication by the host, then replication does not happen on the same node (or happens only according to the number of hosts available). If cache nodes within the cluster are on the same machine, then host-aware replication needs to be disabled to honor **Number of Backup Copies**. When host based replication is disabled then data is replicated on neighbor cache nodes on same machine.
Hostaware Hostname	Hostnames that are used in identifying members (and therefore naming Shared nothing file and folders), are generated from underlying OS. If you would like to assign hostnames manually instead, for reasons such as hostname/machine changes and testing, then provide hostnames in the CDD by using this field.

4. *(Optional)* To enable secure Apache Ignite connection among all nodes, select the **Security Enabled** checkbox, and fill values for the following fields:

Field	Description
Trusted Folder Path	When using a secure connection, provide the path to the base folder location containing the trust certificate files.
Identity	Path to the keystore identity file when using the two-way SSL.
Trust Store Password	Specify a password for the trust store.

5. Save the CDD file.

What to do next

Configure Apache Ignite as the cache provider. See Configuring Apache Ignite as a Cache Provider for more information.

Configuring Apache Ignite as a Cache Provider

Apache Ignite is a good low-latency and high-performance caching option for your projects in TIBCO BusinessEvents. You can use Apache Ignite as a cache provider for your projects that use clusters other than the Legacy ActiveSpaces cluster.

To display advanced properties of Apache Ignite cache on the TIBCO BusinessEvents Studio UI, see Enabling Advanced Settings For Apache Ignite.

To replicate data between clusters, you can choose one of the following two options:

- `Using GridGain Enterprise Edition`
- `Using Ignite CDC`

Using GridGain Enterprise Edition

You can use GridGain Enterprise edition, a platform built on Apache Ignite, as a cache provider. GridGain Enterprise edition provides improved data accessibility across data stores and applications.

Before you begin

- **TIBCO FTL** or **Apache Ignite** must be configured as the cluster provider. For more information, see Configuring TIBCO FTL as the Cluster Provider or Configuring Apache Ignite as the Cluster Provider
- If you have configured **Apache Ignite** as the cluster provider, then you do not need to define the common connection settings. The application uses values for these settings from the cluster provider configuration.
- However, if you have selected **TIBCO FTL** as the cluster provider, then you have to configure the Apache Ignite cache properties and the cluster properties at once.
- (Optional) To use GridGain for data center replication, download GridGain Enterprise Edition. Replace Apache Ignite JARs in the `BE_HOME/lib/ext/tpcl/apache/ignite` directory with the related GridGain JARs including `gridgain-core` JAR. See GridGain documentation for more information.

Procedure

1. In TIBCO BusinessEvents Studio, open the project CDD file and on the **Cluster** tab, select **Object Management**.
2. In the **Configuration** window, specify the cache as follows:
    - From the **Object Management Mode** list, select **Cache**.
    - From the **Cache Provider** list, select **Apache Ignite**.

3. Define the cache settings as follows.

Apache Ignite Cache Properties

Field	Description
Cache Agent Quorum	Specifies a minimum number (quorum) of storage-enabled nodes that must be active in the cluster when the system starts up before the other agents in the cluster become fully active. **Default value**: 1 **Note:** The inference agent does not start processing events if the number of cache members in the cluster does not satisfy the quorum count.
Number of Backup Copies	The number of backup copies specifies the number of members of the distributed cache service that hold the backup data for each unit of storage in the cache. There is no guarantee of no data loss if the Number of Backup Copies is set to 0. **Default value:** 1 For a backup count of 1 to be effective, you need at least two cache agents (or storage enable nodes).

4. *(Optional)* To enable encryption for the stored data when you use `Shared Nothing` persistence, select the **Enable Storage encryption** checkbox.
5. *(Optional)* If you select the **Enable Storage encryption** checkbox, provide the value to the **Identity File Path** field. The identity file provides a keystore with type and password.
6. *(Optional)* To replicate cache data between clusters by using GridGain, select `GridGain Enterprise` from the **Choose cache replication method** dropdown menu.

7. Configure the following GridGain properties.

GridGain Configuration Properties

Field	Description
Data Center ID	Set a numeric value as an identifier for this data center node.
Is Sender Node	Select to make this node a cache data sender. When this checkbox is selected, the following sender-specific connection settings are enabled.

Field	Description
Sender Group Name	Set to a group name to associate this node to a group of sender nodes. This Sender Group Name can be used on multiple nodes if you want a group of nodes to act as sender nodes For example: SenderGroup1.
Receiver Data Center ID	Set to an identifier of a receiver data center node. This must match the Data Center ID provided when configuring the receiver node.
Local Outbound Address	Set to the IP address of the sender node machine. For example: `127.0.0.1`
Receiver Address	Set to the IP address and port number of a receiver node. For example: `127.0.0.1:50001`

GridGain Configuration Properties(Continued)

Field	Description
	(nested table below)

Field	Description
	To add multiple receiver nodes, provide IP addresses of each receiver node with their respective port numbers in a comma-separated list format.
Batch Send Frequency (in milliseconds)	Time interval at which data replication occurs.
Batch Send Size (in bytes)	The maximum size of data that accumulates after which data replication occurs.

Field	Description
Is Receiver Node	Select to make a node as a cache data receiver. When this checkbox is selected, the following receiver-specific connection settings are enabled:

Field	Description
Local Inbound Address	Set to the IP address of the receiver node machine with a valid available port number. For example: `127.0.0.1:50001`

Field	Description
License file location	Browse to GridGain license file (`gridgain-license.xml`) in GridGain installation folder.

> **Note:** To set active-active replication mode, define a node as both sender and receiver.

8. Save the project CDD file.

What to do next

- Set up the configurations to persist data, see Persistence Options
- Set up how to manage objects for cache or store object management, see Domain Objects Configuration.
- Set up the processing units and agents for the project, see Collections, Agent Classes, and Processing Units.

Using Ignite CDC

Apache Ignite is an open source, in-memory computing platform. It provides the foundation for GridGain. You can also replicate cache data by using Apache Ignite's Change Data Capture (CDC) extension. The CDC extension provides Active-Passive or Active-Active types of cross-cluster replication.

> **Note:** Ignite CDC should be implemented in a multihost environment as it is not supported in a single-host environment.

The CDC extension provides the following cross-cluster replication options:

Active-Passive Replication

Only the sender implements the CDC by input to the `Receiver's address`. There is no change required at the receiver's end.

Active-Active Replication

In this, both the sender and the receiver must implement Ignite CDC and input the `Receiver's address`.

To set up a cross-cluster replication, the CDC extension module provides the following methods:

- `IgniteToIgniteClientCdcStreamer`

- `IgniteToIgniteCdcStreamer`
- `IgniteToKafkaCdcStreamer combined with KafkaToIgniteCdcStreamer`

TIBCO BusinessEvents implements `IgniteToIgniteClientCdcStreamer` for cross-cluster replication. See Apache Ignite documentation for more information,.

Before you begin

- **TIBCO FTL** or **Apache Ignite** must be configured as the cluster provider. For more information, see Configuring TIBCO FTL as the Cluster Provider or Configuring Apache Ignite as the Cluster Provider
- If you have configured **Apache Ignite** as the cluster provider, then you do not need to define the common connection settings. The application uses values for these settings from the cluster provider configuration.
- However, if you have selected **TIBCO FTL** as the cluster provider, then you have to configure the Apache Ignite cache properties and the cluster properties at once.
- To use the Apache Ignite's CDC replication feature, set up Ignite CDC extension JAR in the library of BusinessEvents installation. To set up Ignite CDC extension JAR, see Setting up CDC Extension JAR.
- Set up the configurations to **Shared Nothing** as persistence options. See Persistence Options for more information.

Procedure

1. In TIBCO BusinessEvents Studio, open the project CDD file and on the **Cluster** tab, select **Object Management**.
2. In the **Configuration** window, select the following:
    - From the **Object Management Mode** list, select **Cache**.
    - From the **Cache Provider** list, select **Apache Ignite**.

3. Define the cache settings as follows.

Apache Ignite Cache Properties

Field	Description
Cache Agent Quorum	Specifies a minimum number (quorum) of storage-enabled nodes that must be active in the cluster when the system starts up before the other agents in the cluster become fully active. **Default value**: 1 **Note:** The inference agent does not start processing events if the number of cache members in the cluster does not satisfy the quorum count.
Number of Backup Copies	The number of back-up copies specifies the number of members of the distributed cache service that hold the back-up data for each unit of storage in the cache. There is no guarantee of no data loss if the Number of Backup Copies is set to 0. **Default value:** 1 For a back-up count of 1 to be effective, you need at least two cache agents (or storage enable nodes).

4. *(Optional)* To enable encryption for the stored data when you use `Shared Nothing` persistence, select the **Enable Storage encryption** checkbox.
5. *(Optional)* If you select the **Enable Storage encryption** checkbox, provide the value to the **Identity File Path** field. The identity file provides a keystore with type and password.
6. *(Optional)* To replicate cache data between clusters by using the Ignite CDC extension, select `Ignite CDC` from the **Choose cache replication method** dropdown menu.

7. Configure the following Ignite CDC properties.

Ignite CDC Configuration Properties

Field	Description
Data Center ID	Sets a numeric value as an identifier for this data center node.
Receiver Address for Ignite CDC	Sets the IP address and port number of a receiver node. For example: `127.0.0.1:50001` To add multiple receiver nodes, provide the IP addresses of each receiver node with their port numbers in a comma-separated list format.

Note: To set active-active replication mode, set the Receiver's Address at the Sender node and the Sender's address at the Receiver node.

8. Configure the following CDD properties in the **Properties** tab of the `.cdd` file.

CDD Properties Configuration

Field	Description
ignite.cdc.failed.connect.attempt.retry.interval	The time interval after which the Ignite CDC will retry to connect to the destination cluster for a failed attempt. **Default value**: 5000 milliseconds
ignite.cdc.connect.retry.attempt.threshold	The number of times that the Ignite CDC will retry to reconnect to the destination cluster. **Default value**: 5

9. Save the project CDD file.

What to do next

- Set up how to manage objects for cache or store object management, see Domain Objects Configuration.
- Set up the processing units and agents for the project, see Collections, Agent Classes, and Processing Units.

Setting up CDC Extension JAR

You can set up the CDC extension JAR from the open-source GitHub `ignite-extensions` repository by performing the following steps:

Procedure

1. To copy the contents from the GitHub repo to your local machine, navigate to the following URL:

```
https://github.com/apache/ignite-extensions.git
```

2. Clone the GitHub repo to your machine.
3. Open the command prompt from the `ignite-extensions` folder and run the following command:

```
git checkout -b <name> 3a9829901b4b313f3ed8b29b26a9b9da212bbd71
```

Note: You can specify any custom branch-name for `<name>`.

4. Edit the `parent-internal/pom.xml` file as shown in the following steps:

a. Open the `ignite-extensions/parent-internal/pom.xml` file.

b. At line #29 of the file, change the value 2.16.0-SNAPSHOT to 2.16.0.

c. At line #61 of the file, change the value from https://repository.apache.org/snapshots to https://repo.maven.apache.org/maven2.

5. In the command prompt from the `ignite-extensions` folder, run the following commands:

   ```
   cd modules/cdc-ext/
   mvn clean package -DskipTests -Dmaven.javadoc.skip=true
   ```

6. A JAR file `ignite-cdc-ext-1.0.0-SNAPSHOT.jar` is created at the following location:.

   ```
   ignite-extensions/modules/cdc-ext/target/ignite-cdc-ext-1.0.0-SNAPSHOT.jar
   ```

7. Copy the `ignite-cdc-ext-1.0.0-SNAPSHOT.jar` file in the `lib` folder of the installation folder.

   ```
   $BE_HOME/lib/ext/tpcl/apache/ignite
   ```

   Note: If there is any change in the API or design architecture of the Ignite, you might need to change the BusinessEvents module implementation.

Result

The `ignite-cdc-ext-1.0.0-SNAPSHOT.jar` file is copied to the `$BE_HOME/lib/ext/tpcl/apache/ignite` folder.

Enabling Apache Ignite REST Services

You can use the HTTP REST client provided by Apache Ignite for various cache monitoring activities.

To enable Apache Ignite REST Services in your TIBCO BusinessEvents application, you must add `ignite-rest-http` library JAR files in the classpath in the `be-engine.tra` file located at *`BE_HOME`*`/bin`. The required JAR files are available in the *`BE_HOME`*`/lib/ext/tpcl/apache/ignite/optional` folder.

For example, you can update the `CUSTOM_EXT_APPEND_CP` property in the `be-engine.tra` file:

```
tibco.env.CUSTOM_EXT_APPEND_CP=%BE_
HOME%/lib/ext/tpcl/apache/ignite/optional/ignite-rest-http
```

For adding an external library in the classpath from TIBCO BusinessEvents Studio, see "Working with External Library and Custom Function Paths" in *TIBCO BusinessEvents Developer Guide*.

For more information on Apache Ignite REST API, see Apache Ignite documentation.

Configuration of Ignite Cache Indexing

Ignite supports H2 as the default indexing engine and Apache Calcite can be used as an optional indexing engine.

The H2 index JAR files are at location:

```
BE_HOME\lib\ext\tpcl\apache\ignite\ignite-indexing
```

To switch to the Apache Calcite index, set the property `ignite.query.index.h2.enabled=false` and add the required Apache Calcite index JAR files at the Ignite indexing location.

Note: This property is not applicable for external Ignite functions.

External Ignite

In case of external Ignite, the default indexing is H2. If the external ignite server indexing is set to Apache Calcite, then use the following function to switch indexing to Apache Calcite at the client side too:

```
@com.tibco.be.model.functions.BEFunction(
       name = "setCalciteIndexing",
       signature = "void setCalciteIndexing (Object igniteConfig)",
       params = {
                 @FunctionParamDescriptor(name = "igniteConfig", type = "Object",
desc = "IgniteConfiguration Object")
       },
       freturn = @FunctionParamDescriptor(name = "", type = "void", desc =
""),
       version = "6.3.1",
       see = "",
       mapper = @com.tibco.be.model.functions.BEMapper(),
       description = "Sets sqlConfigration as Calcite for given
IgniteConfiguration",
       cautions = "none",
       fndomain = {ACTION},
       example = ""
       )
```

Note: Add the required Apache Calcite index JAR files at the Ignite indexing location.

```
BE_HOME\lib\ext\tpcl\apache\ignite\ignite-indexing
```

For configurations to run the external Apache Ignite with Apache Calcite, see the Apache Ignite documentation.

Configuring TIBCO FTL as the Cluster Provider

You can use TIBCO FTL as the cluster provider for your TIBCO BusinessEvents projects. By doing so, you can enhance performance and add security-related advantages offered by TIBCO FTL to your projects.

Before you begin

- Ensure that you have TIBCO FTL installed on your system. Also, configure and start the TIBCO FTL servers. For details, see TIBCO FTL Documentation.
- A CDD file is added to your project, see Adding a CDD File to the Project.

Procedure

1. Open the `be-engine.tra` file located at *BE_HOME*`\bin` and update the following variable:

   ```
   tibco.env.FTL_HOME= to tibco.env.FTL_HOME=<absolute_path_to_TIBCO_FTL_installation>
   ```

2. To use TIBCO BusinessEvents Studio for configuring the cluster or starting the TIBCO BusinessEvents engine, open the `studio.tra` file available at *BE_HOME*`\studio\eclipse\configuration` and update the variable *tibco.env.FTL_HOME=* to *tibco.env.FTL_HOME=<absolute_path_to_TIBCO_FTL_installation>*.
3. In the TIBCO BusinessEvents Studio, open the CDD file for editing.
4. On the **Cluster** tab, select **Cluster Management** and `TIBCO FTL` as the **Cluster Provider**. For more information, see Cluster Management Reference for TIBCO FTL Cluster.
5. Save the CDD file.

What to do next

For TIBCO FTL as the cluster provider, you can have either Cache or Store as Object Management Type.

- Apache Ignite is the default cache provider. See Configuring Apache Ignite as a Cache Provider.
- ActiveSpaces and Apache Cassandra are the store options present for TIBCO FTL as the cluster provider. See:
    - Configuring ActiveSpaces as a Store Provider
    - Configuring Apache Cassandra as a Store Provider

Cluster Management Reference for TIBCO FTL Cluster

You can change the settings for the TIBCO FTL cluster through the Cluster Management Configuration section in the CDD editor of the TIBCO BusinessEvents Studio.

Property	Description
Realm Properties	
FTL Server	URL of the default TIBCO FTL server.

(Continued)

Property	Description
URL	To use the fault-tolerance support, provide multiple server URLs as pipe-separated values.
FTL Cluster Name	Name of the TIBCO FTL cluster formed of TIBCO FTL servers that you are using. The default value is `ftl.default.cluster`.
Security *(Optional)*	
Security Enabled	Select the checkbox to enable secure authentication. The following fields are displayed only if the **Security Enabled** checkbox is selected. • **FTL Username** • **FTL Password** • **Trust Type**
FTL Username	The FTL server identifies itself to an external authentication service using this username credential.
FTL Password	The FTL server identifies itself to an external authentication service using this password credential.
Trust Type	Choose a trust type from the following: • **Trust All** - Select this option when the TIBCO BusinessEvents engine does not need a certificate in its store to connect to the secure TIBCO FTL server. • **Trust File** - Select this option when the TIBCO BusinessEvents engine uses a certificate file for authentication. • **Trust String** - Select this option when the TIBCO BusinessEvents engine uses a trust string for authentication.
Trust File	Mention the path to the certificate file. This field is required when you select **Trust File** in the **Trust Type** dropdown.

(Continued)

Property	Description
Trust String (PEM)	Provide the trust string. This field is required when you select **Trust String** in the **Trust Type** dropdown.

Setting Up Legacy ActiveSpaces as Cluster and Cache Provider

You can use Legacy ActiveSpaces as the cluster provider. When you select Legacy ActiveSpaces as the cluster provider, the object management type is automatically set to Cache and the cache provider is set to Legacy ActiveSpaces.

Note: Since TIBCO FTL is the default cluster provider, you cannot add a CDD with Legacy ActiveSpaces as the cluster provider by using the new CDD file wizard. You must update the CDD file after it is created for Legacy ActiveSpaces.

Before you begin

- Ensure that Legacy ActiveSpaces is installed on your system.
- For installation procedures, see *TIBCO ActiveSpaces Documentation*.
- Ensure that the *AS_HOME* property in all TRA files is set up for Legacy ActiveSpaces installation. For more information, see *TIBCO BusinessEvents Installation Guide*.
- Add a CDD file to your project with any cluster or object management type. For more information, see Adding a CDD File to the Project.

Procedure

1. In TIBCO BusinessEvents Studio, open the CDD file in the CDD editor.
2. In the CDD editor, select the **Cluster** tab.

3. Click **Cluster Management** and select the following values:
    - **Cluster Management Mode** - `Clustered`
    - **Cluster Provider** - `Legacy ActiveSpaces`
4. Update object management configuration fields as per your requirement. For more information, see Legacy ActiveSpaces Cache OM Settings Reference.
5. Save the resource.

What to do next

- Set up the configurations to persist data. See Persistence Options for more information.
- Set up how to manage objects for cache or store object management. See Domain Objects Configuration for more information.
- Set up the processing units and agents for the project. See Collections, Agent Classes, and Processing Units for more information.

Configuring Limited Cache and Object Table Cache Options

As desired, you can set the cache to limited at the default level and unlimited for specified objects; or you can set the cache to unlimited at the default level and limited for specified objects.

When the cache is limited, the number of cache objects is `limit * no.of cache servers`.

Note: Use of limited cache is supported only when a backing store is used. The backing store retains entries in excess of the limit. Without use of a backing store data inconsistencies could result:

- Entries for an object in the object table (an internally used cache) and in the object cache itself could expire independently of each other.
- Domain object settings for limited cache apply at the object level. Related concepts could have different settings. For example, a container concept could have a limited cache setting and its container concept an unlimited cache setting. Each could be evicted at different times.

Procedure

1. In the CDD editor, select the **Cluster** tab and select **Object Management** node.
2. In the **Entity Cache Size** setting, enter the desired number of objects per entity type.

   See Legacy ActiveSpaces Cache OM Settings Reference.
3. In the **Object Table Cache Size**, enter the desired number of objects (handles) in the object table cache. You cannot set this value differently for different object types.

   See The Role of the Object Table in *TIBCO BusinessEvents Architect's Guide* for more details about the object table.
4. In the navigation section, select the **Domain Objects > Defaults** node.

   Select the **Is Cache Limited** checkbox to enable the limited cache globally. Or clear the checkbox to use an unlimited cache globally.

   See Domain Objects Default Settings Reference.
5. In the navigation section, select the **Domain Objects > Overrides** node. Select an override entry (or add one as needed).
6. Select the **Is Cache Limited** checkbox for the selected object type in one of the following ways:
   - If limited cache is set at the default level, uncheck the overrides **Is Cache Limited** checkbox to use an unlimited cache for objects of this type.
   - If unlimited cache is set at the default level, check the **Is Cache Limited** checkbox to use a limited cache for objects of this type.

     See Domain Object Override Settings Reference.
7. Ensure that multiple clusters do not conflict.

   With Legacy ActiveSpaces clusters, use a different value for Cluster Name (on the Cluster tab, General node) and also use different discovery values.

Cluster Discovery and Internal Communication

When you add a CDD file and select Legacy ActiveSpaces as cluster provider, you must configure how the members of the cache cluster discover each other at runtime and communicate with each other once the cluster is established.

Configure the **Discovery URL** and **Listen URL** fields in Legacy ActiveSpaces cache OM settings, see Legacy ActiveSpaces Cache OM Settings Reference.

For details about both URLs, see:

- Legacy ActiveSpaces Cluster Discover URL
- Legacy ActiveSpaces Cluster Listen URL

Support for Host-Aware Replication

Host-aware replication for Legacy ActiveSpaces is controlled by the property `be.engine.cluster.as.hostaware.enable`. Host-aware replication requires that the member name be a 2-part name separated by a "`.`". For details, see Legacy ActiveSpaces Cluster Configuration Properties Reference.

By default, this property is `true` (or enabled).

Legacy ActiveSpaces Cluster Discover URL

When a cluster starts up, and also when new members join a cluster, a discovery process enables the members to discover each other.

The discover URL specifies how an engine (node) listens for discovery requests from nodes attempting to join the cluster. After the discovery is complete, the members communicate internally using a listen URL (see Legacy ActiveSpaces Cluster Listen URL).

Two types of discovery are available:

- Multicast discovery (PGM), see Multicast (PGM) Cluster Member Discovery.
- Unicast discovery (TCP), also known as "well-known address" discovery, see Unicast (Well-Known Address) Cluster Member Discovery.

Note: A Legacy ActiveSpaces cluster is also known as a *metaspace*.

A TIBCO BusinessEvents engine is a *node* in the metaspace.

If No Other Cluster Members are Started

If a newly started node does not discover any running cluster nodes, the behavior is different depending on the type of discovery used:

- If multicast discovery is used, the newly started node becomes the first node of a newly started cluster.
- If unicast (well-known-address) discovery is used there are two cases:
    - If the listenURL of the newly started node is not in the discover URL's list then it continues to wait for other well-known nodes to start, and a warning is written to the console while it waits.
    - If the listenURL of the newly started node is in the discover URL's list, then it becomes the first node of a newly started cluster.

Multicast (PGM) Cluster Member Discovery

The discover URL for multicast discovery uses PGM (Pragmatic General Multicast) protocol.

Set the URL value in the **Discover URL** field of the Legacy ActiveSpaces Cache OM Settings Reference. For multicast discovery, the value is a URL with the following format: `tibpgm://`*`destinationPort`*`/network/`

The default values equate to the following: `//7888/;239.8.8.9/`

Specify the parameters as follows.

Parameter	Notes
destinationPort	Specifies the destination port used by the PGM transport. Must be the same value on all machines in the cluster. Default value is `7888`.
network	Specifies the IP address of the interface to be used for sending multicast packets, and the multicast group address to be used. The format is as follows: *interface*; *multicast group address* The value for *interface* is unique to a node. It must also be the same in both the discovery and the listen URLs for a node. If there are multiple interfaces on one machine, specify the interface you want to use and do not rely on the default value. The value for *multicast group address* must be the same on all

Parameter	Notes
	machines in the cluster. The default value for *interface* is the first available interface provided by the operating system hosts file for the machine. **Note:** If the desired interface is not listed in the hosts file then PGM picks the first available interface in the file. (On most operating systems, this file is called the `/etc/hosts` file.) If the first interface is the loopback interface (127.0.0.1) then PGM fails to start. In this case you would see a stacktrace exception in the log file such as the following: `SYS_ERROR (multicast_error - (8) grp_iface not a valid multicast interface)` To resolve this issue, either modify the hosts file, or provide the desired interface explicitly in the *network* argument. The default value for *multicast group address* is the multicast group address `239.8.8.9`.

Unicast (Well-Known Address) Cluster Member Discovery

If you cannot or do not wish to use multicast discovery in your environment, then configure unicast discovery, also known as "well-known address" or WKA discovery.

These "well-known addresses" enable a newly started node to discover existing members. Unicast discovery uses the TCP protocol.

Set the URL value in the **Discover URL** field of the Legacy ActiveSpaces Cache OM Settings Reference. For unicast discovery, the value is a semicolon-separated list comprising a sub-set of all the listen URLs (which are different for each PU), using this format:

```
tcp://ip:port[;ip:port]*/
```

Note:
One cluster node in the WKA list must be running at all times

At least one cluster node specified in the discovery URL must be running at all times, so that other new members can join the cluster (metaspace). If all nodes specified in the discovery URL stop, then other nodes that are still running continue to function, but they print warnings to the console and no new members can connect to this cluster.

For WKA discovery, make discover URL a cluster-level property and listen URL a PU-level property

You can define discoveryURL and listenURL in the Legacy ActiveSpaces Cache OM Settings in TIBCO BusinessEvents Studio CDD editor. You can add the discover URL property (`be.engine.cluster.as.discover.url`) and the listen URL property (`be.engine.cluster.as.listen.url`) to overwrite configured default settings.

Legacy ActiveSpaces Cluster Listen URL

The listen URL is used for direct communication between the members of the metaspace. It is configured the same way for multicast and for unicast discovery.

The listen URL value must be different for each cluster member, so configure it at the PU level.

The listen URL uses this format: `tcp://`*`interface`*`:`*`port`*`[-`*`EndPort`*` |*]/`

The cluster member binds to the specified interface and the specified port when creating the TCP socket. Specify the parameters as follows.

Parameter	Notes
interface	To specify a value, use the desired IP address. The value for *interface* must be the same in both the discovery and the listen URLs for a node. If there are multiple interfaces on one machine, specify the interface you want to use and do not rely on the default value.

Parameter	Notes
	The default value for *interface* is the first available interface provided by the operating system for the machine.
port	To specify a single port use the port number in the listen URL, as shown in this example: `tcp://interface:6000/` You can use an auto-incrementing feature, as explained in Auto-incrementing Within a Range of Ports . The default value is the first available port in the `50000+` range.

Multiple Nodes on One Machine

If multiple nodes (engines) are running on one machine, identify each uniquely. Use the same value for *interface*, but a different value for *port* for each node.

Auto-incrementing Within a Range of Ports

If a machine has blocked some ports in the default range, to use a different range, you can configure the listen URL to start with a specified IP address and port, and optionally provide an upper limit. If the specified port is not available, TIBCO BusinessEvents auto-increments the port until it finds an available port, up to the specified upper limit, if any. To specify a specific range use this format:

```
tcp://interface:port-EndPort/
```

For example, given the following listen URL, TIBCO BusinessEvents attempts to open port 8000 and if it is not available it tries the next port number, until it finds an available port, up to 9000 (inclusive). If none is available, it keeps retrying. Make some ports in the specified range available so that the cluster nodes can start.

```
tcp://interface:8000-9000/
```

To specify a range with the upper limit of unsigned short minus one, use this format:

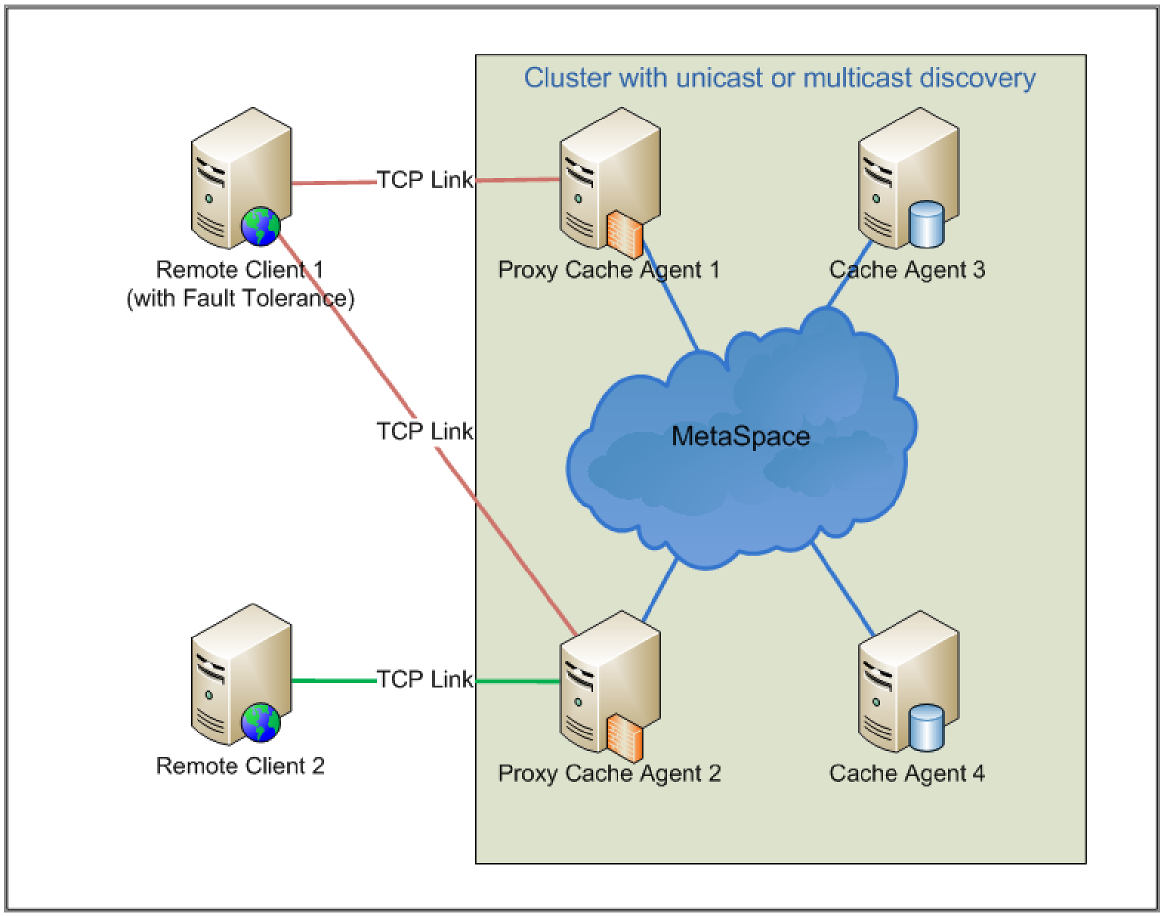
```
tcp://interface:port-*/
```

Remote Client to a Legacy ActiveSpaces Cluster

A remote client acts as a node without actually being a member of the cluster. Instead of being directly connected to space, it is connected through a proxy - typically through a cache-agent.

A remote client does not contribute any of its resources towards maintaining the cluster. TIBCO BusinessEvents extends the same feature to allow its non-cache agents to connect to the cluster as remote members through a cache agent, that acts as a proxy. Using the remote client you can do better data management, as remote clients do not take part in cluster and thus dropping of one or more nodes from the cluster do not affect its processing.

Figure 2: Remote Client Architecture



Remote Client Behavior

- Only the non-cache (like inference) nodes can be remote clients.
- A remote client cannot contribute to cache storage. Thus, the **Enable Cache Storage** checkbox on the **Processing Unit** tab in CDD is ignored.
- Remote clients only operate as long as the cluster is up and reachable. A remote client cannot resume operations across cluster restart.

Best Practices

- For fault-tolerance of proxy nodes, open a remote listen port on two or more of your cache agents and specify a list of these nodes in remote discovery URLs. Thus, the chance of remote client losing connections to the cluster is minimized.
- To ensure consistency in deployment, either configure all inference nodes as remote client or none of them as remote. A mix of remote and non-remote connections is not recommended.

Connecting an Inference Agent as a Remote Client to a Legacy ActiveSpaces Cluster

You can configure one or more cache agent instances which play the role of the seeder and a proxy server for remote clients. On each of the cache node instances that act as a proxy for the remote clients, a remote listen URL (for the remote clients or Inference agents to connect) is configured. In the remote client, specify the list of IP addresses and ports of remote proxy cache agents to connect to.

Before you begin

Ensure the cluster provider is set to Legacy ActiveSpaces, see Setting Up Legacy ActiveSpaces as Cluster and Cache Provider.

Procedure

1. In TIBCO BusinessEvents Studio, open CDD of the cache agent, which is part of the cluster.

2. On the **Cluster** tab, select **Object Management** and specify the IP address and TCP port number to open remote proxy in the **Remote Listen URL** field.

   Selecting IP address and port number is similar to specifying *interface* and *port* number for the DataGrid Listen URL.

   > **Note:** In case, multiple cache agents are running on the same machine, override the remote listen URL with a different port for all such agents. You can override the port either by using the `be-engine` command line or the TIBCO BusinessEvents Enterprise Administrator Agent UI.
   >
   > ```
   > be.engine.cluster.as.remote.listen.url=tcp://<ip>:<port>
   > ```

3. Save the project and restart the agent.

4. In TIBCO BusinessEvents Studio, open CDD of the remote inference agent for editing.

5. On the **Processing Units** tab, add the `be.engine.cluster.as.discover.url` property for the processing unit configured as remote client. Override this property with URL of the proxy cache agent to connect as a remote client.

   ```
   be.engine.cluster.as.discovery.url=<discoveryURL>?remote=true&num_
   connections=2
   ```

   > **Note:** For fault tolerance, make two or more of your cache agents as remote proxies. To do so, open a remote listen port on two or more of your cache agents. Specify a list of these nodes in the remote discovery URLs of the remote client inference agents. This minimizes the chance of remote client losing connections to the cluster in case one of the proxy agent stops working.

   You can specify the time for which the cluster waits for remote clients to reconnect after it is disconnected using the property `be.engine.cluster.as.remote.member.timeout`.

6. Save the project and restart the agent.

Legacy ActiveSpaces Cluster Transport Security

Transport-level security allows you to protect data being transported within the cluster by preventing alteration of traffic, eavesdropping, and exchange of data between untrusted parties.

The available settings for `transport_security` are:

encrypted_normal

Use a secure transport with 128-bit symmetric key encryption (default).

encrypted_strong

Use secure transport with 256-bit symmetric key encryption.

integrity

Use secure transport without encryption.

The two possible node types in a secure Legacy ActiveSpaces cluster are:

Controllers

Nodes dedicated to enforcing a security domain's defined security behavior for a cluster associated with the security domain. Security domain controllers are the only discovery nodes in a cluster.

Requestors

Nodes that require access to the data in the cluster, such as a seeder or a leech, and which need to be authorized by a controller. Requestors can never be used to a discovery node.

Authentication

The controller nodes or processing units are configured with a security policy file. The requester nodes or processing units provide a token file and additional credentials to the controller for authentication. The controller performs authentication as defined in its policy file and using the credentials provided by the requester.

For details on security and authentication in Legacy ActiveSpaces, see TIBCO ActiveSpaces version 2.x Documentation.

Setting Up Legacy ActiveSpaces Cluster Security and Authentication

When security is used, any transmission of messages within the Legacy ActiveSpaces cluster occurs on a secure transport. A security domain's transport security setting controls the level of security used for communication within the Legacy ActiveSpaces cluster.

Before you begin

Ensure that the cluster provider is set to Legacy ActiveSpaces, see Setting Up Legacy ActiveSpaces as Cluster and Cache Provider. Also, ensure that the cluster is set to use TCP-based discovery Legacy ActiveSpaces Cluster Discover URL.

Procedure

1. Start TIBCO BusinessEvents with a non-secure Legacy ActiveSpaces cluster.

   To enable security for the Legacy ActiveSpaces cluster, you first have to configure the cluster to use TCP-based discovery (it cannot use multicast discovery).

2. Create a security policy and security token file by using the Legacy ActiveSpaces admin utility. Ensure that there is a `metaspace_access` entry with the cluster name specified in the CDD file. For details, see TIBCO ActiveSpaces version 2.x Documentation.

3. Shut down the cluster.

4. Open the project CDD file for editing.

5. Each processing unit (PU) is either a Controller or a Requester. You can change its role on the **Processing Unit** tab using the `be.engine.cluster.as.security.mode.role` property to Requestor or Controller. settings. By default, TIBCO BusinessEvents assumes all nodes to be requesters. However, every cluster must have at least one controller node.

   By default, all PUs are requesters so at least one PU in the cluster needs to be a controller. You can override the cluster level controller or requester settings in the PU by checking the **Override** checkbox and specifying a value. In most cases, you would only need to override the key file paths.

6. On the **Cluster** tab, select the **Security Enabled** checkbox for the **Object Management**, to enable the security.

7. Based on the role of PU as Controller or Requestor update their security settings: Supply the security file based on the security with the following properties in the CDD:

   - For the Controller, specify the path of a security policy file and the password for its key in the **Policy File** and **Policy File Identity Password** fields.
   - For the Requestor, specify the path of the security token file and the password for its key in the **Token File** and **Token File Identity Password** fields.

     Requester settings are dependent on the authentication policy defined in the controller's policy file.

     - If the authentication type in the policy file is "userpwd" and the authentication source is "system" or "LDAP", specify **Username** and **Password**. You might also need to specify **Domain**, if the authentication source is "system".
     - If the authentication type in the policy file is "x509", which means that the authentication source is an LDAP configured with certificate-based authentication, then specify **LDAP Identity File** and **Password** (in this case the password is for the private key in the identity file).

Working with an Example

The example for setting up policy and token files shows two nodes in a cluster called `mycluster` . One node is an inference engine and the other is a cache engine.

Procedure

1. Assume that the ListenURL for the cache (controller) is `tcp://203.0.113.0:9091` and the ListenURL for the inference (requester) is `tcp://203.0.113.0:9090`.
2. Generate the policy and token files from `as_admin`.

   Edit the `metaspace_access`line in both files to set the correct cluster name. For details, see TIBCO ActiveSpaces version 2.x Documentation.

3. Also edit the line to place the ListenURL of the controller into the `discovery= setting`.

   ```
   metaspace_
   access=metaspace=mycluster;discovery=tcp://203.0.113.0:9091;
   ```

4. For the inference engine, you can then set the following properties:

   ```
   be.engine.cluster.as.security.enable=true
   be.engine.cluster.as.security.mode.role=Requester
   be.engine.cluster.as.security.file=C:/temp/mytoken.txt
   ```

5. For the cache engine, you can then set:

   ```
   be.engine.cluster.as.security.enable=true
   be.engine.cluster.as.security.mode.role=Controller
   be.engine.cluster.as.security.file=C:/temp/mypolicy.txt
   ```

   Note: There is no rule that caches have to be controllers, or inferences have to be requesters. The roles are interchangeable as long as there is one controller in the cluster.

Schema Model Migration with Shared Nothing Persistence

Using Shared Nothing persistence to migrate a schema model.

You can choose one of these options:

- Deploy the new EAR with the additional field and restart all nodes. The system automatically alters the space as needed during recovery.

- If it is not desired to restart the cache nodes, shut down all inference engines. Connect with the TIBCO ActiveSpaces administrator and alter the space to add new fields as shown in the example below. For details on the `alter space` command, see TIBCO ActiveSpaces version 2.x Documentation.

  The new field must be nullable as is the case for all user fields in BusinessEvents.

  ```
  alter space name"dist-unlimited-bs-readOnly-Test--be_gen_Concepts_
  Simple" add (field name "long_field" type 'LONG' nullable true)
  alter space name"dist-unlimited-bs-readOnly-Test--be_gen_Concepts_
  Simple" add (field name "con_cept_array" type 'BLOB' nullable true)
  ```

  Then start the inference engines.

- Hot deployment of new properties for Shared Nothing persistence has been added.

Note: Adding a concept property of type `contained concept` with the contained concept type set to an existing concept is supported only for the options above.

Hot Deployment of New Properties

Hot deployment of new properties into existing TIBCO BusinessEvents concepts is available only in these cases:

- When the cache Object Management is enabled with no persistence or *Shared Nothing* persistence
- When the concepts with new properties are cache-only

Legacy ActiveSpaces Cache OM Settings Reference

To set up the Legacy ActiveSpaces cache, you can configure the object management settings in the CDD file. All settings can use global variables.

For the related procedure, see Setting Up Legacy ActiveSpaces as Cluster and Cache Provider.

Legacy ActiveSpaces Cache OM Settings

Property	Description
Cache Agent Quorum	Specifies a minimum number (quorum) of storage-enabled nodes that must be active in the cluster when the system starts up before the following occur: • Data is preloaded from the backing store, if a backing store is configured and preloading is configured. • The other agents in the cluster become fully active. The property does not affect the running of the deployed application after startup (though a message is written to the log file if the number of cache agents running falls below the number specified in this property). As a guideline, set the number of cache agents are configured. Default is `1`.
Number of Backup Copies	The number of back-up copies (also known as the back-up count) specifies the number of members of the distributed cache service that hold the back-up data for each unit of storage in the cache. Recommended values are 0, 1, or 2. A value of 0 means that in the case of abnormal termination, some portion of the data in the cache is lost. A value of N means that if up to N cluster nodes terminate at once, the cache data is preserved. A back-up count of 1 means that one server plus one backup is needed, that is, two cache agents (or storage-enabled nodes if cache agents are not used). To maintain the partitioned cache of size M, the total memory usage in the cluster does not depend on the number of cluster nodes and is in the order of M*(N+1). See Asynchronous Replication of Cache Objects for details on replication behavior and options for Legacy ActiveSpaces cluster. Default is `1`.
Entity Cache Size	Specifies the size of the limited cache, in the number of cache entries for each object type. The setting is per processing unit. See Configuring a Limited (or Unlimited) Cache. The default is `10000` (entries per object type)

Property	Description
Object Table Cache Size	Specifies the maximum size of the object table cache, in number of entries. Used with limited cache only. See The Role of the object table in TIBCO BusinessEvents Architect's Guide for more details about the object table. Also see Configuring a Limited (or Unlimited) Cache. Default is `100000` entries
Discovery URL	Specifies how an engine (node) listens for discovery requests from nodes attempting to join the cluster. The PGM protocol is supposed for multicast discovery. TCP Protocol is supported for unicast discovery
ListenUrl	The discovery mechanism is based on pure TCP. All the designated well-known metaspace members are identified by an IP address and a port number.
Remote Listen URL	Specifies on which IP address and TCP port this proxy metaspace member is listening for the remote client connections.
Protocol Timeout	Indicates the protocol timeout value for space. The protocol can be unicast or multicast. The default value is `-1` (forever).
Read Timeout	Indicates the read timeout value for the space, if a read timeout has been set. It specifies the read timeout value for a specified `SpaceDef`. The read timeout value applies to Get operations. The default value is `60000` (ms).
Write Timeout	Indicates the write timeout value for the space, if a write timeout has been set. Specifies the write timeout value that is set for the space. The write timeout value applies to `Put`, `Take`, `Lock`, and `Unlock` operations. The default value is `60000` (ms).
Lock Timeout	For a space that is locked, it specifies how long a member process waits for it to become unlocked.

Property	Description
	The default is `-1` (forever). Other valid values are **0** or any positive value.
Shutdown wait	Indicates the shutdown wait value for the space. The default value is `8500` (ms).
Worker Thread Count	Indicates the thread count specified for the space. The default value is `4`.
Security Enabled	Enables Transport level security for the DataGrid when selected. The following fields are displayed only if the **Security Enabled** checkbox is selected: • Policy File • Policy File Identity Password • Token File • Token File Identity Password • LDAP Identity File • Domain • Username • Password
Policy File	Absolute path to the policy file that contains the security settings that the controller node enforces. It is generated using the `as-admin` utility.
Policy File Identity Password	The password for the identity key in the security policy file.
Token File	Absolute path to the token file that is used by the requestor to connect to a metaspace whose security is defined in the policy file.
Token File	The password for the identity key in the security token file.

Property	Description
Identity Password	
LDAP Identity File	The absolute path for a file containing the key to use for LDAP with the certificate-based authentication.
Domain	Optional. Domain name for system-based user authentication.
User Name	Username for LDAP and system-based authentication.
Password	Password for LDAP and system-based authentication. In case the authentication type in the policy file is "x509" then this is the password for the private key in the **LDAP Identity File**.

Legacy ActiveSpaces Cluster Configuration Properties Reference

Legacy ActiveSpaces cluster properties can be configured using the CDD Cluster Tab. Refer to the following table for a list of properties that you can use to retrieve data from the Legacy ActiveSpaces cluster.

Note: Discovery and listen URL interfaces must match. Ensure that a node's interface (IP address) is specified using the same value in the discover URL and in the listen URL. If there are multiple interfaces on one machine specify the IP explicitly in both properties.

CDD Cluster Tab

Property	Notes
`be.engine.cluster.as.aggregate.prefetch.size`	When queries are executed against DataGrid via the aggregate query functions (found under Query.Datagrid.Aggregate), you can define the number of entries to prefetch for optimum performance. Although TIBCO BusinessEvents has a default value that provides best performance for general use cases, you can adjust the value to meet your specific use case needs. Valid values are any positive long numbers or -1 (-1 = prefetch all). Default value is `-1`.
`be.engine.cluster.as.browser.prefetch.size`	When queries are executed against Legacy ActiveSpaces cluster through any 'select' type of queries, you can define the number of entries to prefetch for optimum performance. Although TIBCO BusinessEvents has a default value that provides best performance for general use cases, you can adjust the value to meet your specific use case needs. Valid values are any positive long number or -1 (-1 = prefetch all). Default value is `-1`.
`be.engine.cluster.as.connection.retry.count`	You can specify the number of attempts allowed for the metaspace connection when the discovery node is not available. The default value is `5`.
`be.engine.cluster.as.lookup.prefetch.size`	When queries are run against Legacy ActiveSpaces cluster by using getByExtByUri() or by loadByExtIdByUri() functions (which return either 1 or no results), you can define the number of entries to prefetch for optimum performance. Although TIBCO BusinessEvents has a default value that provides best performance for most use cases, you can adjust the value to meet your specific needs. The default value is `0`(zero), which indicates that prefetch is disabled.

Property	Notes
`be.engine.cluster.as.member.timeout`	The timeout parameter specifies how many milliseconds DataGrid will wait for a member to reconnect, if it loses connection to the metaspace. The default value is 30000 milliseconds.
`be.engine.channel.as.querylimit`	You can control the query limit for an ActiveSpaces channel using the `querylimit` property. You can set this property in CDD so that the channel can receive entries beyond 10000 (default in TIBCO ActiveSpaces). The default value in TIBCO BusinessEvents is -1, which indicates no limit on queries.
`be.engine.cluster.as.suspend.threshold`	The threshold parameter specifies the number of host connections that can be lost before the cluster moves into a suspended state. When the cluster is suspended, members cannot leave or join the cluster. If connectivity is lost for a seeder member of a space, doing a read or write for the space might cause a protocol timeout. The default value is `-1`, which indicates that the cluster is never suspended.
`be.engine.cluster.as.file.sync.interval`	The amount of time (in milliseconds) to wait between persists to the data store when asynchronous shared-nothing persistence is used. The set value can be viewed as the `File Sync Interval` property value in `as-admin`. The default value is 10000 milliseconds.
`be.engine.cluster.as.discover.url`	The discover URL specifies how an engine (node) listens for discovery requests from nodes attempting to join the cluster. PGM protocol is supported for multicast discovery. TCP protocol is supported for unicast (well-known address) discovery. Configuration is different for multicast and unicast discovery. See Legacy ActiveSpaces Cluster Discover URL for details. The default value for multicast equates to: `tibpgm://7888/;239.8.8.9/`
`be.engine.cluster.as.hostaware.enable`	By default, this property is true (or enabled). • If `true`, the Legacy ActiveSpaces cluster member name will be set as: `hostname.be-engine-name` where be-engine name is what is given on the -n command line option. When host-aware replication is enabled, if the cache nodes are not deployed on multiple machines to satisfy replication by the host, then replication will not happen (or will happen only according to the number of hosts available). For example, if **Number of Backup Copies** is set to "1" and all cache nodes are deployed on a single host, then replication will be disabled (regardless of the number of cache nodes on that single host). If **Number of Backup Copies** is set to "2", and cache nodes are deployed only on 2 hosts, then only "1" backup copies will be maintained. • If the property is `false`, host-aware replication will be disabled and the Legacy ActiveSpaces member name will be set as be-engine-name. Disabling host-aware replication will honor **Number of Backup Copies**, provided that there are enough cache nodes deployed in the cluster.
`be.engine.cluster.as.hostaware.hostname`	Hostnames that are used in identifying members (and therefore naming Shared nothing file/folders), are generated from underlying OS. If you would like to assign hostnames manually instead, for reasons such as hostname/machine changes, testing so on, then provide hostnames in the CDD at each PU level using `be.engine.cluster.as.hostaware.hostname` property.

Property	Notes

`be.engine.cluster.as.listen.url`	The listen URL is used for direct communication between the members of the metaspace after the discovery process. The listen URL uses this format:
	`tcp://interface:port/`
	`You can also use an auto-incrementing feature by specifying a range as follows:`
	`tcp://interface:Port-[toPort  | * ]/`
	The default value for *`interface`* is the first available interface provided by the operating system for the machine. See Legacy ActiveSpaces Cluster Listen URL for details.
	The default value for port is the first available port in the 50000+ range.

`be.engine.cluster.as.remote.listen.url`	Specifies on which IP address and TCP port this proxy metaspace member is listening for the remote client connections. The remote listen URL uses the following format: `be.engine.cluster.as.remote.listen.url=tcp://interface:port`
`be.engine.cluster.as.remote.member.timeout`	Specifies the timeout for remote clients. This is the duration for which the cluster waits for a remote member to reconnect after it got disconnected. If the remote member does not reconnect within this duration, the remote member is considered as disconnected from the cluster. The default value is 120000, that is, 2 minutes.
`be.engine.cluster.as.minSeeders`	TIBCO BusinessEvents sets the value of minimum seeders for user-defined spaces to the same value as the quorum size, by default. To override the default value of minimum seeders, you can specify the new value using this property.
`be.engine.cluster.as.shutdown.wait.millis`	Specifies time (in milliseconds) to wait for the thread that uses ActiveSpaces to complete before shutdown.
`be.engine.cluster.minCacheServers.strict`	When this property is set to true and if the number of Cache nodes drops below Quorum, then the cluster is placed into suspend mode. **Note:** When the system is actively processing messages, it may take a while for it to reach the suspended state as the Inference agents attempt to commit current transactions and empty the internal queues before suspending the operations. Default value is false.
`be.engine.cluster.minCacheServers.strict.selfRepair`	When this property is set to true, the system tries to resume operations if and only when the Quorum is reached again. If this property is set to false once the operations are suspended, then the system will stay in that mode until you manually intervene. Default value is false. **Note:** This property only applies if `be.engine.cluster.minCacheServers.strict=true`.
`be.engine.cluster.quorumCheck.setLenient`	Using this property, you can change the quorum state behavior. The values are:

Property	Notes
	• true - For the initial startup, a minimum number of cache nodes are required as specified in the quorum count. The quorum state of the cluster is maintained after a quorum is established until the number of cache nodes drop below the minimum number of seeders as specified in the `be.engine.cluster.as.minSeeders`. During this period, new cache nodes can join the cluster and function like the quorum is never lost. The new Inference engine can be started as long as the number of cache nodes is not below the minimum number of seeders. • false - For the initial startup, a minimum number of cache nodes are required as specified in the quorum count. After a quorum is established, the system keeps running till the number of cache nodes drop below the minimum number of seeders as specified in the `be.engine.cluster.as.minSeeders`. Now the new inference agents cannot be started until the number of cache nodes reaches the quorum count. The default value is false for inference engines, and true for cache engines. This property can be used at the agent level as well.
`be.engine.cluster.cacheNaming.isDescriptive`	Specifies whether descriptive information (such as dist/repl, unlimited/limited, nobs/bs, and so on) is included in the shared nothing cache names. The values are: • true - names are descriptive. For example, `C:\temp\sharednothing\TestBQL\dist-unlimited-bs-TestBQL--be_gen_Concepts_Agreement\CSU1\CSU1_store_1440456000`. • false - names are not descriptive. For example, `C:\temp\sharednothing\TestBQL\TestBQL--be_gen_Concepts_Agreement\CSU1\CSU1_store_1440456000`. The default value is true.
`be.engine.cluster.as.security.mode.role`	Security role of a node for the secure DataGrid. Possible values are: *Controller* or *Requestor*. The Controller is dedicated to enforcing security behavior for a cluster associated with the security domain. Security Controllers are the only discovery nodes in a cluster. The Requestor requires access to the data in the DataGrid, which needs to be authorized by the Controller. A Requestor can never be used as a discovery node.
`be.engine.cluster.as.security.file`	Path to the policy (for controller) or token file (for requestor), which contains the security settings, based on role defined in the `be.engine.cluster.as.security.mode.role` property.
`be.engine.cluster.as.security.file.identity.password`	The password for the identity key in the security policy file or token file specified in `be.engine.cluster.as.security.file`.
`be.engine.cluster.as.security.requester.identity.keyfile`	The absolute path for a file containing the key to use for LDAP with the certificate-based authentication.
`be.engine.cluster.as.security.domain`	Optional. Domain name for system based user authentication.
`be.engine.cluster.as.security.username`	Username for LDAP and system based authentication.

Property	Notes
`be.engine.cluster.as.security.password`	Password for LDAP and system based authentication. In case authentication type in the policy file is "x509" then this is the password is for the private key in the LDAP identity file specified in `be.engine.cluster.as.security.requester.identity.keyfile`.
`be.engine.cluster.as.remote.tuple.limit`	This property controls the number of entries or records sent from seeder to client. When volume of data in cache gets very high then querying the spaces results into blocking threads, in such case, you can add this property to send only limited records from seeders to the client. This is generic property which can be applied to all agents including the remote client. The valid values are any positive long numbers and -1. The default value is `-1` which indicates no limit on entries or records.
`be.engine.cluster.event.expiry.lock`	Set this property to `true` to enable cluster-level locking on event extID for the event expiry thread. The default value is `false`.
`be.engine.cluster.as.extid.index`	Set this property to `true` to add an index for `extID`. **Note:** The default value is `true` for the Shared Nothing option.
`be.engine.cluster.as.browser.join`	Set this property to `true` to enable the space join on browse calls during processing.
`be.engine.cluster.as.browser.timeout`	Use this property to configure the timeout for an ActiveSpaces browser. Provide the timeout in milliseconds.
`be.engine.cluster.as.invocation.timeout`	Use this property to configure the timeout for an ActiveSpaces member invocation. Provide the timeout in milliseconds. Default is unlimited timeout.

Persistence Options

After defining the object management type as cache, you can configure how you want to persist the data, either at centralized location or at each node.

The persistence options that you can choose are:

- **None** - If you do not want to persist data, then you can also select `None` as **Persistence Option**.
- **Shared Nothing** - If you do not want to persist data in a central database, see Setting Up Shared Nothing Persistence.
- **Store** - To provide for data persistence, you can implement a backing store (database) for use with Cache OM, see Store Configurations.

Note: Adding a concept property of type `contained concept` with the contained concept type set to an existing concept is not supported for hot deployment. This is applicable for the `None` and `Shared Nothing` persistence.

Asynchronous Replication of Cache Objects

Backup count defines the number of backup object copies to make in addition to the primary cache object.

Backup cache writes can be done synchronously or asynchronously.

Legacy ActiveSpaces and Apache Ignite are set up to use asynchronous replication. There is no option to use synchronous replication. Asynchronous replication allows you to run tests using a single cache agent.

With asynchronous replication, the inference agent writes to a cache agent and returns. The cache provider then makes a separate call to another cache agent to make the replica. This means that the writes from the inference agent do not incur the cost of synchronous replication, because replication happens on a different thread in the background. However, a small window exists in which the inference agent has written to the cache, and the cache provider has not replicated the data yet. If the cache agent fails at this point, data is lost because there is no replica. To safeguard the data, use a backing store with cache-aside database write strategy.

Setting Up Shared Nothing Persistence

Shared Nothing persistence allows you to store data at an individual node level, instead of a centralized location.

Note: When using up Shared Nothing persistence, to ensure that data is not lost when nodes leave the cluster, you must set the number of back-up copies to 1 or more.

New concepts and new concept properties are enabled for hot deployment when using the Shared Nothing persistence.

Note: Adding a concept property of type `contained concept` with the contained concept type set to an existing concept is not supported for hot deployment.

Before you begin

The persistence options are displayed only if the `Cache` is selected as **Object Management Mode**.

Procedure

1. In TIBCO BusinessEvents Studio, in the CDD editor, expand **Object Management** and select **Persistence**.
2. Select the **Persistence Option** as `Shared Nothing`.
3. Enter the value for persistence path as the absolute path to the directory where the data is to be stored.

   Note: The Persistence Policy is by default set to ASYNC and cannot be changed, see Asynchronous Replication of Cache Objects.
4. Save the CDD file.

Runtime Configuration to Specify the Engine Name Property

The TIBCO BusinessEvents engine name is used to name the Shared Nothing root directory.

To enable the engine name to be used, you need to explicitly set the engine name property when running clusters with Shared Nothing.

The `-n <name>` must be specified for all the nodes (inference and cache) in the cluster.

Note that only one node will be started in the following cases:

- If the cache nodes in your cluster do not specify any names, the hostname of the machine on which the engine is running is used as the default engine name.
- If the cluster contains cloned cache nodes (nodes with the same engine name), only one cloned node will start.

Nodes with duplicate engine names will not be initialized.

Recovery Options for Shared Nothing Persistence

You can use five policies for the shared nothing persistence as recovery options.

The `be.engine.cluster.recovery.distributed.strategy` parameter is only supported for the shared nothing persistence. When shared nothing persistence is implemented and recovery is issued, then the policy determines how and when the recovery can be made. The default value of the property is `no_data_loss`.

Policy	Policy Description
`no_data`	Recovers the space without any data. This is the same as removing shared nothing persistence files.
`data_loss`	Recovers the space with available data from each seeder. If recovery is done with missing seeders, there is a potential for data loss, because not all members are started, to ensure that all data is recovered. This policy ensures best-effort recovery with the available data.

Policy	Policy Description
`no_data_loss`	(Default). Recovers the space only if there are enough members available to recover the data but fast-batch mode replication is not possibel.to be able to recover the previous state of the cluster. Otherwise recovery generates an exception.
`fast_load_only`	Recovers the space only if the `no_data_loss` conditions are met and replica entries can be distributed among cluster members in fast-batch mode. When the `fast_load_only` is used, it is advisable to set 'Cache Agent Quorum' to the total number of cache nodes. This increases the chances of a successful recovery.
`robust_load_only`	Recover the space only if there are enough members to be able to recover the data before shutdown. If enough seeders are not available to recover the previous state of the cluster, recovery generates an exception. This policy forces the slow recovery of the space.
`force_load`	Forces recovery of the space even if the old shared nothing persistence files are renamed as per the new setup and hostname. This policy bypasses the required host, seeder checks and loads the data anyway to complete recovery from old shared nothing files.

Note: (Legacy ActiveSpaces Only) The `be.engine.cluster.as.minSeeders` property's value is the Cache Agent Quorum value minus the number of back-up copies. The seeder information (the current seeder list) is stored in the shared nothing persistence files during shutdown. This information is then used during startup or restart, to perform recovery. If the cluster is the same during startup (that is, the same members are available and quorum is satisfied), then fast loading of the data is performed.

Store Configurations

To provide data persistence, you can implement a store for use with cache (Cache OM) or without cache (Store OM).

In Cache OM, the cache data is written to the backing store. On system restart, data in the backing store is restored to the cache cluster.

To implement a store, provide a supported database product. Scripts are provided to set up the database for your project's ontology. If the ontology changes, scripts help you adapt the store accordingly and existing store data can be preserved.

Configuring Backing Store for Cache OM

This topic provides the information on configuring the following backing stores:

- Oracle
- SQL Server
- DB2
- MySQL
- PostgreSQL

Additional backing stores are available only for Apache Ignite cache:

- ActiveSpaces, see Configuring ActiveSpaces as a Store Provider.
- Apache Cassandra, see Configuring Apache Cassandra as a Store Provider.

Before you begin

In case of a RDBMS database, ensure to setup the JDBC Connection shared resource. For details about adding a shared resource, see *TIBCO BusinessEvents Developer Guide*.

Procedure

1. Open the project CDD file in TIBCO BusinessEvents Studio and on the **Cluster** tab, configure the cluster and cache provider as per your project requirements.

Cluster and Cache Provider	**Reference Topic**
Legacy ActiveSpaces cluster and cache	Setting Up Legacy ActiveSpaces as Cluster and Cache Provider
TIBCO FTL cluster and Apache Ignite cache	• Configuring TIBCO FTL as the Cluster Provider • Configuring Apache Ignite as a Cache Provider
Apache Ignite cluster and cache	• Configuring Apache Ignite as the Cluster Provider • Configuring Apache Ignite as a Cache Provider

2. In the navigation tree, expand **Object Management** and select **Persistence**.
3. In the Configuration window, from the **Persistence Option** list, select `Store`.
4. From the **Store Type** list, select the backing store as required.
   - Oracle
   - SQL Server
   - DB2
   - MySQL
   - PostgreSQL
   - ActiveSpaces, see Configuring ActiveSpaces as a Store Provider.
   - Apache Cassandra, see Configuring Apache Cassandra as a Store Provider.

5. In the navigation tree, select **Object Management > Persistence > Connection** and in the configuration section, configure the settings mentioned in the JDBC Backing Store Connection Settings.

   You can start with default pool values and monitor the behavior. Using more connections improves runtime performance and can also speed up recovery in the event of a failure.

JDBC Backing Store Connection Settings

Property	Description
URI	Specifies the JDBC project path, that is, the path from the project root to the JDBC Connection resource, to define the connection to the backing store. For example: `/SharedResources/JDBC Connection.sharedjdbc` You can also use a global variable to specify the connection. Default value is *%%DbUri1%%*. To create a of JDBC Connection shared resource, see *TIBCO BusinessEvents Developer Guide*.
Min Size	Minimum number of connections in the JDBC connection pool used for the backing store.
Max Size	Maximum number of connections in the JDBC connection pool used for the backing store. Connections do not exceed the maximum. The value of this property overrides the value of the Maximum Connections setting in the JDBC Connection resource. Although the limit is seldom reached, you can guarantee a connection is always available for a `dbwriter` thread as follows. Set this field to the same value as the `Agent.`*`AgentClassName`*`.dbthreadcount` setting.

Property	Description
Initial Size	Specifies the initial size of the JDBC connection pool used for the backing store, when it is created on startup.

6. In the navigation tree, select **Properties** and add properties based on your requirement. For details about of available properties, see CDD Cluster Tab Backing Store Properties Reference.
7. Save the CDD file.

What to do next

You can select domain objects (entities) to be included in or excluded from the backing store. In addition preloading options are available for loading domain objects from backing store to cache at system startup. See Domain Objects Configuration.

- See Configuring Preloading Options for object settings
- See `be.engine.cluster.recovery.threads` in CDD Cluster Tab Backing Store Properties Reference.

CDD Cluster Tab Backing Store Properties Reference

Use this reference for the backing store properties.

For the related procedure, see Configuring Backing Store for Cache OM.

CDD Cluster Tab Backing Store Properties

Property	Notes
Database Connection Properties	
`be.backingstore.recreateOnRecovery`	Set this property to true if the database pool size does not recover to the initial or minimum connection size, as defined by the Min Size and Max Size properties (in Configuring Backing Store for Cache OM). The default value is `false`.
Other properties	
`be.backingstore.useobjecttable`	The property when set to `true` provides mappings for all entities in the cache. Object table is used to find the actual object either in the cache or in the backing store. When this property is set to `false`, you must use the catalog functions with the "byURI()" pattern so that entities are found from the cache. The default value is `true`.
`be.backingstore.optimize.reads`	Used with Microsoft SQL Server only. Set the property to true to improve the runtime performance. Use `NOLOCK` for `SELECT` statements to avoid locks on the database or table when `SELECT` statements are issued. An example syntax is: `select * from dbo.D_MailerIndex with (nolock) where ...`
`be.backingstore.optimize.writes`	Used with Microsoft SQL Server only. Set the property to true to improve the runtime performance. Use `ROWLOCK` with `UPDATE` or `DELETE` statements to avoid lock contentions. When you use `ROWLOCK` in the T-SQL statement, the SQL Server locks only the rows that match the 'where' condition and not the entire table. An example syntax is: `DELETE FROM dbo.D_Mailed WITH (ROWLOCK) where mailernumber = '12345678895' and time_created$ = 'somedate'`

Property	Notes
`be.backingstore.timestamp.useDataTimeZone`	Used when the backing store is enabled. Set the property to true to ensure that the correct DateTime properties are retrieved when an agent’s time zone changes and the agent restarts.
`be.engine.cluster.recovery.threads`	Recovery threads are used to when pre-loading the cache during startup. For an explanation of pre-loading and other pre-loading controls, see Domain Objects Configuration. Default is `5`.
`be.engine.cluster.recovery.distributed.strategy`	This property is used for Shared All and for Shared Nothing persistence. • For Store: Possible values for this property are `batch` and `nobatch`. `Batch` mode is a distributed batch mode, where one cache node is the 'director' and gives jobs to other nodes while recovering data from the backing store. Therefore, more than one node is needed. All nodes need to be started at once so jobs are distributed evenly. While in `nobatch` mode, each node tries to pick up a job by itself while recovering data from the backing store. Default is `nobatch`. **Note:** By using the JMX MBeans > Pre-load and Recovery Information, you can view which seeders are performing recovery depending on the strategy mentioned in CDD as either Batch and NoBatch in the JConsole. • For Shared nothing: The `be.engine.cluster.recovery.distributed.strategy` parameter has the five following recovery policies added as part of the recovery options. When Shared Nothing persistence is implemented and recovery is issued, the policy determines when and how recovery can be made. The following are the allowed values for the five recovery policies: ◦ The `no_data` recovers the space without any data. This is the same as removing Shared Nothing persistence files. ◦ The `data_loss` recovers the space with available data from each seeder. If recovery is done with missing seeders, there is a potential for data loss, because not all members are started, to ensure that all data is recovered. This policy ensures best-effort recovery with the available data. ◦ The `no_data_loss` recovers the space only if there are enough members to be able to recover the data before shutdown. If enough seeders are not available to recover the previous state of the cluster, recovery generates an exception. ◦ The `fast_load_only` recovers the space only if all the members that were active before the shutdown are available in the cluster. This policy enables fast recovery. If there are more or less cluster members than before the shutdown, recovery generates an exception. When the fast_load_only is used, it is advisable to set **Cache Agent Quorum** to the total number of cache nodes. This increases the chances of a successful recovery. ◦ The `robust_load_only` forces slow recovery. The default value is: `no_data_loss`.

Property	Notes
	Note: The seeder information (the current seeder list) is stored in the Shared Nothing persistence file during shutdown. This information is then used during startup or restart, to perform recovery. If the cluster is the same during startup (that is, the same members are available and quorum is satisfied), then fast loading of the data is performed.
`be.engine.cluster.recovery.distributed.batchsize`	When distributed batch recovery is enabled (`be.engine.cluster.recovery.distributed.strategy=batch`), the recovery manager divides the target table into many smaller batches and assigns the 1-to-n number of these batches to each node (for them to run recovery). This parameter provides an approximation to the size of each such batch. Since batches are defined using the target table's key field (for example, approximated using the ID$ column), actual batch size differs depending on the key distribution. **Note:** In case of the legacy lookup strategy (default), the batch size is a numeric value as set by you to manage the range but with the new key-based lookup strategy (`be.engine.id.useLegacy=false`), the range rather is based on the timestamp. Users can instead define `be.engine.cluster.recovery.distributed.batchpernode=2` and prevent that too few or too many batches are created.
`be.engine.cluster.useDBBatching`	**Note:** For use with cache aside and only when the parallel operations feature is used. This property has no effect if `Agent.`*`AgentClassName`*`.dbOpsBatchSize` is set to 1 (see CDD Agent Classes Tab Properties Reference). This property affects how all RTC transactions that a database writer thread takes from the database operations queue are written to the backing store: • When set to true, the RTC transactions are handled as one job. • When set to false, each RTC transaction is handled as a separate job. For a guide to usage of this and other related properties, refer to the "Database Write Tuning Options for Cache Aside" section in the *TIBCO BusinessEvents Architect’s Guide*. The default value is `false`.
`be.jdbc.multibyte.support`	You can specify whether multibyte characters are supported in the column values. Set the value to `true` for the Microsoft SQL Server where multibyte character support is required. The default value is `false`.

Configuring ActiveSpaces as a Store Provider

You can use ActiveSpaces as the backing store for your TIBCO BusinessEvents application.

You can configure ActiveSpaces as a backing store for TIBCO BusinessEvents applications with only the following cluster and cache combinations:

Cluster	Object Management Type	Cache
Unclustered	Store	No cache
TIBCO FTL cluster	Store	No cache
TIBCO FTL cluster	Cache	Apache Ignite

Before you begin

Install TIBCO ActiveSpaces . ActiveSpaces datagrid should be up and running. For more information, see the TIBCO ActiveSpaces documentation.

Procedure

1. Open the project CDD file in TIBCO BusinessEvents Studio, and on the **Cluster** tab, configure the cluster and cache as per your project requirements.

   See:

   - Configuring TIBCO FTL as the Cluster Provider
   - Configuring Apache Ignite as a Cache Provider

2. Configure the CDD file to use a backing store for the project using either of the following ways:
    - If you are not using any cache with the project, select **Object Management** and set the following values.
        - **Object Management Mode** - `Store`
        - **Store Provider** - `ActiveSpaces`
    - If you are using Apache Ignite as the cache for the project, select **Object Management > Persistence** and set the following values.
        - **Persistence Option** - `Store`
        - **Store Type** - `ActiveSpaces`
3. Configure the **Connection** and **Security** fields.

    See: Backing Store Setting Reference for the ActiveSpaces Store .
4. Generate the ActiveSpaces schema for the project.

    See: "Generating Deployment Scripts for a Store" in *TIBCO BusinessEvents Developer Guide*.
5. Import the schema and generate ActiveSpaces project-specific and base tables. The `create_tables_as.tibdg` file is provided located at `<BE_HOME>/bin/`.

    See: "Persistence Store Setup" in *TIBCO BusinessEvents Developer Guide*
6. *(Optional)*Enable authentication for your application in TIBCO ActiveSpaces.

    See: TIBCO ActiveSpaces documentation

Backing Store Setting Reference for the ActiveSpaces Store

To configure ActiveSpaces as a backing store for your TIBCO BusinessEvents project, you can configure the following connection and security settings in the CDD editor:

Field	Description
Realm URL	URL of the TIBCO FTL Realm server. The default value is http://localhost:5055.
Grid Name	Name of the property that is configured for the data grid name on the ActiveSpaces server. If not provided, the cluster name is taken as the **Grid Name**.
Connection Timeout	The amount of time that calls other than `DataGrid.Connect` wait for a response from the data grid. The default value is `5` second.
Connection Wait Time	This property is supplied to `DataGrid.connect` while connecting to the data grid. This value sets the fixed amount of time a call waits for `DataGrid.connect` while it collects proxy responses. The default value is `0.1` second.
Connection Pool Size	Number of connections in the connection pool. The default value is `10`.
Username	User name for the authorized user configured on the server.
Password	Password for the authorized user configured on the server.

(Optional) To enable security, select the **Security Enabled** checkbox and fill values for the following fields:

Field	Description
Trust Type	Choose the trust type for the project. The values are: • Trust File • Trust Everyone **Default value**: Trust File
Trust File Path	If you select **Trust File** as the **Trust Type**, provide the path to the trust file.

Configuring Apache Cassandra as a Store Provider

You can use Apache Cassandra as a store provider for your TIBCO BusinessEvents application with or without cache.

You can use Cassandra for applications with the following cluster and cache combinations:

Cluster Configuration	Object Management Type	Cache Configuration
Unclustered	Store	No cache
TIBCO FTL cluster	Store	No cache
TIBCO FTL cluster	Cache	Apache Ignite

Before you begin

- Download and install Apache Cassandra. Start the Cassandra server. For more information, see the Cassandra documentation.
- Initialize and create tables on the Cassandra server. Sample `initialize_database_cassandra.cql` and `create_tables_cassandra.cql` files are provided located at *BE_HOME*`/bin`. For more details, see the Backing Store Setup in *TIBCO BusinessEvents Developer Guide*.

Procedure

1. Open the project CDD file in TIBCO BusinessEvents Studio and on the **Cluster** tab, configure the cluster and cache as per your project requirements.

   See:

   - Configuring TIBCO FTL as the Cluster Provider
   - Configuring Apache Ignite as a Cache Provider

2. Configure the CDD file to use a backing store for the project using either of the following ways:
    - If you are not using any cache with the project, select **Object Management** and set the following values.
        - **Object Management Mode** - `Store`
        - **Store Provider** - `Apache Cassandra`
    - If you are using Apache Ignite as the cache for the project, select **Object Management > Persistence** and set the following values.
        - **Persistence Option** - `Store`
        - **Store Type** - `Apache Cassandra`
3. Configure the **Connection** and **Security** fields.

    See: Backing Store Settings Reference for Apache Cassandra
4. Generate a Cassandra schema for the project.

    See: "Generating Deployment Scripts for a Store" in *TIBCO BusinessEvents Developer Guide*.
5. Import the schema and generate Cassandra tables.

    See: "Persistence Store Setup" in *TIBCO BusinessEvents Developer Guide*
6. *(Optional)* Enable one-way (server-side encryption) or two-way SSL (server-side and client-side)authentication for your application in the Cassandra server. For more information, see Cassandra documentation.

Note: TIBCO BusinessEvents does not support transactions when Apache Cassandra is used as a store provider.

Backing Store Settings Reference for Apache Cassandra

To configure Apache Cassandra as a backing store for your TIBCO BusinessEvents® project, you can configure the following connection and security settings in the CDD editor.

With the latest DataStax java driver, the configuration method is changed. Instead of ad hoc configuration classes, the file-based mechanism is used. The `reference.conf` file is packaged in the `java-driver-core` JAR artifact, and used at runtime to provide the default values for all configuration options. This allows configuration changes without recompiling the client application.

You can place an `application.conf` or `reference.conf` in your application’s classpath to override options selectively.

For more details, see: DataStax Documentation.

Field	Description
Server Contact Points	You can provide multiple contact points in the following format: `<host >:<port1>,<host2>:<port2>` The default value of port is `9042`.
Key Space Name	Name of the keyspace to connect with on the Cassandra server. If not specified, the cluster name is used for the keyspace name.
User Name	Username of a keyspace user.
Password	Password for the keyspace user.

(Optional) To enable secure authentication, select the **Security Enabled** checkbox and fill values for the following fields:

Field	Description
Trusted Certificates Location	While using SSL authentication, provide the path to the base folder containing the certificate files.
Trust Store Password	Password for the trust store while using SSL authentication.
Requires Client Authentication	Select this checkbox if you enable two-way SSL authentication.
Identity	Path to identity file while using two-way SSL.

Enabling Advanced Settings For Apache Ignite

If you need to configure advanced properties for your Apache Ignite cache, you can configure the store settings XML file (`store.xml`) to display those advanced settings on TIBCO BusinessEvents Studio.

Procedure

1. Create an XML file and save it with the name as `store.xml` at *BE_HOME*`/lib/ext/tpcl/contrib`.
2. Edit `store.xml` and configure it for the store and add the properties that you want to display in TIBCO BusinessEvents Studio.

   For details on the structure of the `store.xml` file, see Structure of the store.xml File.

   For list of the supported advanced properties, see Cache and Store Advanced Properties.
3. Save the `store.xml` file.
4. *(Optional)* You can bundle `store.xml` in a JAR file as well. Save the `store.xml` file at the base of the JAR file.

   This helps to add multiple `store.xml` files corresponding to cache and store. The JAR name can signify the cache or store type, for example, `ignite_adv_properties.jar` or `cassandra_adv_properties.jar`
5. Save the JAR file at *BE_HOME*`/lib/ext/tpcl/contrib`
6. Restart TIBCO BusinessEvents Studio.

Result

New and updated fields are displayed on TIBCO BusinessEvents Studio based on the respective cache or store selection.

Structure of the store.xml File

In TIBCO BusinessEvents Studio, based on the object management selection, the matching store provider `store.xml` file is picked and form-based UI is generated listing the configuration properties from `store.xml`.

Sample store.xml File

The following is a sample of `store.xml` to add advanced properties for the Apache Ignite store.

For list of the tags that you can use, see XML Tags of the store.xml File.

For a complete list of advanced properties for Apache Ignite cache, see Cache and Store Advanced Properties.

```
<store>
       <type>Ignite</type>
       <label>Apache Ignite</label>
       <class>com.tibco.cep.ignite.store.IgniteStoreProvider</class>
       <schema-generation></schema-generation>
       <description>Apache Ignite</description>
       <version>1.0</version>
       <backingstore>false</backingstore>
       <cache>true</cache>
       <ttl>true</ttl>
       <properties>
       <property-group name="Connection" displayName="Connection">
       <property name="wal-segment-size" displayName="wal segment size"
type="Integer" default="67108864"   mandatory="false"/>
       <property name="write-throttling-enable" displayName="write throttling
enable  " type="Boolean" default="true" mandatory="false"/>
       <property name="wal-mode" displayName="wal mode" type="String"
default="LOG_ONLY" mandatory="false"/>
       <property name="wal-store-path" displayName="wal store path"
type="String" default="" mandatory="false"/>
       <property name="data-region-checkpoint-page-buffer-size"
displayName="checkpoint-page-buffer-size" type="Integer"
       default="" mandatory="false"/>
       <property name="wall-archive-after-inactivity" displayName="wall-
archive-after-inactivity" type="Integer" default="" mandatory="false"/>
       <property name="checkpoint-threads" displayName="checkpoint-threads"
type="Integer" default="" mandatory="false"/>
       <property name="checkpoint-write-order" displayName="checkpoint-write-
order" type="String" default="RANDOM" mandatory="false"/>
               <property name="metrics-log-frequency" displayName="metrics-log-
frequency" type="Integer" default="0"
               mandatory="false"/>
               </property-group>
       </properties>
</store>
```

XML Tags of the `store.xml` File

The following table lists the major tags available in the `store.xml` file and sample values for Apache Ignite and Apache Cassandra:

The store.xml File Tags

XMLS Tags	Purpose
<store>	Identifies that the enclosed XML content is for the store.
<type>	Sets the type of the store. The values are: • `Ignite` • `Cassandra`
<version>	Version number of the XML file. During the startup, TIBCO BusinessEvents Studio compares the version number of the default (version 1.0) and the custom `store.xml` file and displays settings accordingly. • To only add new properties, you can keep the version number same. • To add new properties as well as override existing properties, increment the version number.
<properties>	Container tag for property groups.
<property-group>	Container group to categorize the properties. You can provide display name to the property group which is displayed on the UI as a section containing all child properties.
<security>	Container group for security-specific properties. This displays a **Security Enabled** checkbox on the UI. The properties under this container are displayed on the UI only when the **Security Enabled** checkbox is selected.
<property>	Define the properties that you want to display on the UI. For each property, you can define the appropriate attribute. To display a list of options for the user, you can insert `<choices>` and `<choice>` tags inside the `<property>` tag. The following are the key attributes for the property tag:

XMLS Tags	Purpose
	• `name` - A unique name for this property, internally, this is used to get property values at the runtime. • `displayName` - The label to be shown in the CDD UI against the property. • `type` - The property type. The following are its valid values: ◦ `Boolean` - Use this to have a checkbox in the CDD UI for the boolean type properties. ◦ `Integer` - Use this to have a textbox with numeric validation for non decimal numeric type properties. ◦ `Double` - Use this to show a textbox with decimal number validation for decimal numeric type properties. ◦ `String` - Use this to show a textbox for String type properties. ◦ `File` - Use this to show a file browser in the CDD UI for properties that hold a file path. • `default` - The default value for this property. • `mandatory` - Use this to make the property mandatory. The valid values are: ◦ `true` - to make the property mandatory ◦ `false` - to not make the property • `mask` - Set this to `true` for properties whose value must be masked such as password fields. You can add new (supported) properties under same property group or in a new property group. If you are overriding default properties, ensure to increment the version of the XML file. For a list of supported properties, see Structure of the store.xml File. **Tip:** Use only the suggested data types (the `type` attribute) for the existing and new properties. Using data types other than the suggested data types might result into unexpected errors.

XMLS Tags	Purpose
<choices>	Container tag for options of the property value.
<choice>	Identifies the option of the property.

Implementation of Locking by Using a Store

Earlier locking was implemented as part of the cluster provider: FTL, Ignite, or Legacy ActiveSpaces. Now you can implement locking by using a store. Store-based locking provides a single consistent way of locking irrespective of cluster and store provider combination. Store-based locking can be used in unclustered environments.

Locking Scenarios

To determine which locking implementation is used in the respective scenarios, refer to the following table:

Clustered

Cluster Provider	Cache	Store	Locking
Apache Ignite	Apache Ignite	Yes	Apache Ignite or Store
TIBCO FTL	Apache Ignite	Yes	Apache Ignite or Store
TIBCO FTL	NA	Yes	Store

Unclustered

In the unclustered mode, if the **Store** option is selected then the Store-based locking is used.

Note: Apache Ignite locking is the default locking mechanism that is used unless the Store-based locking is explicitly enabled.

Configuration Properties

On the **Cluster** tab of the CDD file, to enable Store-based locking, configure the following properties.

Property Name	Property Value	Description
`be.engine.store.locking.enabled`	True or False	Enables or disables the Store-based locking feature. **Note:** • The Store-based locking is supported only when ActiveSpaces and Cassandra are used as Store. • However, relational databases, such as Oracle, PostgreSQL do not support this feature.
`be.engine.api.spec.enable`	True or False	Enables or disables the MemberHealthStatus API. This API is used to determine whether a member is dead or alive.
`be.engine.http.ping.port`	Port range	Specifies the port number range on which the MemberHealthStatus API would be available. Specify the lower and upper port range, separated by hyphen (-). Example: 8180-8190

Apache Ignite Locking Properties

Due to network issues, there may be a failure in locking because of disconnection. To ensure Store-based locking for Apache Ignite, on the **Cluster** tab of the CDD file, configure the following properties.

Property Name	Description
`be.ignite.lock.disconnect.retry.interval`	Specifies the interval duration to retry locking when disrupted due to disconnection. The default value is 5 seconds.
`be.ignite.lock.disconnect.retry.attempts`	Specifies the number of attempts allowed to retry locking when disrupted due to disconnection. The default value is 3.

Custom Store

Use the Java API to create your own custom store according to your requirement. After you have created and implemented the custom classes, to add the new store to the CDD editor in TIBCO BusinessEvents Studio, perform the steps in Creating a Custom Store.

Key Classes in the Custom store API

Few of the key Java classes, interfaces, and their key components are mentioned in the following sections. For a complete list of Java classes and interfaces for the custom store, see *Java API Reference*.

To refer to the reference implementations of Redis as a persistent store by using the custom store API's, see reference implementations on GitHub.

BaseStoreProvider

The `BaseStoreProvider` class acts as an entry point for the custom store implementation. Specify the fully qualified name of this class into the `class` element of `store.xml` file.

Additionally, if you have implemented the `StoreProperties` class, override the `getStoreProperties` method to return an instance of a custom implementation of the `StoreProperties` class.

StoreProperties

You can use the `StoreProperties` class for the following:

- Set the maximum table name length. The default value is `-1` which signifies no limit.
- Set the maximum column name length. The default value is `-1` which signifies no limit.
- Implement store-specific sanitization.

StoreDataTypeMapper

Use the `StoreDataTypeMapper` method to map store-specific data types to corresponding TIBCO BusinessEvents data types.

StoreFilterBuilder

The `StoreFilterBuilder` class converts BQL language queries to store-specific queries. You can override this class to provide store-specific behavior. Implement this class only if you want to use TIBCO BusinessEvents query support (Query Agent) in your project.

The operators and keyword `getter` methods can be overridden to specify store-specific keywords values. If a keyword is not supported, override that specific `getter` to return `NULL`. Override the following methods if any store-specific operator needs to be used. Return `NULL` if any operator is not supported in the store.

Operator	Method	Default Value
Equal	getOperatorEqual	=
Not Equal	getOperatorNotEqual	!=
Like	getOperatorLike	like
Greater	getOperatorGreater	>
GreaterOrEqual	getOperatorGreaterOrEqual	>=

Operator	Method	Default Value
Less Than	getOperatorLessThan	<
LessThanOrEqual	getOperatorLessThanOrEqual	<=
IN	getOperatorIN	in
OR	getOperatorOR	or
NOT	getOperatorNOT	not
WHERE	getOperatorWHERE	where

StoreRowHolder

The `StoreRowHolder` class contains write and read data information with filters. Following are some important attributes and methods:

tableName	Specifies table name
selectList	Applicable in case of data reads. Stores the column names required as output of data reads. A null value specifies that all columns are required. The null value works like a `*` operator in a select query.
colDataMap	Stores the map of column names and column data. In case of write operations, this attribute contains column data to be written. In case of read operations, this attribute contains query parameters.
filtersDataMap	Stores the filter information for filters like `where` and `group by`. The `where` filter should be used only when `StoreRowHolder#colDataMap` is empty or null.
ttl	Stores the concept time to live value.

StoreColumnData

The `StoreCloumnData`class contains column data information.

Schema Generation for Custom Stores

Use the `StoreDeployment` class in the schema generation API and provide implementation of abstract methods. If you have implemented the `StoreProperties` class, override the `getStoreProperties` method to return an instance of a custom implementation of `StoreProperties` class.

Set the property `java.property.jdbcdeploy.bootstrap.keyword` file in the `be-storedeploy.tra` file at `*BE_HOME*/bin` for schema generation .

Creating a Custom Store

You can use the custom store API to create a custom store according to the requirement of your project.

For more information about the classes, see *Java API Reference*.

To refer to the reference implementations of Redis as a persistent store by using the custom store API's, see reference implementations on GitHub.

Before you begin

Set the property `java.property.jdbcdeploy.bootstrap.keyword.file` in the `be-engine.tra` file at `*BE_HOME*/bin`

Procedure

1. Create an XML file and save it with the name as `store.xml`.

2. Configure the `store.xml` file for the custom store and add the properties that you want to display in TIBCO BusinessEvents Studio as follows:

    a. Provide the connection-related properties in the `Connection` property group.

    b. Inside the `Security` tag, provide the connection-related properties in the `Connection-Security` property group.

    c. Inside the `class` tag, provide the class name implementation of the `BaseStoreProvider` class.

    d. Inside the `schema-generation` tag, provide class name of implementation of the `StoreDeployment` class.

    For more information, see Structure of the store.xml File.

3. Create all the required Java class files by using the Java API for custom store.

4. Archive all the Java class files for the custom store along with the `store.xml` file as a JAR file. Place the `store.xml` file at the base of the JAR file.

    Ensure that the JAR is an "Uber JAR" which includes all the dependencies. For more details, see the reference implementations on GitHub.

5. Save this JAR file to *BE_HOME*`/lib/ext/tpcl/contrib`.

6. Restart TIBCO BusinessEvents Studio.

7. (Optional) If your custom store implementation has any external dependencies (that are not already present in the Uber JAR that you have created for the custom store), add their respective JAR files to the *BE_HOME*`/lib/ext/tpcl` folder or add their classpaths to the `be-engine.tra` file at *BE_HOME*`/bin`.

    Note: Ensure that you do not add the dependencies that are already present in the *BE_HOME*`/lib` folder.

Result

New custom store is available for editing in the CDD editor in TIBCO BusinessEvents Studio.

Domain Objects Configuration

At system startup, domain object settings determine how entity objects are stored and pre-loaded from the backing store to cache.

Warning: If a backing store is already set up, and you enable any Use Backing Store settings, you must update the backing store setup. The backing store will not operate correctly unless you do so. See Updating Existing Backing Store Schema .

Domain object settings let you configure various behaviors for objects generated by the inference engines and stored in a cache. Many options relate to the way objects move between cache and backing store, so that you can tune memory usage and performance as needed.

You can configure the various behaviors globally (at the default level) and you can set overrides at the object type level. (Not all object level settings, however, are overrides.)

The main options are as follows. (Other options pertain to more specific situations and all are documented in the reference tables.)

- The mode: Cache Only, Memory Only, Store Only. See Cache Modes and Project Design in *TIBCO BusinessEvents Architect's Guide* to understand the effect of the different modes.
- Whether the objects or handles or both are preloaded into the backing store at startup.
- A preload fetch size (one setting for both objects and handles).
- Whether the cache is limited or unlimited. If limited, you can specify the size of the cache. See Configuring a Limited (or Unlimited) Cache.

At the individual object type level only, you can also configure the following:

- Whether the object is stored in the backing store or not.
- A backing store table name (used when setting up a backing store). See *TIBCO BusinessEvents Administration*.

The settings are applied at the object level. For example, a contained concept can have a different limited cache setting from its container concept, and could be evicted from the cache at a different time.

Configuring Preloading Options

Preloading refers to the loading of the cache with objects from the backing store, at system startup.

If preload is not enabled, objects are loaded as needed from store. This can have an impact on the performance when the object is loaded for the first time from store into cache.

At the default level, you can choose to preload or not preload two types of items separately: the objects themselves, and the handles to the objects, which are stored in a separate table called the *object table*. You can also specify the number of objects to preload (one setting for both types of items).

At the individual object level, you can override the preload setting as follows:

- Use the default setting
- Preload (True)
- Do not preload (False)

You can also specify (or override) how many objects to preload.

Note: A tuning property for preloading is available: `be.engine.cluster.recovery.threads`

Before you begin

Set up the cluster with **Object Management Mode** as `Cache`.

Procedure

1. In the TIBCO BusinessEvents Studio CDD editor, select the **Cluster** tab.
2. Select **Object Management > Domain Objects > Default** node and configure settings as explained in Domain Objects Default Settings Reference.

3. To add object-level overrides and other object-level settings do the following:
    a. Select **Object Management > Domain Objects > Default** node and click **Add**.
    b. In the Entity Selection dialog, select the ontology object type or types you want to configure and click **OK**.

        The first entity in the list is selected and the configuration section for the entity displays on the right.
    c. Select the **/uri** node for each selected entity in turn and configure the settings on the right as needed. You can also edit existing override entries, and remove entries not needed (by clicking Remove). See Domain Object Override Settings Reference.
4. Save the resource.

Domain Objects Default Settings Reference

The Domain Object Default settings apply to all objects except those for which you explicitly configure overrides, using the Domain Object Overrides section. The options are available based on the object management type selected.

For Cache Object Management

Domain Object Default Settings for Cache OM

Property	Description
Mode	With Cache OM, you can keep memory objects in the cache or Rete network using the following cache modes. **Memory Only** Objects are not persisted in the cache. They are kept in the Rete network (working memory) only. **Cache Only** Objects are persisted in the cache. They must be loaded into working memory as needed. This is the most common choice for a cache cluster. See Cache Modes and Project Design in the *TIBCO BusinessEvents Architect's*

Property	Description
	Guide to understand the effects of this setting. Default is `Cache Only`. **Note:** If you set the mode to Memory Only, the rest of the properties in this section are not relevant and are ignored.
Preload Entities	Specifies whether objects are loaded into the cache from the backing store on system startup (both normal startup and recovery). **Checked** All objects are preloaded into the cache from the backing store. Lower level settings can override this setting by excluding specified objects. **Unchecked** No objects are preloaded. into the cache from the backing store. Lower level settings can override this setting by including specified objects. Default is `unchecked`.
Preload Handles	Specifies whether object handles are loaded into the `ObjectTable` cache. The `ObjectTable` cache holds references (handles) to the objects themselves. **Checked** All object handles are preloaded. Lower-level settings can override this setting by excluding handles for specified objects. **Unchecked** No object handles are preloaded into the cache from the backing store. Lower-level settings can override this setting by including handles for specified objects. Default is `unchecked`.
Preload Fetch Size	If **Preload Entities** or **Preload Handles** or both are checked, this setting specifies the number of entity objects or handles (or both) to preload for each entity type whose objects or handles (or both) are configured to be preloaded.

Property	Description
	This setting applies to both objects and handles and cannot be set differently for each. Objects and handles are fetched in a non-deterministic manner. This setting can be overridden at the entity level. Set to `0` to preload all. Set a number to load that number of objects or handles (or both). Default is `0`. **Note:** This setting is ignored unless **Preload Entities** or **Preload Handles** or both are checked.
Check for Version	This field applies to concepts that use cache-only mode. An inference agent uses its L1 cache, a local cache of limited size, to improve access time to the concepts stored in the cluster cache. When an agent finds a concept instance in this local cache, the Check for Version setting determines whether the agent uses the instance directly, or instead check in the cluster cache for a more recent version. **If Checked** (default value) The agent checks in the cluster cache for a more recent version. If a more recent version exists, it is used, and replaces the one found in the local cache. **If Not Checked** The agent uses the instance found locally. When content-aware load balancing is used, the local instance can be used without checking for version, improving performance. Default is `checked`.
Evict from Cache on Update	Used only if the `Agent.`*`AgentClassName`*`.cacheTxn.updateCache` property is set to false (see CDD Agent Classes Tab Properties Reference).

Property	Description
	If selected then whenever a rule action changes the value of any of an entity's properties or inserts a new entity, the entity instance is evicted from the cache (updates are saved in the backing store). Use as needed to improve performance and cache memory management. For example, if an entity is not accessed frequently, it may save memory in the cache if the entity is evicted from the cache after it is updated.
Is Cache Limited	If selected, the cache size is limited. Limited cache requires use of a backing store. See Configuring a Limited (or Unlimited) Cache. The size of the entity cache and the size of the object table cache are set in the Object Management section of the Cluster tab. If not checked, the cache size is unlimited. You can override this default setting in entity overrides. Default is `unchecked`.
Concept TTL	Time-to-live (in seconds) for the concept. After the time-to-live is expired, the concept expires within the acceptable limits of the timeout. The default value is -1, which means the concept does not expire, and must be explicitly consumed. If you do not need to write the concept to cache, set this field to 0. The following are some constraints for using the **Concept TTL** field: • **Containment relationship:** Ensure that parent and child concepts expire together or that the time-to-live for the parent is less than the time-to-live for the child concept. Thus, the application never references the child once the parent has expired. • **Reference Relationships:** Do not set this field for concepts that reference other concepts, or for concepts that are referenced in other concepts.

Property	Description
	Note: After the time-to-live expires, the concept is removed from the object table after a specified delay interval. This delay helps reduce an object table load. You can configure this interval using the `objectTable.eviction.delay.interval` CDD property. Thus, *Actual eviction time from object table = TTL +* `objectTable.eviction.delay.interval` The default value is 60 seconds.
Entity Cache Size (in MB)	Specifies the size of the limited cache, in total memory capacity for each object type. The setting is per processing unit. See Configuring a Limited (or Unlimited) Cache. The default value is `50MB` and the minimum value should be greater than `15MB`. For limited cache, the entity cache size for non-overridden entities is set to the maximum value between the specified entity cache size and 50% of the available memory.
Object Table Cache Size (in MB)	Specifies the maximum size of the object table cache, in total memory capacity. Used with limited cache only. See "The Role of the Object Table" in the *TIBCO BusinessEvents Architect’s Guide* for more details about the object table. Also see,Configuring a Limited (or Unlimited) Cache. The default value is `200MB`. The Object Table Cache Size property is available only when the legacy lookup strategy is enabled.

For Store Object Management

Domain Object Default Settings for Store OM

Property	Description
Mode	With Store OM, you can keep memory objects in the store or Rete network using the following modes. **Memory Only**

Property	Description
	Objects are not persisted in the store. They are kept in the Rete network (working memory) only. **Store Only** Objects are persisted in the store. They must be loaded into working memory as needed. This is the most common choice for a Store OM type cluster. See Cache Modes and Project Design in the *TIBCO BusinessEvents Architect's Guide* to understand the effects of this setting. Default is `Store Only`. **Note:** If you set the mode to Memory Only, the rest of the properties in this section are not relevant and are ignored.
Concept TTL	Time-to-live (in seconds) for the concept. After the time-to-live is expired, the concept expires within the acceptable limits of the timeout. The default value is -1, which means the concept does not expire, and must be explicitly consumed. If you do not need to write the concept to store, set this field to 0. The following are some constraints for using the **Concept TTL** field: • **Containment relationship:** Ensure that parent and child concepts expire together or that the time-to-live for the parent is less than the time-to-live for the child concept. Thus, the application never references the child once the parent has expired. • **Reference Relationships:** Do not set this field for concepts that reference other concepts, or for concepts that are referenced in other concepts.

Property	Description
	Note: After the time-to-live expires, the concept is removed from the object table after a specified delay interval. This delay helps reduce an object table load. You can configure this interval using the `objectTable.eviction.delay.interval` CDD property. Thus, *Actual eviction time from object table = TTL +* `objectTable.eviction.delay.interval` The default value is 60 seconds.

Domain Object Override Settings Reference

You can override some of the default domain object settings for each ontology object.

For details on default settings, see Domain Objects Default Settings Reference.

CDD Cluster Tab Domain Object Override Settings

Property	Notes
Entity URI	Specifies the project path to the entity for which overrides are being set. Defaults to the selected entity's URI. For example: `/Concepts/MyConcept`.
Mode	Overrides the Default level setting for this object type. **Memory Only Mode** If you set the mode for an entity to Memory Only, the rest of the properties in this section are not relevant and are ignored. Backing store is disabled for entities that use Memory Only mode. **Warning: Caution:** If you change from Memory Only mode to a cache mode after the backing store has been set up, you must update the backing store schema. See Updating Existing Backing Store Schema.

Property	Notes
Preload Entities	Specifies whether objects of the specified type are loaded into the cache from the backing store on system startup (both normal startup and recovery). Overrides the Preload Entities setting at the Default level. **default** Use the Preload Entities setting specified at the default level. **true** Objects of the specified type are preloaded into the cache from the backing store. If the default level setting is not to Preload Entities, you can use this override to preload selected entities. **false** No objects of the specified type are preloaded into the cache from the backing store. If the default level setting is to Preload Entities, you can use this override not to preload selected entities. Default is `default`.
Preload Handles	Specifies whether object handles for the specified type are loaded into the cache from the backing store on system startup (both normal startup and recovery). Overrides the Preload Handles setting at the Default level. **default** Use the Preload Handles setting specified at the default level. **true** Handles for the specified type are preloaded into the cache from the backing store. If the default level setting is not to Preload Handles, you can use this override to preload

Property	Notes
	selected entities' handles. **false** No handles for the specified type are preloaded into the cache from the backing store. If the default level setting is to Preload Handles, you can use this override to prevent preloading the selected entities' handles. Default is `default`.
Preload Fetch Size	Overrides the Preload Fetch Size setting in the Default settings.
Check for Version	Overrides the value of the same-named setting in the Default settings.
Evict from Cache on Update	Overrides the value of the same-named setting in the Default settings.
Is Cache Limited	Overrides the value of the same-named setting in the Default settings.
Entity Cache Size (in MB)	Specifies the size of the limited cache, in total memory capacity for each object type. The setting is per processing unit. See Configuring a Limited (or Unlimited) Cache. The default value is `50MB` and the minimum value should be greater than `15MB`. The overridden Entity Cache Size takes precedence when the value is set to greater than zero for all entity types. The default Entity Cache Size is used when the value is less than zero or not mentioned.
Concept TTL	*(Concepts only)* Overrides the value of the same-named setting in the Default settings.
Backing Store Section	

Property	Notes
Has Backing Store	Used only if the `Store` is selected as the **Persistence Option**. To exclude an entity from the backing store, clear the **Has Backing Store** checkbox. **Warning:** If you enable this override setting after the backing store has been set up, you must update the backing store schema. See Updating Existing Backing Store Schema. **Concepts Related by Containment or Inheritance** All concepts related by containment or inheritance must either be included in the backing store or excluded from the backing store. That is, they must share value for the Has Backing Store setting. By default, the checkbox is selected.
Table Name	Specifies a table name to be used in the backing store. Typically used if the entity name is long. For details on ontology identifiers that exceed the DBMS maximum column length, see *TIBCO BusinessEvents Developer Guide*.
Properties Metadata Section	
Property	*(Read-only)* Displays the property name.
Present in Key	**Restriction:** Available only for concepts and when a key-based data lookup strategy is enabled. For details, see *TIBCO BusinessEvents Administration Guide*. Select the checkbox if the property is used as the primary key for data lookup in the store.
Present in Index	Select the checkbox if the property is used in the index.
Encrypted	Specifies whether the property should be encrypted. The values are:

Property	Notes
	• true • false (default) **Note:** • For field level encryption to work, cluster level "Security" must be enabled and a policy file must have 'data_encryption=true' set in it. • Fields that are indexed cannot be selected for encryption. • Fields that are used in query filters should not be encrypted.
Max Length	Used with a backing store to specify the length of string properties that exceed 255 characters (that is the actual contents stored in the column is more than 255 characters). Specifies the expected maximum length for the property, see *TIBCO BusinessEvents Developer Guide*
Reverse References	This setting is for use only with ConceptReference type concept properties. With a backing store, database updates related to a referring concept in a referenced concept can cause decreased performance. This happens when there are very many reverse references in a shared instance (referenced by many other instances). To address this issue, set the value to `false` for ConceptReference type properties. If you set the value to `false`, you must explicitly remove ConceptReference properties for deleted referenced concepts in the referring concept in your code. For example, if employee is a ConceptReference type property in a concept acme, and smith is an instance of a concept type employee, then you would set Reverse References to true for the employee ConceptReference property, and you would add

<table>
<tr><th>Property</th><th>Notes</th></tr>
<tr><td></td><td>something like this to the rules:<pre>acme.employee = null;
Instance.deleteInstance(smith);</pre>Or, for array properties:<pre>Instance.PropertyArray.removeConceptReference
(acme.employee, smith);
Instance.deleteInstance(smith);</pre>Default is <code>true</code>.</td></tr>
<tr><td>Affinity Key</td><td>Sets the property as the key for easily accessing objects in Apache Ignite cache to improve the performance. After the affinity key is applied, objects with the same value in the affinity key property are stored in the same node in the cache.

The affinity key is applied when the legacy lookup strategy is disabled.

<b>Note:</b><ul><li>You can select only one property as the affinity key per entity and it is applied to fields present in the key.</li><li>In a parent-child entity-relationship, the affinity key from a parent entity (if defined for a parent entity) is inherited to its child entity when a child entity does not have any affinity key defined.</li><li>The affinity key must be set on the same property for a concept and its contained or reference concepts irrespective of their relationship to persist them on the same node.</li></ul></td></tr>
</table>

Composite Indexes Section

The table lists the composite indexes for the entity. Composite indexes are based on multiple columns (entity properties). You can add, delete, and edit a composite index from the list.

Property	Notes
Click the **Add** icon () to create a composite index. In the Create Composite Index window, select the properties that you want to include in the composite index and click **OK**. A new composite index (with an autogenerated name) is displayed on the composite indexes list.	
Index Name	Displays the name of the composite index. You can perform the following actions on the selected composite index: • **Edit the composite index** - Select the index row and click the **Edit** () icon to edit the composite index. You can either add a property or remove a property from the index. • **Delete the composite index** - Select the index row and click the **Delete** () icon to delete the composite index.
Properties	Displays the properties that are included in the composite index.

Application Metrics Configurations

You can add the following metric configurations for your project:

- **Control Plane**. For more information, see Configuring Control Plane as a Metrics Store.
- **InfluxDB**. For more information, see Configuring InfluxDB as a Metrics Store.

Before you configure a metrics store, set the following **Application Metrics** on the **Cluster** tab in the project CDD file.

Configuration	Description
Queue Size	Specify the size of the queue to limit the amount of data you want to accept while publishing to the metrics store provider.

Configuration	Description
	The **Queue Size** and **Thread Count** fields are used for scaling purposes depending on the data size and load.
Thread Count	Specify the number of threads for starting publishing the data to the configured metrics store provider. You need multiple threads to publish data to metrics store concurrently.
Max Retries	Specifies the maximum number of retry attempts for connecting to the configured metrics store provider.
Retry Wait Time	Specifies the time interval before retrying to connect to the configured metrics store provider.
Monitor System Metrics	Enables the monitoring of system metrics. The following system metrics can be monitored. By default, only CPU usage and Heap Memory are published. To publish other metrics, use the properties mentioned in the Properties section below: • CPU Usage • Memory Utilization • Number of cluster-wide locks • Event throughput • Rule execution time • RTC Transaction Manager Report • Individual Agent Performance Attributes • Cache Performance Attributes • Overall Inference Agent Statistics Individual Agent Performance Attributes and Cache Performance Attributes are published only for the Entity for which you want to publish metrics. By default, system metrics monitoring is disabled. Monitoring rule execution time additionally requires setting two properties in the project CDD file. The properties are as follows:

Configuration	Description

	`com.tibco.be.metric.publish.enable=true`
	
	`be.stats.enabled=true`

System Metrics Publish Time (ms)	Specify the time interval in milliseconds to publish the system metrics to the configured metric store. This property is enabled on selecting the Monitor System Metrics checkbox.
Properties	Add properties and specify the values as required. To publish metrics, use the following properties:

Metric	Property
CPU Usage	be.system.metrics.cpuUsage Default: true
Memory Utilization	be.system.metrics.heapMemoryUsed Default: true
Number of cluster-wide locks	be.system.metrics.clusterWideLocks Default: false
Event throughput	be.system.metrics.eventThroughput Default: false
Rule execution time	be.system.metrics.ruleExecutionTime Default: false
RTC Transaction Manager Report	be.system.rtc.managment.report

Configuration	Description

Metric	Property
	Default: false
Individual Agent Performance Attributes	be.system.agent.stats Default: false
Cache Performance Attributes	be.system.cache.stats Default: false
Overall Inference Agent Statistics	be.system.inference.stats Default: false

Integrating TIBCO BusinessEvents with InfluxDB and Grafana

Integrating your TIBCO BusinessEvents application with InfluxDB and Grafana is a great open-source solution to your time-series database and real-time visualization requirements.

You can configure InfluxDB as the metrics store for your application and connect it to a Grafana server for creating visualizations and monitoring.

Before you begin

- You must have an existing Grafana and InfluxDB setup in working condition.
- Make a copy of the sample dashboard configuration file `FraudDetectionVisualizations.json` at `BE_HOME/examples/metrics/FraudDetectionInfluxDB`, save it to the project, and edit it as per your requirement.

Procedure

1. Configure the CDD file for the TIBCO BusinessEvents application that you want to integrate with InfluxDB. For more information, see Configuring InfluxDB as a Metrics Store.
2. Start the InfluxDB and Grafana servers for your respective operating system. For more information, see the InfluxDB documentation and Grafana documentation.
3. Setup InfluxDB Server to create organization and bucket for your application.
4. Add the InfluxDB database as the data source on the Grafana server and add values for the following fields according to your TIBCO BusinessEvents application:

Field	Description
Name	Give a name to the data source that you are adding.
Query Language	Use Flux as query language.
Connection	
URL	Add the URL of the InfluxDB server associated with your project.
Organization	Add the name of the InfluxDB organization associated with your project.
Token	Add a token to be used to access data from InfluxDB server.
Default Bucket	Add the name of the bucket associated with your project.

What to do next

Create a Grafana dashboard by importing the sample dashboard and start the TIBCO BusinessEvents engine for your application.

Configuring InfluxDB as a Metrics Store

To integrate your TIBCO BusinessEvents application with InfluxDB, you can configure InfluxDB as the metrics store. You can use the steps below to configure the InfluxDB metrics store through the CDD file:

To know more about InfluxDB and Grafana integration, see Integrating TIBCO BusinessEvents with InfluxDB and Grafana.

Procedure

1. In TIBCO BusinessEvents Studio, open the project CDD file for editing.
2. On the **Cluster** tab, under **Application Metrics**, select **Metrics Store**.
3. Under **Metrics Store**, from the **Store Provider** dropdown list, select **InfluxDB**.

4. In the Configuration window, provide the values for the following configurations:

Field	Description
InfluxDB URL	Specify URL of the InfluxDB server.
Organization	Specify organization of the InfluxDB database.
Bucket	Specify bucket to use from the organization.
Auth Scheme	Select authentication scheme to use. • `TOKEN`: Token is used to authenticate. • `SESSION`: A user session is created with the provided username and password. Use this to authenticate. Specify a username and password with write permission to the organization. The default value is `TOKEN`.
Token	Used when **Auth Scheme** is `TOKEN`. Specify a token with write permission to the organization.
Username	Used when **Auth Scheme** is `SESSION`. Specify a username with write permission to the organization.
Password	Used when **Auth Scheme** is `SESSION`. Specify the password for the **Username** specified.
Connect Timeout	Specify socket timeout in milliseconds. **Default value**: 10000
Write Timeout	Specify write timeout in milliseconds **Default value**: 10000
Write Precision	Specify the write precision for the batch.

Field	Description
	Default value: NANOSECONDS
Log Level	Specify log level for the InfluxDB client library.

5. (Optional) To enable secure authentication, select the **Security Enabled** checkbox and fill values for the following fields:

Field	Description
Trusted Certificates Location	While using SSL authentication, provide the path to the base folder containing the certificate files.
Trust Store Password	Specify a password for the trust store.

Note: To make a TIBCO BusinessEvents application available even when InfluxDB is not, set the `com.tibco.be.metric.influxdb.readycheck` property to `false` in the application CDD file.

Configuring Control Plane as a Metrics Store

To integrate your TIBCO BusinessEvents application with Control Plane, you can configure Control Plane as the metrics store. The metrics configured for the Control Plane is pushed through Hawk. You can use the steps below to configure the Control Plane metrics store through the CDD file:

Procedure

1. In TIBCO BusinessEvents Studio, open the project CDD file for editing.
2. On the **Cluster** tab, under **Application Metrics**, select **Metrics Store**.
3. Under **Metrics Store**, from the **Store Provider** dropdown list, select **Control Plane**.

What to do next

Register TIBCO BusinessEvents in a data plane and monitor it by using TIBCO® Control Plane. For more information, see TIBCO® Platform Documentation.

Adding Entity Configurations for Metric Stores

You can use TIBCO BusinessEvents Studio to specify entities that you want to publish to the TIBCO Web dashboard. You can specify a filter to select only those instances which qualify the filter.

Procedure

1. In the BusinessEvents Studio, open the project CDD file for editing and open the **Cluster** tab.
2. From the **Application Metrics** list, select **Entity Configurations**, and click **Add** to add an entity configuration.

3. Specify the value for the following fields for the new entity and save the agent.

Field	Description
Entity Uri	Browse and select the supported entity that you want to send to the publisher based on your filter. The supported entities are concepts, events, and scorecards.
Entity Filter	Specify a query to filter out the entity based on your requirement. For example, the following values specifies to send the Account concept to the publisher only when Balance is greater than 10000. • **Entity Uri** - /Concepts/Account • **Entity Filter** - Balance > 10000 If this field is empty, then all instances of the entity are pushed to the configured metrics store provider.
Publish Application Metrics	Select the checkbox to publish Application metrics for the entity. **Note:** In the previous release, the application metrics were published for all the entities. By using this checkbox, you can configure whether the application metrics of an entity need to be published or not.
Publish Agent Stats	Select the checkbox to publish Agent metrics for the entity.
Publish Cache Stats	Select the checkbox to publish Cache metrics for the entity.
Tag Entity Id	(Only applicable for InfluxDB as a metric store) When enabled entity id is tagged with each measurement. **Default value**: *selected*

Custom Application Metrics Store

Use the Java API to create your own custom metrics store according to your requirement. When implementing a custom metrics store, you can extend custom metrics classes and override methods that TIBCO BusinessEvents invokes at startup and during run time. After you have created and implemented the custom classes, follow the steps in Creating a Custom Application Metrics Store to add the new application store to the CDD editor in TIBCO BusinessEvents Studio.

Key Classes in the Custom Application Metrics Store API

Few of the key Java classes, interfaces and their key components are mentioned in the following sections. For a complete list of Java classes and interfaces for the custom metrics store, see *Java API Reference*.

To refer to the reference implementations of Elasticsearch as a metric store by using the custom metrics store API's, see reference implementations on GitHub.

MetricsStoreProvider

The `MetricsStoreProvider` interface acts as an entry point for the custom metrics store implementation. The custom metrics store needs to implement this interface. Provide the implementation of all the methods in the interface. Specify the fully qualified name for the implementing class parameters of `metrics-store.xml`.

MetricsRecordBuilder

The class implementing `MetricsStoreProvider` uses `MetricsRecordBuilder` to create and build `MetricRecord` instances for each entity in a RTC change list that is configured to publish to a target metric store.

MetricRecord

`MetricRecord` holds a store-specific metric type and an operation type. The store-specific implementation of `MetricsStoreProvider` builds an instance of `MetricRecord` using `MetricRecordBuilder` and publishes it to the target store.

Creating a Custom Application Metrics Store

You can use the custom store API to create a custom metrics store according to the requirement of your project.

For more information about the classes, see *Java API Reference*.

To refer to the reference implementations of Elasticsearch as a metric store by using the custom metrics store API's, see reference implementations on GitHub.

Procedure

1. Create an XML file and save it with the following name: `metrics-store.xml`
2. Edit `metrics-store.xml` and configure it for your custom metrics store. Add the properties that you want to display in TIBCO BusinessEvents Studio. For more information, see Structure of the metrics-store.xml File.
3. Create all the required Java class files by using the Java API for custom metrics store.
4. Archive all the Java class files for the custom metrics store along with the `metrics-store.xml` file as a JAR file.
5. Save this JAR file to `*BE_HOME*/lib/ext/tpcl/contrib`
6. Restart TIBCO BusinessEvents Studio.

Result

New custom metrics store is available for editing in the CDD editor in TIBCO BusinessEvents Studio.

Structure of the metrics-store.xml File

In TIBCO BusinessEvents Studio, based on the object management selection, the matching store provider `metrics-store.xml` file is picked and a form-based UI is generated listing the configuration properties from `metrics-store.xml`

Sample metrics-store.xml File

The following is a sample of `metrics-store.xml` to add InfluxDB as a metrics store.

```
<?xml version="1.0" encoding="UTF-8"?>
<metrics-store>
    <type>InfluxDB</type>
    <label>InfluxDB</label>

<class>com.tibco.cep.metrics.influxdb.InfluxDBMetricsStoreProvider</clas
s>
    <description>InfluxDB is a open source time series
database.</description>
    <version>1.0.0</version>
    <!--The input required by the metrics store provider e.g connection
details, ssl config etc.-->
    <properties>
        <property name="influx.url" displayName="InfluxDB URL"
type="String" default="http://localhost:8086" mandatory="true"/>
        <property name="influx.org" displayName="Organization"
type="String" default="" mandatory="true"/>
        <property name="influx.bucket" displayName="Bucket"
type="String" default="" mandatory="true"/>
        <property name="influx.authScheme" displayName="Authentication
Scheme" type="String" default="TOKEN">
          <choices>
              <choice displayed="TOKEN" value="TOKEN" />
              <choice displayed="SESSION" value="SESSION" />
        </choices>
        </property>
        <property name="influx.token" displayName="Token" type="String"
default="" mask="true"/>
        <property name="influx.username" displayName="User Name"
type="String" default="" mask="false"/>
        <property name="influx.password" displayName="Password"
type="String" default="" mask="true"/>
       <property name="influx.connectTimeout" displayName="Connect
Timeout" type="Integer" default="10000"/>
        <property name="influx.writeTimeout" displayName="Write Timeout"
type="Integer" default="10000"/> <    <security>
                   <property name="influx.trustStoreLocation"
displayName="Trusted Certificates
Location" type="File" directory="true" default="" mask=""
mandatory="true"/>
```

```
                       <property name="influx.trustStorePwd"
displayName="Trust Store Password"
type="String" default="" mask="true" mandatory="false"/>
               </security>
    </properties>
    <!--The input required for the entity-set, these properties will
apply to all entities-->
    <entity-set>
           <!--The input required for each entity-->
           <entity>
               <property name="influx.tagEntityId" displayName="Tag Entity
Id" type="Boolean" default="true"/>
           </entity>
    </entity-set>
</metrics-store>
```

XML Tags of the `metric-store.xml` File

The following table lists the major tags available in the `metrics-store.xml` file and sample values for InfluxDB:

The store.xml File Tags

XML Tags	Description
<metrics-store>	Identifies that the enclosed XML content is for a metrics-store.
<type>	Sets the type of the metrics store and is used to register and load metrics store.
<label>	Displays the label for the metrics store type
<class>	Fully qualified class implementing the `MetricsStoreProvider` interface.
<description>	A short description of the metrics store.
<version>	Version number of the XML file. During the startup, TIBCO BusinessEvents Studio compares the version number of the default (version 1.0) and the custom `metrics-store.xml` file and displays settings accordingly.

XML Tags	Description
<properties>	Container tag for property tags.
<property>	Define the properties that you want to display on the UI. For each property you can define the appropriate attributes such as displayName, type, and default. To display a list of options for the user, you can insert `<choices>` and `<choice>` tags inside the `<property>` tag. **Tip:** Use only the suggested data types (the type attribute) for the existing and new properties. Using data types other than the suggested data types might result in unexpected errors.
<security>	Container group for security-specific properties. This displays a **Security Enabled** checkbox on the UI. The properties under this container are displayed on the UI only when the **Security Enabled** checkbox is selected.
<entity-set>	Container tag for options of the `entity` tag.
<entity>	The `entity` tag can further have multiple `property` tags to define entity specific properties.
choices	Container tag for options of the property values.
choice	Identifies the option of the property.

Telemetry Data Collection

TIBCO BusinessEvents integration with OpenTelemetry allows you to monitor the health and performance of your application across all the services. OpenTelemetry provides support for tracking the progression of a request across multiple services in the application. This tracking is termed *tracing*. Tracing helps you to identify any bottlenecks in your applications and monitor each request across the services.

The telemetry data collected from your TIBCO BusinessEvents application can be displayed on data visualization software. Due to the vendor-agnostic design of the OpenTelemetry specification, you can use one or more open-source or commercial visualization software for the telemetry data.

For complete details about OpenTelemetry, see the OpenTelemetry documentation.

For more details about tracing in TIBCO BusinessEvents, see "TIBCO BusinessEvents Application Tracing" in *TIBCO BusinessEvents Configuration Guide*.

Key Terms

The following terms are the key terms of OpenTelemetry implementation. A brief introduction to these terms is given here. For complete details about OpenTelemetry terminology, see the OpenTelemetry documentation.

Observability

Ability to understand and measure the state of a system by collecting data such as traces, metrics, and logs.

Telemetry

Exporting monitoring data from an application to external analysis software.

Trace

Traces track the progression of a single request, as it is handled by services that make up an application.

Span

Span is a unit of work in a trace. A span has a start time, end time, attributes (key-value pairs), and events.

Context

A span has a span context, which is a set of globally unique identifiers that identifies a request. A context must be carried across threads and services to be able to trace a transaction or request uniquely across process boundaries.

Propagator

In order to extend trace beyond a single process, a context propagation mechanism is required and a propagator must be registered with the OpenTelemetry API. A context when shared with a remote application, it is serialized or deserialized to the vendor-specific protocol using propagators. An application should use one or more propagators that are used by other applications in the ecosystem to be able to parse context.

TIBCO BusinessEvents Application Tracing

In the TIBCO BusinessEvents application, the tracing feature for artifacts is provided out-of-the-box and can be enabled and configured in CDD through TIBCO BusinessEvents Studio.

Using OpenTelemetry, TIBCO BusinessEvents trace all units of work starting from the incoming event processing to post RTC by creating a span and using the span context to correlate them. All spans for a processed event in a TIBCO BusinessEvents application have the same `trace-id` and can be viewed as a unique transaction using analyzing software. The following analyzing software support is provided out-of-the-box with TIBCO BusinessEvents:

- OpenTelemetry Collector
- Jaeger
- Zipkin

To use or add analyzing software other than the out-of-the-box supported software, see TIBCO BusinessEvents Contributions repository on GitHub.

To activate the instrumentation of your TIBCO BusinessEvents application, see Enabling the Application Tracing.

Components Tracing

The following table lists the components for which span is created by default if the telemetry data collection is configured in the TIBCO BusinessEvents application. Telemetry is initialized and your application code is instrumented only when a **Span Exporter** (value other than `None`) is configured for the application. For details about the Telemetry configurations in your TIBCO BusinessEvents application, see Telemetry Configurations Settings.

Note: If required, you can configure your application to ignore some of the entities for tracing. Use the telemetry configurations in the CDD to ignore entities using the specific URI or partial URI.

TIBCO BusinessEvents Components Telemetry Data

Component	Tracing operation
Destination	• *Deserializer*- A message received on a channel destination is deserialized to an event. While deserializing, if the message is carrying a span context from a remote application, it is used to initialize a new span using propagators. • *Serializer* - When an event is sent, it is serialized to transport a specific message. The span context is added to the message to pass on the `trace-id` and other information using propagators.
Rule function	A rule function invocation including a preprocessor execution creates a new span using the current context.
Rule	A rule action creates a new span using the current context.
State machine	As the state transition in state machines happens on events, each state transition is different execution and they could spread over one or many JVM processes (TIBCO BusinessEvents engine). The OpenTelemetry span is limited to a single process hence state machine state scope cannot be traced using a single `trace_id`. Instead, the state machine `instance_id` is added as an attribute to each span and all the state transitions related to an instance can be searched using that `instance_id`.
Post-RTC	Applicable only for cache OM (with or without store) or store OM. • *Cache provider* - A span is created for each cache operation. • *Store provider* - A span is created for each `save` transaction on the configured store.
Virtual rule function	A span is created for each `invoke` call of VRF catalog functions.
DataGrid catalog functions	A span is created for every call to `Cluster.DataGrid` catalog functions.
HTTP Catalog Functions	A span is created for each `send` call to HTTP client catalog functions.

Span Attributes

The following attributes are added to every span created in the TIBCO BusinessEvents application.

- Cluster name
- Engine name
- Processing unit name
- Thread name
- Thread ID
- Class name and method of catalog function or call origin
- Artifact URI for the following artifacts:
    - Destination
    - Rule
    - Rule function
    - Virtual rule function
    - State machine

- Attributes based on span context by the TIBCO BusinessEvents engine:
  - Destination Serializer/Deserializer - Default event URI and the serializer name.
  - Rule - ExtId of business objects in scope. Other properties are not added as span attribute because it might impact the performance; however, if needed you can enable them by adding the following properties in CDD:
    - `otel.trace.payload` - when set to `true`, event payload is added as the span attribute for all events. The default value is `false`.
    - `otel.trace.properties` - when set to `true`, entity properties are added as the span attribute for all entities. The default value is `false`.
    - `otel.trace.payload.`*Events/Application* - when set to `true`, the event payload is added as the span attribute for this specific event. The default value is `false`.
    - `otel.trace.properties.`*Concepts/Account* - when set to `true`, entity properties are added as the span attribute for this specific entity. The default value is `false`.

Sampling

You can use the sampling strategy to control the noise and overhead by reducing the number of samples of traces collected and sent to the back end. TIBCO BusinessEvents provides you the option to select the sampling strategy as per your requirement. These sampling strategies are designed as per the OpenTelemetry specification:

- Send all the spans
- Don't send any span
- Send only a fraction of spans

For more details, see:

- OpenTelemetry documentation
- OpenTelemetry Specification on GitHub

Catalog Functions

Use OpenTelemetry catalog functions to add tracing to specific code in rules, rule functions or custom catalog functions. Each Span created using the catalog function uses the current context only.

The three major groups of OpenTelemetry catalog functions provided with TIBCO BusinessEvents Studio are:

- Scope
- Span
- SpanBuilder

For the complete list of OpenTelemetry catalog functions and their syntax, see the TIBCO BusinessEvents Functions Reference.

Logging

Contextual information such as, `trace_id`, `span_id` and `trace_flags` can be added to application logs. A custom log configuration must be used by adding below pattern to the `PatternLayout` tag under the `Appenders` tag in the `log4j2.xml` file.

```
trace_id[%X{trace_id}] span_id[%X{span_id}] trace_flags[%X{trace_flags}]
```

The application log with `trade_id` and `span_id` can be used to correlate telemetry traces by exporting the log files to the OpenTelemetry collector. You can use a third-party (open-source or commercial) log processor and forwarder agent to export log files to the OpenTelemetry collector.

Using OpenTelemetry Tracing with TIBCO BusinessEvents and TIBCO FTL

To use the same OpenTelemetry transaction when using OpenTelemetry tracing with TIBCO BusinessEvents and TIBCO FTL, add the `_traceparent` field of type String to the TIBCO FTL message format. You can either set this field in the `tibrealm.json` file before starting the TIBCO FTL server or configure it through the TIBCO FTL server GUI.

Enabling the Application Tracing

In TIBCO BusinessEvents, by default the tracing feature is implemented for various artifacts. However, spans are not created for those artifacts until you configure the telemetry settings in the application CDD. Span creation and instrumentation for your application code happens only after you specify a span exporter in the application CDD.

Before you begin

- Set up and configure your analyzing software. For the setup and configuration details of these software, see the respective documentation.
- Run the analyzing software.

Procedure

1. Open your application in the TIBCO BusinessEvents Studio, and open its CDD file.
2. In the CDD file, click the Cluster tab and select Telemetry Configuration.
3. Provide the values for the configuration fields, see Telemetry Configurations Settings.
4. Save the CDD file.
5. Run the TIBCO BusinessEvents engine for your application.

Example

TIBCO BusinessEvents provides OpenTelemetry implementation example for the fraud detection scenario. The example is available at *BE_HOME*`/examples/standard/FraudDetectionTelemetry/FraudDetectionTelemetry`.

You can import the example application in TIBCO BusinessEvents to see the sample CDD configuration. To view the sample tracing for the fraud detection scenario, run the example using the readme.html file at *BE_HOME*`/examples/standard/FraudDetectionTelemetry`.

Telemetry Configurations Settings

To configure the default telemetry data collection and enable the instrumentation of your application, you can modify the telemetry configuration settings in your TIBCO BusinessEvents application CDD.

For more details about the tracing in TIBCO BusinessEvents, see TIBCO BusinessEvents Application Tracing.

The following table lists the telemetry configuration settings available in the CDD editor of TIBCO BusinessEvents Studio.

Telemetry configurations

Property	Description
Service Name	A name to identify the application in the telemetry data store. In the visualization software, you can identify your application using the **Service Name**. The default service name is `default`.
Sampler	(Optional) A sampler strategy helps control the volume of telemetry data that you want to export. • `always_on` - Use this strategy to send every *span* to the export pipeline. • `always_off` - Use this strategy to ensure that no spans are sent to the export pipeline. • `traceidratio` - Use this strategy to send a specified fraction of spans based on their TraceId. You can specify the fraction value in the TraceId Ratio field. • `parentbased_always_on` - When the request is coming from another application or service, use this strategy to use the sampling algorithm of the parent application or service, otherwise to send every span to the export pipeline. • `parentbased_always_off` - When the request is coming from another application or service, use this strategy to use the sampling algorithm of the parent application or service, otherwise not to send any span to the export pipeline. • `parentbased_traceidratio` - When the request is coming from another application or service, use this strategy to use the sampling algorithm of the parent application or service otherwise to send a specified fraction of spans based on their TraceId. You can specify the fraction value in the TraceId Ratio field. The default value is `always_on`.
TraceId Ratio	Enabled only when the `traceidratio` or `parentbased_traceidratio` sampler strategy is selected. A sampling ratio must be specified to a value between `0.0` to `1.0`. The `0.0` value specifies that no spans are to be sent

Property	Description
	while the `1.0` value specifies that all spans are sent. For example, to send 50% of spans to the export pipeline, select the value as `0.5`. The default value is `1.0`.
Propagators	Enter a comma-separated list of propagators to be used to serialize and deserialize the context. Click the **Browse** button to open the list of available propagators. You can select one or more than one propagator for your application. The following propagators are supported exclusively in TIBCO BusinessEvents: • `tracecontext` • `baggage` • `b3` • `b3multi` • `jaeger` • `ottrace` The default propagator is `tracecontext`. For details about propagators, see OpenTelemetry documentation.
Enabled Tracing Resource List	(Optional) By default all resources (/) are enabled for tracing. You can specify which resources need to be traced. All Resources starting with the URI specified will be enabled for tracing. There is AND operation between Enable Resource list and Disabled Resource list. For example: If Enabled Resource List consist of URI: `/Rules/Order/CreateOrderRule` and Disabled Resource List Consist of URI: `/Rules/Order`. In this case, all rules starting with URI `/Rules/Order` are disabled for tracing. The Rule `/Rules/Order/CreateOrderRule` is also not enabled for tracing.
Disable Tracing Resource List	(Optional) Add resources here that you do not want to trace. You can use the full URI of rule, rule function, or destination, or partial paths to ignore all resources starting with that path. The list could also be updated through JMX at runtime.

Property	Description
	Note: If tracing is turned off for a destination, tracing is turned off for all rules and rule functions running for that event. If a rule function is added to this list, the tracing data is not exported for the rule function and also for any catalog function invoked during execution.
Span Exporter	(Optional) Span exporter exports the telemetry data to the specified collector. The following options are provided as span exporter: • `None` • `OTLP` (OpenTelemetry collector) • `Jaeger` • `Zipkin` The default value is `None`.
Endpoint	The endpoint URL of the selected **Span Exporter**. This field is not displayed when the **Span Exporter** value is `None`.
Timeout	The request timeout (in milliseconds) for the selected **Span Exporter**. This field is not displayed when the **Span Exporter** value is `None`.
Headers	Provide comma-separated key-value pairs as headers with the *equals to* symbol = separating the key and value. This field is displayed when the **Span Exporter** value is `OTLP`.
Security Enabled	Select the checkbox to enable the secured connection. This field is displayed when the **Span Exporter** value is `OTLP`.
Security Certificate	Provide the path of the security certificate for the span exporter. This field is displayed when the **Span Exporter** value is `OTLP` and **Security Enabled** checkbox is selected.

Collections, Agent Classes, and Processing Units

You can perform advanced configurations for destinations on the Agent Classes tab. (Destinations that are added to agent classes individually can be configured on the Agent Classes tab.)

Collections

On the Collections tab, you can (optionally) group rules, rule functions, and destinations into collections so that they can be easily assigned to agent classes (and processing units in the case of log configurations).

See Configuring Collections of Rules, Rule Functions, and Destinations.

Agent Classes

Agent classes define the different sorts of agents that you can deploy. Various agent types are available depending on the object management (OM) type and on the add-on products used. Each agent type is configured differently.

Note: See Using Properties at Different Levels to understand the effect of using agent class properties at the cluster level and at the processing unit level to widen the scope of the property.

Configuring Agents with Collections and Individual Resources

Different agent types use different types of resources.

In the Agent Classes node (on the left side of the CDD editor) you see categories of collections. Here, you add collections you defined earlier, as needed to configure the agent class.

Rules

(Inference agent classes only.) It can be convenient to organize rules into collections for use in different inference agent classes. Select rule collections and individual rules as needed to define what rules run on inference agents of such classes at runtime.

Input Destinations

Different agents listen for messages arriving at different destinations. When you select a destination for use in a collection or an individual agent, you add deploy time configuration settings to create a *destination configuration*. For example, you define an event preprocessor and a threading model to use. Each destination configuration is assigned a unique ID.

In the configurations for Input Destinations under **Collections**, you can optionally specify this rule function under the **Thread Affinity Rule Function**.

Note: This is only available for **Shared Queue** and **Destination Queue** threading configurations. When specified, the return value from the rule function is used to pick up the thread to which the message will be dispatched. The rule function is called with the event as the parameter, and is expected to return a `not null` value. Events that return the same value always assigns to the same thread.

It is required that the rule function handles its exceptions and does null checks and returns `non-null` values. Failure to do so causes the message allocation to a thread to fail also. In other words, the message fails to propagate.

The rule function should be lightweight and should only perform read-only operations on the event or its payload.

Note: It is only a convenience mechanism to identify, compute, and return the relationship key. It should not perform the wider range of operations that are allowed in rule functions used elsewhere, such as in pre-processors or in rules, such as `load`, `create`, `update`, `delete objects`, `acquire locks` and so on. Doing so may cause unknown behavior.

Startup Functions and Shutdown Functions

Select function collections and individual functions as needed, to define which functions run at engine startup and shutdown respectively.

Tip: How Startup Rule Functions and Shutdown Rule Functions are Executed

- The order of the functions (including the order of functions within collections) is the order in which they run at runtime.
- Put startup rule functions (for use at startup) into different collections from shutdown rule functions (those used at shutdown) so you can select them appropriately on the agent classes tab.

Log Configurations

Also on the Collections tab, you can add different log configurations. These are used to when you configure processing units.

See Log Configurations.

Configuring Collections of Rules, Rule Functions, and Destinations

The purpose of configuring collections of rules, rule functions, and destinations is to make it simpler to configure agent classes. When you configure an agent, you can add collections of resources or individual resources or both. Two collections are predefined: an all-rules collection and an all-functions collection.

The procedure is in general the same for rules, rule functions, and destinations, so in these instructions, the word *item* is used to refer to any rule or rule function or destination.

A collection can have references to items (rules, rule functions, or destinations), and also references to other collections of the same type. References are identified in the groups tree by a reference symbol (↗). This mechanism enables you to reuse collections for more efficient configuration.

For the log configurations procedure, see Configuring Log Configurations

Procedure

1. On the Collections tab do any of the following:
   - To add a new collection, select the parent for the collection type, Rules, Destinations, Functions, or Log Configurations as needed, and click **Add**.
   - In the Item Collection field that appears on the right, enter a name for the group and click **Add** again.

     Ensure that the collection name is unique across different collections in the CDD. For example, Rule collections and Destination collections in the CDD cannot have the same name.
   - To add *items* and *item* group references to a collection, select the item collection and then click **Add**.

     You see the Select Items dialog.
2. In the Select *Items* dialog do any of the following:
   - To add *items*, in the **Items** tree click the checkboxes of *items* you want to add to the group you are defining.
   - To add collection references, in the **Collection References** tree click the checkboxes of collections you want to add (by reference) to the collection you are defining.

     When you select a collection on the left, you see details on the right: For example, the path to item you selected, and the names of collections you selected.
3. For function collections only, reorder the functions as needed, so that they execute in the correct order at runtime (that is, at startup or shutdown). Highlight a rule function in the tree on the left, and then click Move Up or Move Down as needed.
4. For destination collections only, configure each destination in turn. Select the destination on the left and complete the settings on the right to define characteristics such as the threading model to use, and the event preprocessor. See CDD Collections Tab Input Destination Settings Reference for information about each setting.
5. Save the CDD.

Updating Collections

You can update the already configured collections in several ways.

Procedure

1. To remove an item in a collection or the collection itself, select the item or the group on the left and click **Remove**.
2. To reorder rule functions in a function collection, select a rule function in the tree on the left, then click **Move Up** or **Move Down**. This is important for startup and shutdown rule functions. Ensure that you put startup and shutdown rule functions into appropriate separate collections.
3. You can change the URI (project path) of project resources to match their actual locations. To change the URI of an item, select the item on the left and edit the URI on the right.

CDD Collections Tab Input Destination Settings Reference

These agent-specific destination configuration settings are also available from the Agent Classes Tab. Collections enable you to configure once and use in multiple agents.

CDD Collections Tab Input Destination Settings

Property	Notes
Input Destination ID	Uniquely identifies this destination configuration at runtime. Edit as needed to ensure that each destination in the cluster has a unique deployment name. Default value is destination name.
URI	Project path to the destination (that is path to the destination in the design-time project).
Preprocessor	Specifying an event preprocessor for a destination is optional. **Tip** If you specify a preprocessor, you generally also specify worker thread settings, because event preprocessors are multithreaded (unless Caller’s Thread threading model is used, which is single-threaded). Select the rule function that has been configured as this destination’s

Property	Notes
	event preprocessor. Event preprocessors are rule functions with one argument of type simple event.
Threading Model	If you specified a preprocessor, also specify thread settings. Select one model: **Shared Queue** Uses the TIBCO BusinessEvents system-wide shared queue and threads. For queue size and number of threads settings, see CDD Agent Classes Tab Settings Reference. **Caller's Thread** Uses the thread (and queue size) provided by the channel resource client. There is one thread per destination. **Note** If it is important to ensure that acknowledgements are sent in the expected order with Caller's Thread threading model, do not use the parallel operations option. See `Agent.`*`agentClassName`*`.enableParallelOps` in CDD Agent Classes Tab Properties Reference . **Destination Queue** TIBCO BusinessEvents creates a dedicated thread pool and set of worker threads in each destination. See Thread Count and Queue Size below. For more information on threading models see Threading Models and Tuning in *TIBCO BusinessEvents Architect's Guide*.
Thread Count	If you specified Destination Queue in the Threading Model setting, specify the number of threads for this destination here.
Queue Size	If you specified Destination Queue in the Threading Model setting, specify the queue size for this destination here.

Agent Classes (All OM Types)

Agent class types are as follows:

Cache Agent

Used only with Cache OM. Cache agents handle distributed cache object storage. A processing unit can contain only one cache agent.

Inference Agent

Used with all OM types. For inference agent classes, you distribute a project's resources among the agent classes to define the specific work each agent will do. In Memory OM uses only inference agents, and each agent operates independently. With Cache OM or Store OM, the agents share the data (as explained in *TIBCO BusinessEvents Architect's Guide*).

Query Agent

Query agents use an SQL-like query language. You can query data that is in the cache or store. You can also query data arriving in events, known as event stream processing or ESP.

For details, see *TIBCO BusinessEvents Event Stream Processing Query Developer Guide*.

Adding an Agent Class

You can begin by configuring classes provided by the wizard. You can rename the classes as desired. Then add more classes as needed.

Procedure

1. In the TIBCO BusinessEvents Studio, open the CDD file in the CDD editor.
2. Select the Agent Classes tab and click **Add Agent**.
3. In the New Agent Class dialog, enter an Agent Class Name and select an Agent Class Type. from the list (see section introduction). Valid types for your project depend on object management type, and whether you use any TIBCO BusinessEvents add-on products. Click **OK**.

4. The new agent class name appears on the left. Select the agent class name. Appropriate settings for that agent type appear in the Configuration section.
    - Complete the settings as explained in CDD Agent Classes Tab Settings Reference
    - Add any properties as needed. The available properties are explained in CDD Agent Classes Tab Properties Reference.
5. For inference, dashboard, and query agent class types, add the resources you want to use. In the agent class tree on the left, click each type of resource collection in turn and configure as explained next.

   (In the instructions below, the word *item* stands in for destination, function, and rule depending on the collection category.)
    a. Highlight a category of collections (for example Input Destination Collections).
    b. Click **Add**. You see the Select *items* dialog.
    c. In the upper section of the dialog, select individual project *item* resources, as desired.
    d. In the lower section of the dialog (the Reference Groups section), select *item* collections you defined earlier, as desired.
    e. Click **OK**. A list of *item* IDs appears in the box on the right.
6. If you added any individual destinations to the Input Destinations Collections category, highlight their name on the left and configure their settings on the right. See CDD Collections Tab Input Destination Settings Reference for details.

   (Destinations within input destination collections are configured at the Collections tab.)
7. Do any of the following as needed:
    - Click a collection category on the left to see a list of collections and *items* you selected from that category on the right.
    - Expand a category on the left and click a collection reference within it. You see a list of its item IDs and paths, and any collection references within that collection, on the right.
    - Edit the project paths for individual items you add here. You would do this only if the project location of that item changed.
8. Save the CDD.

CDD Agent Classes Tab Settings Reference

The following tables explain settings used with inference agents, and query agents.

CDD Agent Classes Tab Inference Agent and Query Agent Settings

Setting	Notes
Inference Agent and Query Agent Settings	
Max Size (Local Cache)	Specifies the maximum number of objects (entities) in each agent's L1Cache (inference agent) or local cache (query agent). The L1 cache is a local cache used by the inference agent for local access to recently used objects. It is used to optimize access to objects. The query local cache is used in a way similar to the inference agent L1Cache. The query agent's local cache stores cache data locally for efficient reuse. The local cache listens to and synchronizes the locally stored entity instances with those in the main cache, so that the local cache stays up-to-date. When the threshold is reached, oldest entities are removed first. Default is `1024` (unit is objects).
Eviction Time (Local Cache)	Specifies an age limit on the cached entities in seconds. After this period, they are removed from the local cache. **Note** Age resets each time an entity is accessed by a query engine. Default is `900`.
Queue Size (Shared Queue)	Used for destinations whose threading model is Shared Queue (see Threading Model in CDD Collections Tab Input Destination Settings Reference). Specifies the queue size for the processing unit-wide shared queue. **Note** In this release, set the same value for all agents configured to deploy in the same processing unit.

Setting	Notes
	You can also use global variables as values for this setting. If set to 0 (zero), the queue size is unlimited. Default is `1024`.
Thread Count (Shared Queue)	Used for destinations whose threading model is Shared Queue (see Threading Model in CDD Collections Tab Input Destination Settings Reference). Specifies the number of processing unit-wide shared threads. **Note** In this release, set the same value for all agents configured to deploy in the same processing unit. You can also use global variables as values for this setting. As a guideline, set the value to the number of processors available to the JVM. Default value is `10`.
Max Active	Specifies the maximum number of active agents of this class. This value is used for fault tolerance. Deployed agents that are acting as standby can take over from active instances that fail. In many cases, there is no need to keep standby instances. A value of `0` indicates an unlimited number of active instances. For example, the **Max Active** field is set to 1 and you start two BusinessEvents configured store engine. As per the **Max Active** field, only one engine is active while the other one is passive. If the active engine fails, the passive engine takes over and become active. See Fault Tolerance of Agents in *TIBCO BusinessEvents Architect's Guide* for more details. Default is `0`.
Inference Agent Settings	

Setting	Notes
BusinessWorks Repo URL	If this project will integrate with a TIBCO ActiveMatrix BusinessWorks project, enter the Repo URL for the ActiveMatrix BusinessWorks project repo URL here. Use forward slashes.
Concurrent RTC	If checked, enables concurrent run to completion cycles, generally shortened to RTC cycles. (Also known in prior releases as concurrent Rete and `concurrentwm`). Concurrent RTC does not require cache OM but does require local locking. The number of concurrent cycles is determined by the number of available threads. See Collections, Agent Classes, and Processing Units for details. Also see Concurrency and Project Design in *TIBCO BusinessEvents Architect's Guide* for important information on using concurrency features.
Check for Duplicates	By default, TIBCO BusinessEvents checks if the external IDs (`@extId`) of entities are unique within the current agent. To check for uniqueness of external IDs across the cluster, check this checkbox. Performing this check affects performance. Default is unchecked.

CDD Agent Classes Tab Properties Reference

Properties are available for inference agents, cache agents, and query agents.

Inference Agent and Query Agent Properties

`com.tibco.cep.runtime.channel.payload.validation`

XML event payloads are validated when this property is set to true. There may be some loss of performance due to the extra processing.

Default is `false`.

`com.tibco.cep.runtime.threadpool.shutdown.timeout.seconds`

Specifies the time (in seconds) to wait for the worker thread to complete before shutdown.

Inference Agent Properties

`Agent.`*`AgentClassName`*`.recoveryPageSize`

Specifies the number of entries per page to be used while recovering objects from the cache.

For example, if you set the value to 10,000, then the engine loads handles in blocks of 10,000, instead of trying to load them in a single batch. Smaller batch sizes result in slower recovery. Experiment with batch size to establish the best batch size to use for your environment.

A value of 0 means that the objects are recovered in one iteration.

Default is `0`.

`Agent.`*`AgentClassName`*`.cacheTxn.updateCache`

Used only if a cache-aside database write strategy is used.

If set to false: When a rule action changes the value of an entity's properties or adds a new entity, then the entity instance is evicted from the cache instead of updating it. Updates are saved in the backing store. Use this setting and `Agent.`*`AgentClassName`*`.threadcount` as needed to improve performance and cache memory management.

This property interacts with the Cluster > Domain Objects setting, Evict From Cache on Update (and its override settings if any):

- When this CDD property is set to true, the domain objects Evict From Cache on Update setting is ignored, in the agent for which the property is set.
- When this CDD property is set to false, the domain objects Evict From Cache on Update setting overrides this CDD property.

  See Domain Objects Default Settings Reference for details on the Evict From Cache on Update setting.

  Possible values are `true` and `false`.

  Default is `true`.

> **Note:** `AgentName` in the property refers to the name of the agent that actually evicts entities, for example, `inference-class`.

`Agent.`*`AgentClassName`*`.threadcount`

For use with cache aside and only when parallel operations feature is used (see `Agent.`*`agentClassName`*`.enableParallelOps`).

Defines the number of `$CacheWriter` threads performing cache writing jobs.

See Threading Models and Tuning in TIBCO BusinessEvents Architect’s Guide for usage guidelines.

The default value is `2`.

`Agent.`*`AgentClassName`*`.checkDuplicates`

This property is available only when the legacy id lookup strategy is enabled for a project.

This property affects how TIBCO BusinessEvents checks the uniqueness of entity external IDs (@extId).

If set to false, checks for uniqueness of external IDs within the agent

If set to true, checks for the uniqueness of external IDs across the cluster. Performing this check affects performance so use it with care.

Default is false.

Inference Agent Database Writer Thread Tuning Properties

> **Note:** For use with cache aside and only when the parallel operations feature is used.
>
> For a guide to usage, refer to the "Database Write Tuning Options for Cache Aside" section in the *TIBCO BusinessEvents Architect’s Guide*.

Agent.*agentClassName*.enableParallelOps

If true, parallel operations are used

Post-RTC phase operations are done in parallel:

- Writes to the cache
- Writes to the database (only relevant cache aside strategy is used)
- Runs the actions list, for example, sends messages (events) and acknowledges events as needed.

  Use of parallel operations generally requires the use of locking to ensure data integrity.

If false, sequential operations are used

All post-RTC phase operations are done by the worker thread in the order shown above. When concurrentRTC is enabled, multiple worker threads perform post RTC operations in parallel (locking required).

This property is set to false for specific needs such as when the Caller's Thread threading option is used.

Another reason to set the value to false is to ensure that the system waits to send a reply event confirming that some work has been done, until the result of the work can be seen in the cache.

Defaults to true only if cache-aside write strategy and concurrent RTC are both used. Otherwise defaults to false.

Agent.*AgentClassName*.dbthreadcount

Defines the number of database write threads available to process the RTC transactions from the queue, that is, the number of threads performing database writing jobs (`$DBWriter` thread pool). Writes include applying entity inserts, updates, and delete to the database.

Although the limit is seldom reached, you can guarantee that a connection is always available for a `dbwriter` thread as follows. Set this field to the same value as **CDD Cluster tab > Persistence Store > Connection > Max Size field**.

Default is `2`.

Agent.*AgentClassName*.dbOpsQueueSize

The size of the queue (a Java blocking queue) for database writing jobs.

Zero (0) or a negative value means that the queue size is unlimited.

Default is `8`.

> **Note:** When the queue is full, all engine operations are blocked.

`Agent.`*`AgentClassName`*`.dbOpsBatchSize`

Used in the post-RTC phase. Sets the maximum number of RTC transactions that a database writer thread takes from the database operations queue and processes in one batch.

Database write threads process the RTC transactions from the queue. The number of threads is defined by `dbthreadcount`.

A database write thread takes up to the `dbOpsBatchSize` number of RTC transactions, processes them and commits them to the database. (When database write threads are idle, they take available jobs from the database operations queue, even if there are less jobs than `dbOpsBatchSize`.)

Default is `10`.

Query Agent Properties

`be.agent.query.localcache.prefetchaggressive`

If set to true, then the objects required for a query are prefetched while the query is running.

The prefetch feature improves performance, but CPU and memory usage increases as a result of the aggressive prefetching. You may have to try different values until you find the optimal settings for your environment.

Ensure that the cache size is large enough to accommodate objects that are prefetched.

Default is `false`.

`be.engine.queryAgent.channel.disable`

By default, query agents connect to channels. In some cases, however, query agents do not need to connect to channels. To prevent query agents from connecting to channels, set this property to `true`.

The default value is `false`.

Cache Agent Properties

`be.engine.cacheServer.channel.disable`

By default cache agents connect to channels. In most cases, however, cache agents do not need to connect to channels. To prevent cache agents from connecting to channels, set this property to `true`.

The default value is `false`.

`be.engine.cluster.scheduler`

A single agent of the cluster acts as a scheduler. By default all agents of a cluster can act as a scheduler. To avoid that a Cache Agent acts as a scheduler, add the property and set it to false for the Cache Agent classes.

The default value is true

`be.engine.cluster.datagrid.scheduler.persistence.policy`

Specifies the type of communication to be used to maintain persistence in the scheduler cache: asynchronous (`ASYNC`) or synchronous (`SYNC`). This property overrides the **Persistence Policy** field (set at the cluster level) for the scheduler. This property has no default value and if not set, it takes the value of the **Persistence Policy** field.

- `ASYNC` - This policy is recommended to avoid frequent IO operations, which can slow cache agents.
- `SYNC` - This policy is recommended for flushing out the completed entries from the scheduler caches.

`com.tibco.cep.runtime.scheduler.registerCacheEvents`

The scheduler processes events in the following ways:

1. Periodically looking into the cache and scheduling and expiring those events.

2. Listening to puts on the cache and as soon as the put happens, check if it can be scheduled.

When there is a continuous load, Apache Ignite cache events (puts) are scheduled even though they are not required to be scheduled immediately. In most cases, you do not schedule events to be triggered immediately. This cause Ignite buffer limits to exceed for a slower network. Because of this, nodes get disconnected and reconnected continuously.

To disable such inline processing of scheduler events, set the property `com.tibco.cep.runtime.scheduler.registerCacheEvents` to `false`.

The default value is `true`.

Log Configurations

Each processing unit references a log configuration. The log configurations are defined on the **Collections** tab.

You can replace the default line layout implementation with your own. You can also override the default logging mechanism in TIBCO BusinessEvents to use the log4j mode. See Overriding the Default Logging Mode for more details. For some custom log4j configuration examples, see Custom Log4j Configuration Examples.

For more information about log4j, see Apache wiki page.

Log File Settings

For a reference to the settings, see the CDD Collections Tab Log Configurations Settings Reference in CDD Collections Tab Log Configurations Settings Reference.

- Log File Name and Location

  Set the name and location of the log file for a log configuration using the Name and Directory settings. If you do not enter a leading slash, the files are stored relative to the working directory (the directory in which you start the `be-engine.exe` executable). If you do not specify a name, the engine name is used. If no engine name is specified the name defaults to `<hostname>.log` or `cep-engine.log`.

- Number and Size of Log files

  You can also set the size of a single log file, the number of files to keep, and whether a log file is flushed when an engine starts, or whether entries are appended. Log Configuration Levels and Syntax

Log Configuration Levels and Syntax

In a log configuration that uses the provided line layout implementation, you select a level of logging for each *module* in the TIBCO BusinessEvents runtime.

Levels

A level corresponds to how much logging is filtered out. They are ordered where `all` is lowest and `off` is highest:

Level	Description
Off	Highest possible rank. Filters out all logging messages (turns logging off for the specified module).
Fatal	Logs only severe runtime errors that cause the application to stop running.
Error	Also logs runtime errors that might not cause the application to stop running.
Warn	Also logs potentially harmful runtime events or situations.
Info	Also logs runtime informational events of general interest.
Debug	Also logs detailed runtime informational events, for use in identifying issues.
Trace	Also logs even more detailed runtime information.
All	Lowest possible rank. Turns on all logging including any custom logging levels.

Syntax

Enabling a lower level automatically enables the higher levels. For example, enabling `info` automatically enables `fatal`, `error`, and `warning`.

Assign each module to a level using a space-delimited list. The levels are not case sensitive. The syntax is as follows:

```
module1:level module2:level ...
```

To assign a certain level of logging to *all* modules, use an asterisk:

```
*:info
```

This syntax means that logging for all modules is at the `info` logging level.

You can use the asterisk syntax and also specify exceptions that use a different logging level. For example:

```
*:info  driver.tibrv:debug
```

This syntax means that all modules use logging level `info`, except the module `driver.tibrv` which uses `debug` level.

The Apache Ignite log also follows the CDD log configurations. To use a different logging level than the one configured in the CDD for Apache Ignite logs, set the module `org.apache.ignite` to the desired logging level. for example:

```
*:info org.apache.ignite:debug
```

This syntax means that all modules use logging level `info`, whereas Apache Ignite uses logging level `debug`.

Configuring Log Configurations

See Log Configurations for an explanation of the logging levels, modules, and syntax details.

Procedure

1. On the Collections tab, select Log Configurations and click **Add**.
2. In the Configuration section, give the log configuration a name.
3. Add the log levels that you want to enable in this configuration. See CDD Collections Tab Log Configurations Settings Reference for details on the fields.
4. To redirect the STDERR or STDOUT streams, check the **Enable** checkbox.

   For details, see CDD Collections Tab Log Configurations Settings Reference.
5. Save.

6. To specify a Custom Line Layout class, on the Collections tab select Log Configurations and click **Add**.
7. In the Configuration section, give the log configuration a name.
8. In the Custom Line Layout section, check the **Enable** checkbox, and complete the Class and Argument fields as shown in CDD Collections Tab Log Configurations Settings Reference.
9. Save.

Overriding the Default Logging Mode

You can disable the default logging mode and use the log4j mode for logging.
After the default logging mechanism is disabled, the log4j mode for logging is automatically activated. The default log4j configuration is located at `BE_HOME\lib\ext\tpcl\apache\log4j2.xml`. You can also specify your custom log4j file location to be used for logging in the `be-engine.tra` file.

Procedure

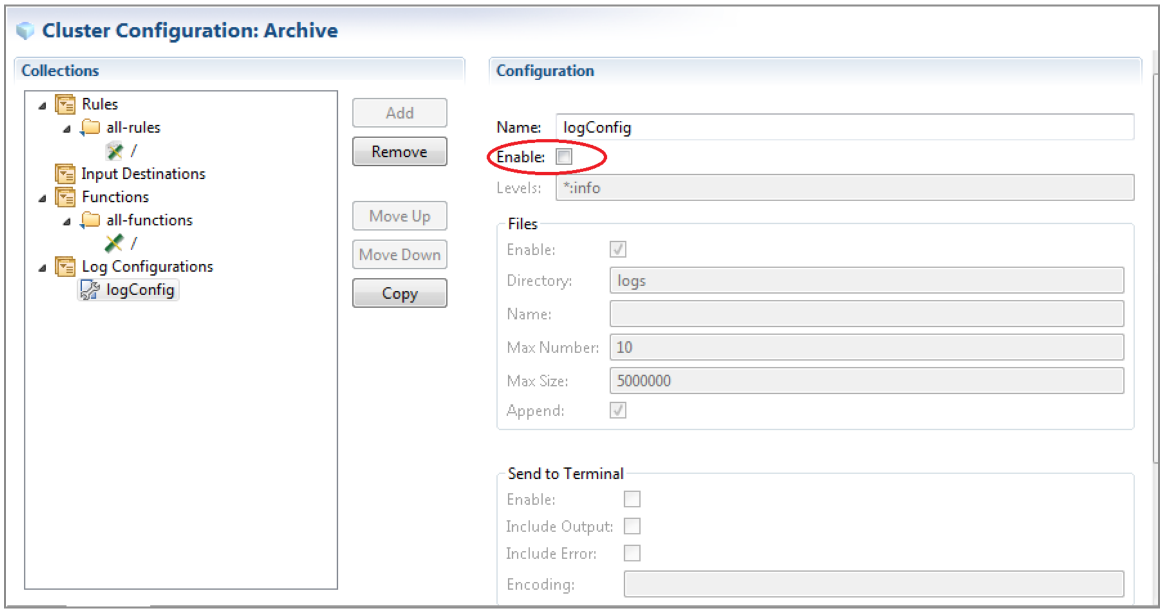
1. On the **Collections** tab of the project's CDD, select an existing log configuration under the Log Configurations.

   For details on how to add a log configuration, see Configuring Log Configurations.

   The log configuration properties are displayed.

2. Clear the **Enable** checkbox to disable the default logging mechanism.

   Log Configurations Properties

   Name: logConfig
   Enable: ☐
   Levels: *:info
   Files
   Enable: ✓
   Directory: logs
   Name:
   Max Number: 10
   Max Size: 5000000
   Append: ✓
   Send to Terminal
   Enable: ☐
   Include Output: ☐
   Include Error: ☐
   Encoding:

3. Save the CDD file.

4. To use your custom log4j configuration, add the following property in the `be-engine.tra` file.

   This property is specified in the `be-rms.tra` file for RMS and in the `log4j.properties` file for TIBCO BusinessEvents Enterprise Administrator Agent as shipped.

   ```
   java.property.log4j.configurationFile=[custom_log4j_path]
   ```

   Note: If the file is not in the classpath but located elsewhere on your device, use the following path:

   ```
   java.property.log4j.configurationFile=file:///[custom_log4j_file_path]
   ```

Note: In TIBCO BusinessEvents, HTTP response logs are by default set to `Info` level. To get the HTTP response debug logs for JSON and similar services, you must update the `org.apache.http` logger and set its log level to `Debug` in the `log4j2.xml` file.

Custom Log4j Configuration Examples

You can create your own custom log4j configurations that you can use for logging instead of the default `log4j2.xml` file.

You can also see the official log4j wiki page for more information on log4j configurations at the Apache Log4j 2 webpage.

Following are the few example XML configurations for some logging use cases, which you can use after overriding the default logging (see Overriding the Default Logging Mode for more information).

Multiple Log Files Using Single log4j Configuration

Define multiple FileAppender classes to create multiple logging by adding appender reference to the respective loggers as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
    <Appenders>
        <Console name="STDOUT" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{yyyy MMM dd HH:mm:ss.SSS
'GMT'XXX} %p [%c{3}] - %m%n" charset="UTF-8"/>
        </Console>
        <RollingFile name="RollingFile1" fileName="${LOG_
DIR}/application.log" filePattern="${LOG_DIR}/application.%d{yyyy-MM-
dd}.log.gz" ignoreExceptions="false">
        <PatternLayout>
            <Pattern>%d{yyyy-MM-dd HH:mm:ss} %-5p %m%n</Pattern>
        </PatternLayout>
        <Policies>
                <SizeBasedTriggeringPolicy size="100KB" />
        </Policies>
        <DefaultRolloverStrategy max="5" />
        </RollingFile>
        <RollingFile name="RollingFile2" fileName="${LOG_
DIR}/action.log" filePattern="${LOG_DIR}/action.%d{dd-MMM}.log.gz"
ignoreExceptions="false">
            <PatternLayout>
                <Pattern>%d{yyyy-MM-dd HH:mm:ss} %-5p %m%n</Pattern>
            </PatternLayout>
            <Policies>
                <SizeBasedTriggeringPolicy size="100KB" />
            </Policies>
            <DefaultRolloverStrategy max="10" />
        </RollingFile>
    </Appenders>
    <Loggers>
        <Logger name="com.tibco.rta.action" level="info"
additivity="false">
            <AppenderRef ref="STDOUT"/>
            <AppenderRef ref="RollingFile1"/>
            <AppenderRef ref="RollingFile2"/>
        </Logger>
            <Logger name="org.apache.catalina" level="info"
additivity="false">
            <AppenderRef ref="STDOUT"/>
            <AppenderRef ref="RollingFile1"/>
        </Logger>
    <Root level="info">
        <AppenderRef ref="STDOUT"/>
        <AppenderRef ref="RollingFile1"/>
        <AppenderRef ref="RollingFile2"/>
    </Root>
    </Loggers>
</Configuration>
```

Note: When you copy a code snippet from the documentation and paste it in your environment, it might contain unwanted spaces in-between. It is best practice to verify the copied code snippet before using it.

Rotating Log Files Based on Time

If using RollingFileAppender, then use TimeBasedRollingPolicy to specify when to roll over log files based on date time. The FileNamePattern property defines the name pattern for rolled over files. In the given example, it creates daily a log file (max 5 files).

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
       <Appenders>
       <Console name="Console" target="SYSTEM_OUT">
       <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} -
%msg%n"/>
       </Console>
               <RollingFile name="RollingFile" fileName="logs/myapp.log"
filePattern="logs/myapp-%d{yyyy-MM-dd}.log">
               <Policies>
               <TimeBasedTriggeringPolicy interval="1" modulate="true"/>
```

```
                </Policies>
                <DefaultRolloverStrategy>
                <Delete basePath="logs" maxDepth="1">
                <IfFileName glob="myapp-*.log">
                        <IfAny>
                                <IfAccumulatedFileCount exceeds="5"/>
                        </IfAny>
                </IfFileName>
                </Delete>
                </DefaultRolloverStrategy>

        <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} -
%msg%n"/>
                </RollingFile>
        </Appenders>

        <Loggers>
                <Logger additivity="true" level="debug" name="kernel.core">
                        <AppenderRef ref="Console"/>
                        <AppenderRef ref="RollingFile"/>
                </Logger>
                <Root level="INFO">
                        <AppenderRef ref="Console"/>
                        <AppenderRef ref="RollingFile"/>
                </Root>
        </Loggers>
</Configuration>
```

The following table displays the example values of the DatePattern parameter that defines the time when to roll over the logs.

DatePattern Values For DailyRollingFileAppender

Time	Value
Minutely	`.%d{yyyy-MM-dd-HH-mm}`
Hourly	`.%d{yyyy-MM-dd-HH}`
Half-daily	`.%d{yyyy-MM-dd-a}`
Daily	`.%d{yyyy-MM-dd}`

Time	Value
Weekly	`.%d{yyyy-ww}`
Monthly	`.%d{yyyy-MM}`

CDD Collections Tab Log Configurations Settings Reference

You can define settings for log file on the **Collections** tab of the CDD.

CDD Collections Tab Log Configurations Settings

Property	Global Variable?	Description
Name	No	Name of this log configuration.
Enable	No	Check the Enable checkbox to enable this log configuration. All other Enable settings are ignored if this checkbox is unchecked.
Levels	Yes	Space-separated list of levels and modules used in this log configuration. See Log Configurations for more details. Default is `info`.
Files		
Enable	No	Check the Enable check box to enable log files to be written. Configure the settings in this section to specify details. If this check box is unchecked, all other properties in this section are ignored.
Directory	No	Enter the absolute path to the directory in which you want to store the files. If you do not enter a leading slash, the files are stored relative to the working directory (the directory in which you start the `be-engine.exe` executable).

Property	Global Variable?	Description
Name	No	Name of the log file. The default value is the engine name. If no engine name is set, then the default value is `cep-engine.log`
Max number	Yes	Number of log files to keep. When the Max size setting value is reached, a new log file is created for the next log entries. Files are created up to the Max number setting size. The oldest file is deleted when a new file is added after this value is reached. Default is `10`.
Max size	Yes	Maximum size of one log file. Default is `10000000`.
Append	No	If checked then new entries are added to the end of the file. If not checked, the contents of the file are flushed each time the engine starts.

CDD Collections Tab Log Configurations Settings

Property	Description
Send to Terminal	
Enable	Check the Enable checkbox to enable the redirection specified in this section. If this checkbox is unchecked, all other properties in this section are ignored.
Output redirection	If true, the `STDOUT` stream is written to the terminal. If `false`, it is not.
Error redirection	If true, the `STDERR` stream is written to the terminal. If `false`, it is not.
Custom Line Layout	

Property	Description
Enable	Check the Enable check box to enable the custom line layout entries to take effect. Configure the settings in this section to specify details of a custom layout. If this check box is unchecked, all other properties in this section are ignored. If this check box is checked all properties in the other sections are ignored (except Name, and Enable in the upper section).
Class	The custom line layout class. This class must implement `org.apache.logging.log4j.core.layout.PatternLayout` and must be available in the runtime classpath. The class needs 2 constructors: • One with no argument • One with a single String argument, which receives the value of the Arguments field.
Arguments	A String parameter used for the custom line layout class, if required: • To use the constructor that requires an argument, specify the argument • To use the constructor that does not expect an argument, leave the field empty.

Note: To override the default TIBCO BusinessEvents CDD logging properties, see Overriding the Default Logging Mode.

Additional log settings can be added to the **Processing Unit** tab **Properties**, see Processing Units Configuration Settings.

Logging for the Legacy ActiveSpaces Cluster

You must configure logging for the Legacy ActiveSpaces Cluster in TIBCO BusinessEvents separately. To do so, set the properties in the CDD file.

Properties in the CDD File

Property	Description
`be.engine.cluster.as.log.dir`	The directory to which the Legacy ActiveSpaces log files will be written. If unspecified, the logs will be written to the same directory as the TIBCO BusinessEvents logs.
`be.engine.cluster.as.log.filename`	File name of the log file. By default, the file name is <*engineName*>-*as*.log.
`be.engine.cluster.as.log.level`	The log level specifying how much logging is to be filtered out. See Configuration Levels for the Legacy ActiveSpaces Cluster Logging for details. By default, the log level is set to INFO.
`be.engine.cluster.as.logfile.count`	Specifies the number of rolling log files allowed. Count is specified in integer.
`be.engine.cluster.as.logfile.size`	Log files are rolled over to a new log file when the specified size limit is reached. Size is specified in bytes.
`be.engine.cluster.as.logfile.append`	Set to true/false, specifies whether to append the logs to new files.

Configuration Levels for the Legacy ActiveSpaces Cluster Logging

Following levels can be used to specify the level of logging for the Legacy ActiveSpaces cluster logs. Note that the logging levels are case insensitive.

Level	Description
None	Highest possible rank. Filters out all logging messages (turns logging off for the specified module).

Level	Description
Fatal	Logs only severe runtime errors that cause the application to stop running.
Error	Also logs runtime errors that might not cause the application to stop running.
Warn	Also logs potentially harmful runtime events or situations.
Info	Also logs runtime informational events of general interest.
Fine	Also logs detailed runtime informational events, for use in identifying issues.
Finer	Also logs even more detailed runtime information.
Finest	Lowest possible rank. Turns on all logging including any custom logging levels.

Logging for the Apache Ignite Cluster

You can configure the following property in the CDD file.

Property	Description
`be.trace.log.dir.ignite`	The directory to which the Apache Ignite log files are written. If unspecified, the logs are written to the `BE_HOME\bin` directory.

Configuring the Date Format in the Log Files

As per your requirement and locale, you can customize the date and time format used in the BusinessEvents logs.

Procedure

1. Add the `be.trace.date.format` property to the project's CDD file or `be-engine.tra`.

2. Set the value of `be.trace.date.format` property with the required date and time format.

   The date and time formats conforms to the formats of the java.text.SimpleDateFormat class of JDK 1.6, or the org.apache.commons.lang.time.FastDateFormat class of Apache. Refer the URL http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html for more information on the supported formats.

   For example, setting the value to "`yyyy MMM dd HH:mm:ss:SSS z`" translates to "`2015 Feb 23 10:47:23:456 IST`" in the logs.

   ```
   be.trace.date.format = "yyyy MMM dd HH:mm:ss:SSS z"
   ```

Processing Units (All OM Types)

To configure a processing unit (PU), you add the items you configured earlier, and any additional properties required. If you do not find a configuration item you require, click the appropriate tab and add it, then return to **Processing Units** tab and continue configuration.

One processing unit named `default` is provided out of the box. You can change this name. It has no significance, except that TIBCO Administrator expects a processing unit of this name by default, which can be useful for testing purposes.

PUs with Unique Agent Instance Properties

Depending on configuration, some processing units can be deployed more than once in a cluster. Others have unique configuration details that make them deployment-specific, that is, that limit them to being deployed only once in a cluster. Configuring the following CDD settings for a processing unit makes it deployment-specific processing unit:

Agent Key

At the processing unit node, you can associate a unique key with a selected agent class. This key identifies an agent instance uniquely at runtime. The purpose of the agent key is to retrieve scorecards from the backing store. Scorecards are local to an agent and the key enables the correct scorecard to be returned to the correct agent.

Note: Add agent keys for inference agents only. To avoid any error, keep the **Key** field for query agents blank.

Agent Priority

The agent priority determines which agents of a given class are active, when fault tolerance is used. Each deployed agent of an agent class can have a different priority; however, if the agent have the same priority then cluster decides which agents it has to activate.

See "Deployment-Specific Processing Units and Global Variables" section in *TIBCO BusinessEvents Administration* for other ways a processing unit can be deployment-specific.

Adding a Processing Unit

A processing unit deploys as a TIBCO BusinessEvents engine. One engine runs in one JVM.

Each processing unit contains one or more agents of different types. The main types are inference agents which performs the inferencing work, and cache agents, which manage the objects. For details on the fields present on the Processing Units tab, see Processing Units Configuration Settings.

Procedure

1. In the TIBCO BusinessEvents Studio, open the CDD editor.
2. In the CDD editor, select the **Processing Units** tab and do the following:
   - Select the default processing unit and configure it. You can rename it as needed.
   - Click **Add** to add more processing units as needed.
3. Specify the value of the configuration settings for the PU that you have selected.

   For details on the configuration settings, see Processing Units Configuration Settings.

   Note: For deployment, TIBCO Administrator by default looks for a processing unit called `default` and a CDD file called `default`.
4. In the Agents section, click **Add** and select an agent class.

5. If needed, assign to each agent a key and a priority.

   See PUs with Unique Agent Instance Properties for details.

6. In the **Properties** section, add any additional configuration properties as required.

7. Save the resource.

Processing Units Configuration Settings

CDD Processing Units Tab Settings

Property	Notes
Name	Enter a name that is unique across the cluster.
Log Configuration	Browse to and select a log configuration, configured on Collections tab. See Log Configurations for more details.
Hot Deploy	Check the checkbox to enable hot deployment for this processing unit. See *TIBCO BusinessEvents Administration* for details about hot deployment.
Enable Cache Storage	Check the checkbox to enable cache storage on this processing unit (PU). Settings available depend on the types of cache agents in the PU. • In PUs used to host cache agents: The check box is checked and cannot be unchecked. • In PUs that host inference agents or query agents (or both): ◦ If checked, the PU is used for storing cache data. ◦ If unchecked, the PU is not used for storing cache data. **Note:** Enable cache storage in PUs running inference and query agents for test deployments only. Not recommended in production. Default value for PUs containing inference agents or query agents (or both) is unchecked.
Enable DB	Check the checkbox to enable database concepts functionality for this

Property	Notes
Concepts	processing unit.
Agents Section	
Agent	Name of the agent class you selected. Agent classes are defined on the Agent Classes tab.
Key	Specifies a value that uniquely identifies an instance of an agent of this class at deploy time. Required for recovery of scorecards. Recommended in all cases, for situations that require an agent instance to be uniquely identified. The value for Key must uniquely identify the agent. No default value.
Priority	Specifies the priority of the agent for load balancing purposes. The priority indicates the order in which standby agents become active, and conversely, the order in which active agents become standbys, when new agents join the cluster. The lower the number, the higher the agent is in the activation priority list. For example, an agent with priority 2 has a higher priority than an agent with a priority of 6. For agents with equal priority, the cluster decides which ones to activate. No default value.

Load Balancer

Agents are configured to work cooperatively as routers and receivers to ensure that related messages arriving from queue sources are handled by the same agent, so that related information is available locally.

Currently, only queue messages from TIBCO Enterprise Message Service are supported for this configuration.

Load Balancing Options

Load balancing is available for messages arriving from queues. Do not use load balancing for topic-based or other broadcast sources.

Two kinds of load balancing configurations are available: basic load-balancing and content-aware load balancing. Both of these configurations support messages arriving from TIBCO Enterprise Message Service queue sources.

Every JMS destination that is configured to be an input destination runs in its own JMS Session. This provides good throughput on queues for processing, and less connections.

Basic Load Balancing

With basic load balancing, events from queue sources are automatically distributed between deployed instances of an agent class. To set up this kind of load balancing, you deploy multiple instances of an agent class that listens to a JMS destination. Each deployed agent instance runs in a different processing unit.

This method can be useful when there is no relationship between the events that would require them to be processed in a certain order. If the order or grouping of events received is important, use content-aware load balancing.

Content-Aware Load Balancing

With content-aware load balancing, all related events arriving from queues are routed to the same agent using a routing key.

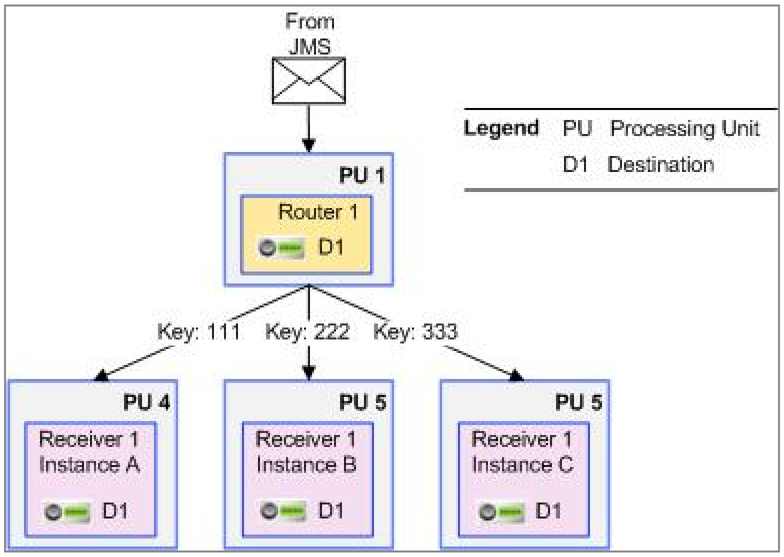
The key is formed using the event properties (single or multiple). For example, if the event property is ZipCode then a routing key is a specific zip code. All messages relating to one zip code are routed (over TCP) to the same agent, providing "session stickiness."

Content-aware load balancing uses *routers* and *receivers*. You can configure routers and receivers from the CDD Load Balancer tab. One receiver can handle more than one set of related events. For example if the routing key is a zip code, one receiver might handle events for multiple zip codes.

Use of content-aware load balancing simplifies project configuration, and makes runtime behavior more efficient. For example, only local locking is generally required (whereas basic load balancing requires cluster-wide locking). Also the L1 cache does not have to be checked for version consistency.

Note: The JMS message acknowledgment modes `EXPLICIT_CLIENT_ACKNOWLEDGE` and `EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE` are only supported.

Figure 3: CDD Based Router and Receiver Configuration



Routers

A router PU receives messages from the JMS server and routes them to appropriate receivers. Routers do no other work. For example, they should not execute rules.

A router PU contains one inference agent with one or more sets of JMS channels and destinations. Each destination has a default event. Values of one property or combination of multiple properties of that event are used at runtime as *routing keys*.

Event preprocessors can be used as needed to populate the routing key property, for example using some calculation or combination of other event properties.

The router redirects events over TCP to a receiver, based on the destination and the routing key values. The router transparently distributes the load across the available receivers. If a receiver agent fails, its messages (that is, messages with the key that the router was sending to that agent) are routed to another receiver and continue to be handled by that other receiver.
You can run multiple router instances using the same routing configuration, and they will all follow the same routing strategy. If one of the router fails, the one that is running will continue with routing process. No configuration is required for routers to work in fault tolerance mode as they do it intrinsically.

Receivers

Receivers are the inference or query agents that do the actual work. A receiver PU contains one inference or query agent. A set of receivers belongs to the same agent class. Receivers can also do other work, in addition to the work they receive from the router.

With CDD configuration, a receiver agent class is configured with one of the channel and destination configurations defined in the router. The destination, however, functions as a pseudo destination.

Content-Aware Load Balancer

When setting up content-aware load balancing, you first choose between two methods: pair configuration and adhoc configuration.

Pair Configuration

Preconfigured using the Load Balancer tab of the CDD.

The pair configuration uses a pair of processing units to act as routers and receivers. Two agent classes, a receiver inference class and a router inference class, are configured to use the same EMS destination. The router processes messages from the EMS queue and sends it to a receiver based on the routing key specified in the CDD.

See Load Balancer Configuration Fields for details.

Adhoc Configuration

Allows minimal preconfiguration using the Load Balancer tab of the CDD. Use catalog functions to implement the load balancer at runtime.

For details about Load Balancer catalog functions, see *TIBCO BusinessEvents Functions Reference* .

Only the load balancer name and the local destination are configured in the CDD. The router agent and receiver agent classes are configured using catalog functions in their rules and rulefunctions.

Note: Ensure that the destination has a default event associated with it.

For details, see:

- Creating the Load Balancer
- Configuring the Receiver

Creating the Load Balancer

The startup rule function has to be provided. It creates and returns a load balancer that can be used to send messages to load balanced remote destinations.

Procedure

1. Create the router TCP connection in a startup rule function.

   ```
   LoadBalancer.Router.createLoadBalancerTo(adhocConfigName);
   ```

2. Send Event to the Receiver .

   Use this rule function as an event preprocessor. It sends an event to a remote receiver. The router agent does not have any destinations. The routing decision is made using the routing key.

   ```
   void LoadBalancer.Router.send(Object
                                 loadBalancer , SimpleEvent
                                 event, String
                                 routingKey);
   ```

3. Discard the Load Balancer.

   Put this rule function in a shutdown rule function.

   ```
   Object loadBalancer = Collections.Map.remove(Collections.Map.getMap
   (String mapID), Object key);
   ```

   ```
   Collections.Map.deleteMap(String mapID);
   ```

4. This rulefunction discards the Load Balancer.

   ```
   void LoadBalancer.Router.discardLoadBalancer(Object loadBalancer);
   ```

Configuring the Receiver

Configuring a Receiver consists of obtaining information, creating the Receiver, creating the Receiver TCP connection, and discarding the Receiver.

Procedure

1. Use the following function for the Receiver side:

   `LoadBalancer.Receiver.*`

2. Obtain information for a local channel needed for the Receiver:

   ```
   int port = System.getSystemPropertyAsInt("receiver_localchnl_localdest_
   port", 34567);
   ```

3. Create a Receiver.

   The receiver object, which receives messages from a router, is created and returned. Messages will be received from the router on the local channel and destination specified.

4. Create the Receiver TCP connection in a startup rule function.

   ```
   Object LoadBalancer.Receiver.createTcpReceiverFor(String
   adhocConfigName
   ```

5. Discard the Receiver in the shutdown rule function.

   ```
   LoadBalancer.Receiver.discardReceiver(Object loadBalancedReceiver;
   ```

Load Balancer Configuration Fields

A load balancer is configured by specifying the receiver and router agents, and setting the destination

To configure the load balancer, use the CDD Load Balancer tab.

CDD Load Balancer Tab Properties

Property	Notes
Pair Configuration	
Name	Name of the pair configured load balancer.
JMS Destination	Destination used by the router and receiver agents. Ensure that the destination has a default event configured.
Key	Routing key used by the pair configuration.
Router	Router agent class for the load balancer configuration.
Receiver	Receiver agent class for the load balancer configuration.
Adhoc Configuration	
Name	Name of the adhoc load balancer. When creating a load balancer using the catalog function `createLoadBalancerTo`, the name of the load balancer must be specified. For example, `Object loadBalancer = LoadBalancer.Router.createLoadBalancerTo ("adhocLoadBalancerName");`
Local Destination	Destination used by the router and receiver agents. In an adhoc configuration, a local channel with a local destination is used to communicate between the router and receiver agents.
Properties	
transport	Transport to enable communication between the receiver and the router. Typically, this happens through an internal TCP connection. The default value is tcp.
hostname	Hostname where the load balancer is configured.

Property	Notes
	The default value is localhost.
port	Port number used by the specified transport.

Note: The `transport`, `hostname`, and `port` properties must be in lowercase.

Configuration Properties Reference

The configuration properties can be added to the CDD on various levels and it defines the scope of a property. Not all properties are valid at all levels.

JMS Server Connection Properties

You can add the JMS server connection properties at the cluster level (the **Cluster** tab) if they apply to JMS channels in all processing units in the cluster or you can add for the PU on the **Processing Unit** tab.

See *TIBCO BusinessEvents Developer Guide* for details about configuring a JMS channel.

JMS Server Connection Properties

Property	Notes
`com.tibco.tibjms.connect.attempts`	Specifies the number of reconnection attempts, and the interval between each attempt to connect to the JMS server. The value must use the format: *attempts,retry interval*. For example: `10,500` means 10 attempts, with a 500-millisecond interval between each retry attempt. This property is used only for channels that have a TIBCO Enterprise Message Service provider. **Note:** Use either `be.jms.reconnect.timeout` or `com.tibco.tibjms.connect.attempts`. If you set both the properties, then `com.tibco.tibjms.connect.attempts` takes precedence.

Property	Notes
	The property does not work for the JNDI connection. The default value is `2, 500`.
`be.jms.ignore.startup.error.channels`	Ignores the startup errors for the specified JMS URIs. You can specify a comma-separated list of JMS channel URIs as the value of the property. The default value is an empty string to retain existing behavior. Set the value to an asterisk (*) to ignore startup errors for all JMS channels. Also, set the property `be.jms.reconnect.timeout=<`*`value in seconds`*`>` to retry the connection to Enterprise Message Service servers.
`be.jms.reconnect.timeout`	Specifies the retry interval (in seconds) for reconnecting to the JMS server when the connection is broken. A value of zero (0) means do not retry. Any other value means keep retrying (with no limit to number of retries), and use the specified interval between each attempt. If you require incremental interval between the reconnect attempts then set the `be.jms.reconnect.timeout.incremental.enabled` property. Unacknowledged messages (Events) are resent to the TIBCO BusinessEvents engine, which may result in duplicate events. **Note:** Use either `be.jms.reconnect.timeout` or `com.tibco.tibjms.connect.attempts`. If you set both the properties, then `com.tibco.tibjms.connect.attempts` takes precedence.

Property	Notes
	The default value is `0` (zero)
`be.jms.reconnect.timeout.incremental.enabled`	Using this property the random and incremental delays between JMS reconnect attempts are available. Thus, the reconnect requests are spaced out and do not send too many connect requests at once to the JMS server. The intervals start with the value of `be.jms.reconnect.timeout` as the minimum value and keep on increasing with each attempt up to a certain maximum limit, beyond which the interval is always the same as the maximum limit. The default value is false. This property can only be used along with the `be.jms.reconnect.timeout` property.
`be.jms.reconnect.msgCodes`	Specifies a case-insensitive character pattern that matches all error messages or error codes that will cause a reconnect attempt. This property is used for JMS channels with providers other than TIBCO Enterprise Message Service. Default is * (that is, the wildcard matched by any characters.)

`be.channel.tibjms.queue.disabled`
 	
`be.channel.tibjms.topic.disabled`
	By default, be-engine connects to all defined channels on startup, including those not mentioned in the CDD file. This is because such channels can be used as output channels. However this is not always desired.
	
	To disable queue or topic connections for specific JMS channels, add the following properties as appropriate. Enter the project path to the JMS channel as the individual value. Use commas or

Property	Notes
	spaces as the delimiter. Use forward slashes in the project path. For example: `be.channel.tibjms.queue.disabled=/channels/1jmschannel, /channels/3jmschannel` `be.channel.tibjms.topic.disabled=/channels/2jmschannel, /channels/4jmschannel`
`be.channel.jms.unified`	By default, TIBCO BusinessEvents creates two connections to a JMS server, with the client IDs. Set this property value to `true` for all agents to create a single connection with the same client ID as specified in the channel resource properties, or in the JMS Connection shared resource, if used. In the unified mode only a single value is required. **Note:** When the connection is configured using a JMS Connection shared resource, ensure that the topic and queue connection factories on the JMS Connection shared resource **Advanced** tab match each other. Also, when using TIBCO Enterprise Message Service, use GenericConnectionFactory for both.
`be.channel.jms.disallow.dup.clientid`	Specifies whether the duplicate client ID can be used in the engine. When enabled, the engine fails to start when a duplicate client ID is encountered, even if the duplicate client ID is in the second engine. If the `be.channel.jms.disallow.dup.clientid` property is enabled in the default mode (not the unified mode), the JMS channel needs two client IDs (whitespace separated) to startup successfully (one for the queue connection, and one for the topic connection). In the unified mode, only one client ID is required.

Property	Notes
`be.channel.jms.sender.session.pool.maxsize`	Specifies the maximum pool size for the JMS sender session pool. Once set, it activates the JMS sender session pool for each JMS channel in the project. By default no sender session pool is created, and a single shared session is used by sending functions per channel. This property is applicable for the non-transacted mode only. The minimum allowed value of the property is `1`.
`be.channel.jms.session.recover.onAckFailure`	Specifies whether to autorecover a session that failed after acknowledgment failure of unexpired messages. To stop session autorecovery for such failures, set the `be.channel.jms.session.recover.onAckFailure` property to `false`. The default value is `true`.
`be.jms.error.endpoint.enable`	Specifies whether to enable forwarding of messages to error queue or topic for all destinations. If this property is enabled, TIBCO BusinessEvents forwards any unsupported message from the original destination queue or topic to an error queue or topic. These unsupported messages are then cleared from the original destination queue. The original message is acknowledged by BusinessEvents and proper error message is thrown for the unsupported message. If this property is enabled then you might also want to define `be.jms.default.error.queue.name` and `be.jms.default.error.topic.name` properties. The default value is false.

Property	Notes
	Note: The unsupported type messages are forwarded to the error queue or topic on the same JMS server, on which the original message arrived. Therefore, in case of an application interacting with multiple JMS servers, individual error queue or topic are required on each server.
`be.jms.default.error.queue.name`	Name of the error queue. The default value is `be.application.error.queue`.
`be.jms.default.error.topic.name`	Name of the error topic. The default value is `be.application.error.topic`.
`be.engine.channel.<`*`DestinationURI`*`>.deliverydelay`	Set the delayed delivery time (in milliseconds) for all messages using the JMS channel destination.
`be.channel.jms.session.lock.enable`	To enable the synchronized session for JMS receiver thread, set this property value to `true`. The default value is `false`.

HTTP Channel Connection Properties

To configure your HTTP channel application, you can add HTTP channel connection properties provided by TIBCO BusinessEvents in the system TRA file or on the **Cluster** tab in the project CDD file.

You can set the following properties by using the system properties also. You can set the system properties on the Java command line by using the `-Dpropertyname=value` syntax. For more information about Java HTTP client properties, see Java SE documentation.

HTTP Channel Configuration Properties

Property	Notes
`com.tibco.be.http.client.userAgent`	Specifies the custom `User-Agent` value to Override the `User-Agent` HTTP request header property. The default value is `TIBCO BusinessEvents/<version>`.
`com.tibco.be.http.client.useExpectContinue`	Enables or disables the expect-continue paradigm for a TIBCO BusinessEvents HTTP client. The default value is true.
`com.tibco.be.http.client.retryCount`	Specifies the number of retry attempts to connect to the HTTP client. By default, retry attempts are disabled.
`be.engine.channel.http.client.connectionTimeout`	Specifies the number of seconds to wait while a connection is being established. This property works when a new connection needs to be established. If a new connection does not need to be established and a connection can be reused from a previous request, then this timeout duration has no effect.
`com.tibco.be.http.root`	Specifies the directory path to store the temporary files used by the HTTP server. By default, the HTTP server will create a new directory every time

HTTP Channel Configuration Properties(Continued)

Property	Notes
	it starts. To avoid creating multiple directories, you can reuse an existing directory by specifying its path.
`com.tibco.be.http.client.maxConnPerRoute`	Specifies the total maximum connections per route for the HTTP client. The default value is `10`.

FTL Channel Connection Properties

To configure the FTL channel, you can add the FTL channel connection properties provided by TIBCO BusinessEvents in the system TRA file or on the **Cluster** tab in the project CDD file.

FTL Channel Connection Properties

Property	Description
`be.channel.ftl<destination URL>.com.tibco.ftl.client.subscriber.explicitack`	Set this property to true to enable explicit acknowledgment in the FTL channel.

Cache and Store Advanced Properties

Define the properties that you want to display on the UI. For each property, you can define the appropriate attribute, such as `displayName`, `type`, `default` and so on.

- Apache Ignite Advanced Properties

Apache Ignite Advanced Properties

Following tables list the advanced properties that you can add to the `store.xml` file for Apache Ignite. The table lists the property key names that are unique for each property and data type. You can provide the UI display name and default value as you want in `store.xml`.

For details about the attributes of the property tag, see XML Tags of the store.xml File.

IgniteConfiguration

The following properties define the configuration parameters required to start a grid instance.

Property	Type	Description
long-query-timeout	Long	Sets timeout in milliseconds after which a long query warning is printed.
management-thread-pool-size	Integer	Sets the management thread pool size to use within the grid.
metrics-expire-time	Long	Sets the time in milliseconds after which a certain metric value is considered expired.
metrics-history-size	Integer	Sets the number of metrics kept in history to compute totals and averages.

Property	**Type**	**Description**
metrics-log-frequency	Long	Sets the frequency of metrics log print out.
metrics-update-frequency	Long	Sets Apache Ignite metrics update frequency in milliseconds.
network-compression-level	Integer	Compression level for internal network messages.
network-send-retry-level	Integer	Sets the counts of message send retries.
peer-class-loading-thread-pool-size	Integer	Sets the thread pool size to use for peer class loading.
public-thread-pool-size	Integer	Sets the thread pool size to use within the grid.
query-thread-pool-size	Integer	Sets the query thread pool size to use within the grid.
rebalance-batch-prefered-count	Long	The number of batches generated by a supply node at the rebalancing procedure start.
rebalance-batch-size	Integer	The supply message size in bytes to be loaded within a single rebalance batch.
rebalance-thread-pool-size	Integer	Sets Max count of threads can be used at rebalancing.
rebalance-throttle	Long	Time in milliseconds to wait between rebalance messages to avoid overloading of CPU or network.
rebalance-timeout	Long	Rebalance timeout for supply and demand messages in milliseconds.
service-thread-pool-	Integer	Sets the service thread pool size to use within the grid.

Property	Type	Description
size		
system-thread-pool-size	Integer	Sets the system thread pool size to use within the grid.
system-worker-blocker-timeout	Long	Sets maximum inactivity period for system worker.
utility-cache-alive-time	Long	Sets keep alive time of thread pool size that will be used to process utility cache messages.
utility-cache-pool-size	Integer	Sets default thread pool size that will be used to process utility cache messages.
striped-pool-size	Integer	Sets striped pool size that should be used for cache requests processing.
enable-authentication	Boolean	Enables user authentication for Apache Ignite. **Note:** Authentication can be enabled only for the cluster with enabled persistence.

DataStorageConfiguration

The following properties define the configuration parameters required for persistence.

Property Key	Data Type	Description
write-full-page	Boolean	Sets flag that enforces writing full page to write-ahead log (WAL) on every change (instead of delta record).
checkpoint-frequency	Long	Sets the checkpoint frequency that is a minimum interval when the dirty pages are written to the persistent store.
checkpoint-lock-timeout	Long	Sets timeout for the checkpoint read lock acquisition.

Property Key	Data Type	Description
checkpoint-threads	Integer	The number of threads to use for checkpoint purposes.
concurrency-level	Integer	The number of concurrent segments in Apache Ignite internal page-mapping tables.
lock-timeout	Long	Timeout in milliseconds to wait when acquiring a persistence store locks file before failing the local node.
wal-archive-size	Long	Sets a maximum allowed size (in bytes) of WAL archives.
metric-enabled	Boolean	Sets flag indicating whether persistence metrics collection is enabled.
page-size	Integer	Configures the page size.
wal-archive-path	String	Configures the path for the WAL archive directory.
wal-buffer-size	Integer	The size in bytes of WAL buffer.
wal-compaction-enable	Boolean	Sets flag indicating whether WAL compaction is enabled.
wal-compaction-level	Integer	New archive level to WAL compaction.
wal-flush-frequency	Long	The frequency WAL will be fsync-ed in the BACKGROUND mode.
wal-fsync-delay-nanos	Long	Sets property that allows to trade latency for throughput in WALMode.FSYNC mode.
wal-page-compression-level	Integer	Sets algorithm specific page compression level.
wal-store-path	String	Sets a path to the directory where WAL is stored.
wal-record-buffer-	Integer	Sets property defining how many bytes iterator read from

Property Key	Data Type	Description
size		disk (for one reading), during go ahead wal.
wal-segments	Integer	The number of WAL segments to work with.
wal-segment-size	Integer	The size in bytes of a WAL segment.
wal-thread-local-buffer-size	Integer	The size of thread local buffer.
write-throttling-enable	Boolean	Sets flag indicating whether write throttling is enabled.
wal-page-compression	String	Configures algorithm specific page compression.
wal-mode	String	Configures the WAL mode.
wal-archive-after-inactivity	Long	Sets the time in millisecond to run the auto-archiving segment (even if incomplete) after the last record is logged.
checkpoint-threads	Integer	Sets the number of threads to use during a checkpoint
checkpoint-write-order	String	Order `(RANDOM/SEQUENTIAL)` of writing pages to the disk storage during a checkpoint.
cdc-wal-path	String	Configure the CDC WAL Path
data-storage-defrag-thread-pool-size	Integer	Defragmentation thread pool size
wal-min-archive-size	Long	Configure WAL archive minimum size
wal-force-archive-timeout	Long	Configure WAL force archive timeout

DataRegionConfiguration

The following properties define the configuration parameters required for a data region.

Property Key	Data Type	Description
default-data-region-initial-size	Long	Sets the initial memory region size defined by this data region.
default-data-region-max-size	Long	Sets the maximum memory region size defined by this data region.
data-region-initial-size	Long	Sets the initial memory region size defined by this data region.
data-region-max-size	Long	Sets the maximum memory region size defined by this data region.
data-region-metrics-enabled	Boolean	Enables metrics for this data region
data-region-empty-page-pool-size	Integer	Specifies the minimal number of empty pages to be present in reuse lists for this data region.
data-region-checkpoint-page-buffer-size	Long	Sets the amount of memory to be allocated for the checkpoint temporary buffer. When the checkpoint is in progress, this buffer is used to create temporary copies of pages that are being written to disk and updated in parallel.
data-region-eviction-threshold	Double	The eviction threshold of Data regions
data-region-lazy-memory-allocation	Boolean	Enable lazy memory allocation for Data regions
data-region-	Boolean	Enable Data region CDC

Property Key	Data Type	Description
cdc-enabled		
data-region-page-eviction-mode	String	Configures the data region page eviction mode
data-region-page-replacement-mode	String	Configures the data region page replacement mode
data-region-swap-path	String	Data region swap path

CacheConfiguration

The following properties define the configuration parameters required to start a cache within a grid instance.

Property Key	Data Type	Description
disk-page-compression-level	Integer	The algorithm-specific disk page compression level.
egaer-ttl	Boolean	Sets the eager TTL flag.
encryption-enable	Boolean	Sets an encrypted flag.
event-disable	Boolean	Sets the events disabled flag.
mangement-enable	Boolean	Enables management
max-conc-async-oper	Integer	The maximum number of concurrent asynchronous operations.

Property Key	Data Type	Description
query-iter-count	Integer	The maximum number of query iterators that can be stored.
onheap-cache-enable	Boolean	Configures on-heap cache for the off-heap based page memory.
query-detail-metrics-size	Integer	The size of queries detail metrics that will be stored in memory for monitoring purposes.
query-parallelism	Integer	Sets query parallelism.
read-from-backup	Boolean	Sets read from the backup flag.
rebalance-mode	String (`ASYNC` /`NONE`/`SYNC`)	Sets cache rebalance mode.
rebalance-order	Integer	Sets cache rebalance order.
sql-inline-size	Integer	Sets maximum inline size for SQL indexes.
sql-onheap-enable	Boolean	Sets whether SQL on-heap cache is enabled.
sql-onheap-size	Integer	Sets maximum SQL on-heap cache.
statistics-enable	Boolean	Enable Statistics.
store-conc-load-threshold	Integer	Sets the concurrent load-all threshold used for cases when keys' values are being loaded from CacheStore in parallel.

TCP Communication SPI

The following properties define the configuration parameters required for the TCP Communication SPI.

Property	Type	Description
tcp-communication-spi-ack-send-threshold	Integer	TCP Communication SPI acknowledgment sends threshold
tcp-communication-spi-max-connections-per-node	Integer	TCP Communication SPI maximum connections per node
tcp-communication-spi-direct-buffer	Boolean	Enable SPI direct buffer
tcp-communication-spi-direct-send-buffer	Boolean	Enable SPI direct send buffer
tcp-communication-spi-filter-reachable-addresses	Boolean	Enable TCP Communication SPI filter reachable addresses
tcp-communication-spi-force-client-to-server-connections	Boolean	Enable force clients to server connections
tcp-communication-spi-idle-connection-timeout	Long	Configure idle connection timeout
tcp-communication-spi-local-address	String	Configure the local address
tcp-communication-spi-max-connection-timeout	Long	Maximum connection timeout
tcp-communication-spi-msg-queue-limit	Integer	Configure message Queue limit
tcp-communication-spi-name	String	Configure the SPI name
tcp-communication-spi-reconnect-count	Integer	Configure SPI reconnect count
tcp-communication-spi-selectors-count	Integer	Configure SPI selectors count
tcp-communication-spi-selector-spins	Long	Configure SPI selector spins

Property	Type	Description
tcp-communication-spi-slowclient-queue-limit	Integer	Configure slow client queue limit
tcp-communication-spi-socket-receive-buffer	Integer	Configure socket receive buffer
tcp-communication-spi-socket-send-buffer	Integer	Configure socket send buffer
tcp-communication-spi-tcp-no-delay	Boolean	Enable TCP no delay
tcp-communication-spi-unacked-msg-buffer-size	Integer	Configure unacknowledged messages buffer size
tcp-communication-spi-use-paired-connections	Boolean	Enable use paired connections

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the TIBCO BusinessEvents® Enterprise Edition Documentation page.

To directly access documentation for this product, double-click the file at the following location:

`TIBCO_HOME/release_notes/TIB_businessevents-enterprise_6.3.1_docinfo.html`

where *TIBCO_HOME* is the top-level directory in which TIBCO products are installed. On Windows, the default *TIBCO_HOME* is `C:\tibco`. On UNIX systems, the default *TIBCO_HOME* is `/opt/tibco`.

Other TIBCO Product Documentation

When working with TIBCO BusinessEvents Enterprise Edition, you may find it useful to read the documentation of the following TIBCO products:

- TIBCO ActiveSpaces®: It is used as the cluster, cache, or store provider for the TIBCO BusinessEvents Enterprise Edition project.
- TIBCO FTL®: It is used as the cluster provider for the TIBCO BusinessEvents Enterprise Edition project.

How to Access Related Third-Party Documentation

When working with TIBCO BusinessEvents® Enterprise Edition, you may find it useful to read the documentation of the following third-party products:

- Apache Ignite
- Apache Kafka
- Confluent Kafka Schema Registry
- TIBCO Messaging - Schema Repository for Apache Kafka
- Apache Pulsar
- GridGain
- Apache Cassandra
- Grafana
- InfluxDB
- OpenTelemetry
- Control Plane
- Apache Maven

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.