TIBCO BusinessEvents® Enterprise Edition

Developer 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

Project Development

You can build a TIBCO BusinessEvents project using core project resources such as channels, events, concepts, rules, and so on.

TIBCO BusinessEvents Studio® is an Eclipse-based UI used to design, build, and maintain TIBCO BusinessEvents projects. It is integrated into the standard Eclipse menus where appropriate, and works with many established Eclipse UI methodologies and plug-ins.

TIBCO BusinessEvents WebStudio is an online component that allows business users to create and manage business rules in a web browser. In it the user defines an executable rule (business rule) based on the rule template and on the rule template view defined by the developer in TIBCO BusinessEvents Studio.

TIBCO BusinessEvents WebStudio is explained in the document *TIBCO BusinessEvents® WebStudio User Guide*.

A TIBCO BusinessEvents Studio project contains resources used to build, test, and view a design-time TIBCO BusinessEvents project.

Before you begin to use *TIBCO BusinessEvents Developer Guide*, gain a basic familiarity with the product by completing the tutorials in *TIBCO BusinessEvents Getting Started*.

TIBCO BusinessEvents Developer Guide provides practical details about using TIBCO BusinessEvents Studio to build a project and *TIBCO BusinessEvents Configuration Guide* provides information on how to configure the project's cluster deployment descriptor.

For in-depth explanations about designing a TIBCO BusinessEvents project and other topics, read *TIBCO BusinessEvents Architect Guide*. References to relevant topics in that guide are provided in *TIBCO BusinessEvents Developer Guide*.

When it is time to configure your application for deployment, deploy, and manage it, refer to *TIBCO BusinessEvents Administration*.

Creating a Project

This section explains the basic procedure for creating a project.

See *TIBCO BusinessEvents Getting Started* for tutorials on building and populating a basic TIBCO BusinessEvents project.

Tip: To import a TIBCO BusinessEvents Studio project into your workspace, select **File > Import > General > Existing projects into Workspace**. Do this for current release projects only.

To import a project from a prior release, see *TIBCO BusinessEvents Installation*.

Procedure

1. Start TIBCO BusinessEvents Studio. In Windows, click **Start > All Programs > TIBCO BusinessEvents Studio**.
2. If prompted, select or create the Eclipse workspace directory where your project files are stored.

   Note: If you check the option to use this workspace as a default, you are not prompted again.
3. In the TIBCO BusinessEvents Studio UI, select **File > New > Project > TIBCO BusinessEvents > Studio Project** and click **Next**.
4. In the **New Studio Project** field, enter a project name.

   Names follow Java variable naming restrictions. Do not use any reserved words.
5. Accept the default location or clear the **Use default location** checkbox and specify the desired location.
6. Click **Finish**.

Result

A new folder tree appears in BusinessEvents Studio Explorer with a default set of folders that you can use as desired. The `defaultVars` folder, however, is required.

Note: Source Control

If the project is under source control using Perforce, editing a resource automatically checks out the resource and makes it writable.

If the file is not under Perforce but is read-only, then you are prompted to make the file writable when a change is made.

Importing Projects in TIBCO BusinessEvents Studio

You can import projects from earlier versions or the same version using TIBCO BusinessEvents Studio.
A command-line utility to do the same actions is also provided. After importing using the command-line utility, an additional procedure is required before you can work with the project in TIBCO BusinessEvents Studio.

Procedure

1. Select **File > Import**.

   The Import wizard is displayed.

2. In the Import wizard, select **TIBCO BusinessEvents > Existing TIBCO BusinessEvents Studio Project** and click **Next**.

   The Existing TIBCO BusinessEvents Studio Project Import Wizard is displayed.

3. In the **Existing project root directory** field, click **Browse** and select the project root directory of the project you are importing.

4. Select the XPath version to be used for the project in the **XPath Version** dropdown.

5. Do one of the following:

   - Select the **Copy project into workspace** checkbox. This option copies the project into your current workspace.
   - Clear the **Copy project into workspace** checkbox and specify an import location.

6. Do one of the following:
   - If your project has no HTTP channels, click **Finish** (this is the only option).
   - If your project has one or more HTTP channels, click **Next**.
7. If you clicked **Next**, then in the Select Processing Unit window, select the project CDD in the **Cluster Deployment Descriptor** dropdown.
8. Select the processing unit in the **Processing Unit**dropdown that contains the HTTP properties to be migrated.

   An informational panel displays the settings that will be migrated.
9. Click **Finish**.

   The imported project is displayed in the TIBCO BusinessEvents Studio.

   Note: When you import a BusinessEvents project from 6.3.1 pr previous versions, then by default `pom.xml` is generated. If the project `pom.xml` already exists, then it creates a backup copy, `projectname_backup_pom.xml`. Eventually, there will be a single pom.xml for the specified project.

Finding a Project Element

You can find a project element in various ways, and then take some action relating to that element, such as opening its editor or creating a dependency diagram. Depending on the size and complexity of the project, some methods of finding elements might be more convenient than others.

An element can be a part of a resource (such as a concept property):

- Expand the project tree in BusinessEvents Studio Explorer and double-click the element name.
- Select **Navigate > Open Studio Element** and select an element from the alphabetical list of elements. You can search using wildcards.

- Select the Resource perspective, then select **Navigate > Open Resource** (or press Ctrl+Shift+R). A dialog allowing you to search through all resources in your workspace appears. To display the whole list, enter double asterisks (`**`). You can also use the asterisk (`*`) as a wildcard. For example, to display all concept resources, enter `*.concept`.
- Open the Project View diagram and navigate to the element in the diagram. You can search using filters. Double click the element on the diagram to open it.

Exporting (Generating) Concept and Event Schema (XSD) Files

Using the Generate Schema utility, you can export concepts and events to XML Schema Definition (XSD) files, one per entity, in a specified location. You can generate schema for all concepts, all events, or both. You cannot generate schema for one selected entity type. The files use the same folder structure as the project from which they are exported. In addition, `_BaseConcept.xsd` and `_BaseEvent.xsd` are generated in the root of the selected directory.

XML schemas are used for interoperability between TIBCO BusinessEvents and third-party tools or SOA platforms that use well-defined XML for message communication, transformation, and validation.

In the XSD files, concepts are represented as follows:

- Each concept is exported to its own complex type using its own namespace.
- Referenced concepts have a corresponding ref attribute in the parent's complex type.
- Contained concepts have a corresponding type attribute in the parent's complex type.

Generating an XML Schema (XSD File)

Procedure

1. Select the project of the schema that you want to generate.

2. From the **File** menu, select **Export**. In the Export wizard, expand **TIBCO BusinessEvents** and select **Generate Schema**.

   You can also select this utility from the option (right-click) menu anywhere in BusinessEvents Studio Explorer.

3. In the **Schemas Folder** field and select the folder where you want to put the schema files.

4. Select the **Override TIBCO BusinessEvents Namespace** checkbox to specify a different namespace.

   If you do not select a different namespace, an informational message is displayed. Click **Yes** to continue or **No** to return to the dialog and provide a namespace.

   Provide a different namespace to avoid conflicts with the source concepts and events. If you do not provide a namespace, then the default TIBCO BusinessEvents namespace is used. TIBCO BusinessEvents events and concepts have a hidden schema. If the source entities and generated schema files are in the same folders, use of the default TIBCO BusinessEvents namespace results in a namespace conflict. In this case, you must provide a namespace.

5. In the Select Resources panel, select Concepts or Events or both. Schemas for all concepts or all events in the project (or both) are generated accordingly.

6. Click **Finish**.

Result

The XSD files for the selected resources (all concepts, all events, or both) are generated in the subdirectories of the selected directory. Subdirectory names match the project folders. The `_BaseConcept.xsd` and `_BaseEvent.xsd` files are generated in the root of the selected directory.

Validating a Project or Project Resource

When you save a resource, TIBCO BusinessEvents performs validation checks to ensure that all resource requirements are met. For example, it checks that required fields are completed, names are valid, syntax in rules is correct and no unknown functions are called.

You can also perform validation explicitly for an entire project or selected project resource.

Note: XML schemas are validated to see if they actually define conflicting elements before marking them with warnings and ignoring the conflict. If the system property `schema.conflictingNS.ignore` is true in the `studio.ini` file, conflicts in schema elements are not checked, and a warning is not added.

Procedure

1. In BusinessEvents Studio Explorer, do one of the following:
   a. Right-click a project name, folder name, or a project resource name, and select **Validate Project**.
   b. Select a project name, folder name, or a project resource name, and select **Project > Validate Project**.
2. A pop-up window displays the message “Validation was successful”, or it displays summary information about any problems. Details about problems are displayed in the Problems view.
3. Many validation issues can be fixed using the Quick Fix feature. In the Problems view, right-click on a problem and select Quick Fix.

   See also Using the Quick Fix Feature in the Rule Editor for a related use of this feature.
4. In addition to validating for internal consistency, you can run a project analyzer feature. It helps you to understand your project better, and find ways to improve it.

   For example, it identifies rule functions that are not used in any rule, and rules that will never be triggered at run time. See Project Analyzer and Selected Entity Project Diagrams.

Working with External Library and Custom Function Paths

When you work with external libraries or custom functions in your project at design time, and when you run, test, or debug such projects in TIBCO BusinessEvents Studio, ensure that the engine can find all libraries including dependencies on third-party libraries and custom functions, if the project requires any.

For example, if a project uses FTL channels, Java allows the user to set a property `java.library.path`, to ensure that the environment path contains a reference to the directory containing FTL DLLs (or on UNIX the SO files).

Similarly, TIBCO BusinessEvents captures the native library path along with the build path information to pass to engines running inside TIBCO BusinessEvents Studio. You can enter this information as described below, or you can enter it when you are configuring a run configuration or a debug configuration.

Note: To make the Test Connection feature work for JMS Connection and JDBC Shared Connection shared resources, see Enabling the Test Connection Feature.

Adding or Removing Third-Party or Custom Function Libraries

To add an external library or custom function paths, perform the following steps:

Procedure

1. Open the project in TIBCO BusinessEvents Studio.
2. In BusinessEvents Studio Explorer, right-click the project and click **Properties** (or press Alt+Enter). You see the properties dialog for the project.
3. In the left panel, select **Build Path**.
4. To add the custom function or third-party libraries, select the **Java Libraries** tab.
5. Click **Add Library** and select the required JAR file.
6. Click **OK**.
7. Save the resource.

Note:

- `Java Build Path` is not available in the `Build Path` for new BusinessEvents projects from the version 6.3.1 as these are native Maven projects. However, if these projects are converted to Java, then the `Java Build Path` is displayed.
- The older BusinessEvents projects prior to 6.3.1 that are defined as Java projects, when imported in 6.3.1, display in the `Java Build Path`.

Result

Deploytime Action Required

To make custom functions or external libraries available at runtime, complete the following steps on all machines where TIBCO BusinessEvents is installed:

- Either copy the JAR or JAR files to the `lib/ext/tpcl` directory, or other directory in the class path; or update the class path in the TRA file to point to the location of the JAR or JAR files.
- If a JAR has dependencies on native libraries, edit BE_HOME`/bin/be-engine.tra` and depending on the operating system update `PATH LD_LIBRARY_PATH`, `SHLIB_PATH`, and `LIBPATH` as needed.

  For more details, see *TIBCO BusinessEvents Administration*.

To remove the third-party or custom functions libraries, follow the instructions in Adding or Removing Third-Party or Custom Function Libraries, but instead of **Add Library**, select the JAR file you want to remove and click **Remove Library**

Note: When a third-party, custom function, or a project library is added or removed, it is reflected in the project's pom.xml. So, the project pom.xml dependencies are kept in sync with the project's `Build Path`.

Adding or Removing Variables Using Properties Tag in POM File

In businessevents-enterprise version 6.3.1, you can use variables by adding them as custom properties under the `properties` tag in the pom.xml file. For example:

```
<properties>
        <beHome>C:/tibco/be/6.3</beHome>
        <thirdPartyDir>C:\tibco\be\6.3\lib</thirdPartyDir>
        <jdkHome></jdkHome>
        <projectVersion>6.3.1</projectVersion>
 </properties>
```

At runtime, as per the requirement of a project you can override these properties from the command line. For example, to override BusinessEvents installation use `-D` parameter in mvn package phase:

```
mvn package -DbeHome=/my/other/behome
```

This leverages CI/CD seamlessly so that the same pom.xml file with custom properties is used during the build.

Working with Project Libraries

This section explains how to work with project libraries in TIBCO BusinessEvents Studio. You can also work with project libraries at the command line.

Project libraries (design-time libraries) are archives that enable you to create project resources once, and share them with other projects.

Project libraries can contain any resources from a TIBCO BusinessEvents Studio project. For example, a project library might contain some concepts that are standard across projects (such as Person), so that multiple projects do not need to redefine these concepts.

Project libraries can include, or, as desired, can consist only of the global variables.

Project libraries have the file extension `.projlib`. TIBCO BusinessEvents Studio includes the following features to allow the refactoring of project resources into project libraries.

Note: Ensure that the project library does not use elements already in use in the local project. If elements in an imported project library conflict with the local project elements, those in the local project have priority. The project library element is ignored in this case for build purposes and is not included in the EAR file.

If multiple project libraries have the same element, then the first one on the build path wins, and the other ones are ignored.

Creating (Exporting) a Project Library

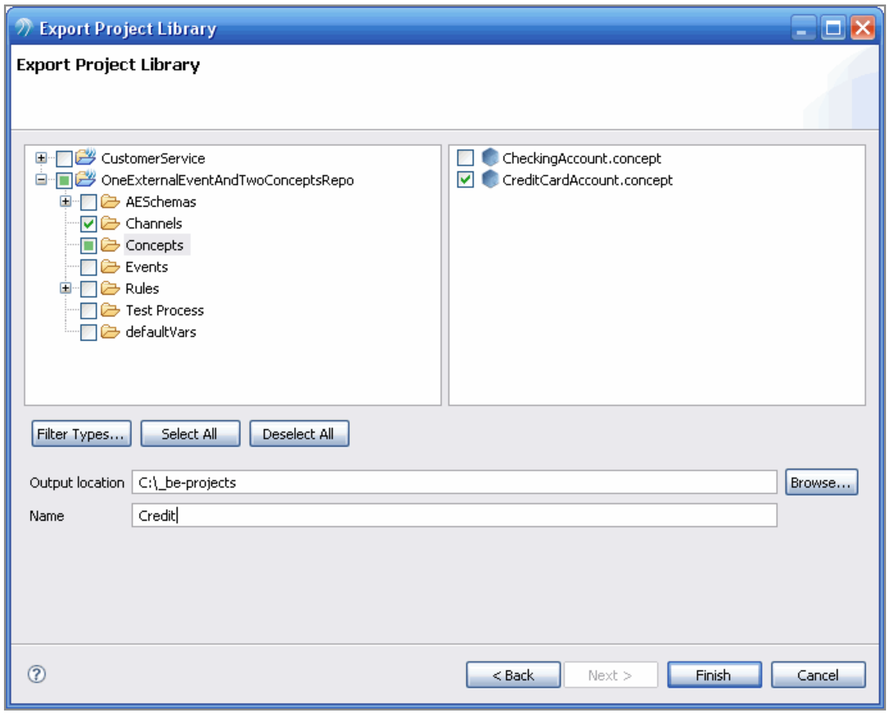
To create a project library, export selected resources to an archive.

Procedure

1. Optionally, select a project or any of its resources in TIBCO BusinessEvents Studio.
2. Do one of the following:
    - From the **File** menu select **Export**.
    - Right-click anywhere in BusinessEvents Studio Explorer and select **Export**.

      In the Select dialog, select **TIBCO BusinessEvents > Project Library** and click **Next**. You see the Export Studio Project Library dialog.

3. In the Export Studio Project Library dialog, select the resources you want to export. If you selected resources in BusinessEvents Studio Explorer they are selected automatically.



Output location: C:_be-projects

Name: Credit

The left panel displays folders. The right panel displays resources. Select what to export in any of the following ways:

- Select a folder to select all resources within it.
- Highlight a folder to display its resources in the right panel. Select individual resources as desired.
- To reduce the set of selected resources, first select resources then click the **Filter Types**. Select one or more resource types, then click **OK**. The Export Studio Project Library dialog selection is now reduced to only resources of the selected type or types.

4. Enter an output location and give the project library a name, then click **Finish**.

Note: To avoid validation issues, include the library you just exported.

Adding (and Removing) Project Libraries in a Project

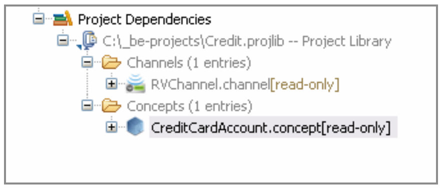
Adding a project library does not physically import the files. Instead a pointer to the files is maintained in the project and the resources appear in BusinessEvents Studio Explorer as if they are in the project. The physical location can be in the project folders or external to them, as long as they are available to the project at design time.

You can add a project library to a project in either of these two ways:

- By including a project library
- By importing a project library

After you add the project library, it appears at the root of the project tree as a Project Dependencies node.

Note: Whenever you add or remove a project library, it is added or removed from the project pom.xml. The dependencies in the project pom.xml are thus kept in sync with the project `Build Path`.


Project Dependencies
C:_be-projects\Credit.projlib -- Project Library
Channels (1 entries)
RVChannel.channel[read-only]
Concepts (1 entries)
CreditCardAccount.concept[read-only]

Including a Project Library (Adding it to the Build Path)

Procedure

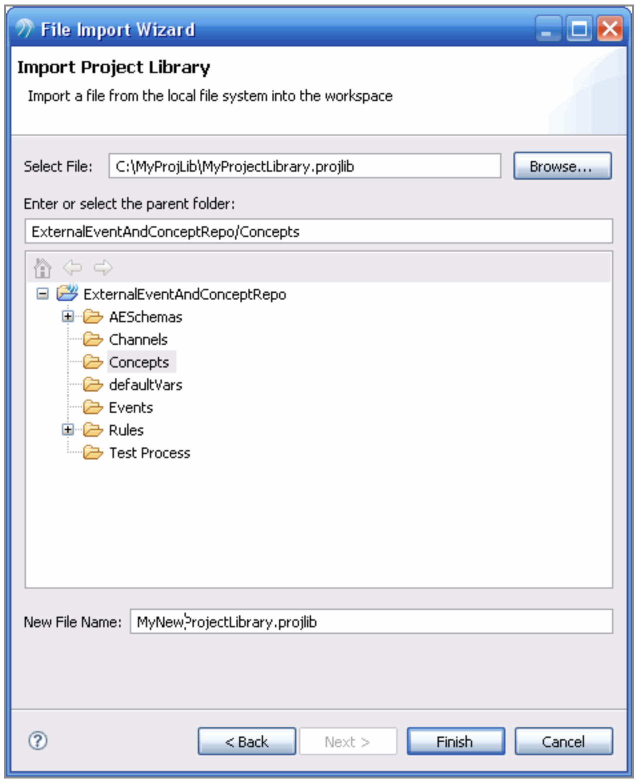
1. Open the project in TIBCO BusinessEvents Studio.
2. In BusinessEvents Studio Explorer, right-click the project name and click **Properties** (or press Alt-Enter, or select **Project > Properties**). You see the properties dialog for the project.
3. In the left panel, select **Build Path** and then select the **Project Libraries** tab.
4. Click **Add Library** and browse to and select the desired project library (`.projlib`) file.
5. Click **OK**. You are prompted to rebuild the project.

6. Save the resource. The project library appears at the root of the project tree as a Project Dependencies node.

Importing a Project Library

Procedure

1. Do one of the following:
   - From the File menu, select **Import**.
   - Right-click anywhere in BusinessEvents Studio Explorer and select **Import**.
2. In the Import wizard, select the dialog, select **TIBCO BusinessEvents > Project Library**, and click **Next**. You see the Import Project Library dialog.


File Import Wizard

Import Project Library

Import a file from the local file system into the workspace

Select File: C:\MyProjLib\MyProjectLibrary.projlib

Enter or select the parent folder:
ExternalEventAndConceptRepo/Concepts

New File Name: MyNewProjectLibrary.projlib

3. Select the **File** field, browse to, and select the project library (`.projlib`) file.
4. Select a project from the project tree or type the name in the field above the project tree area. You can only import at the root of a project.

   Ignore the New File Name field. It is not used by TIBCO BusinessEvents.

5. Click **Finish**. The project library appears at the root of the project tree as a Project Dependencies node.

Removing a Project Library

Procedure

1. Open the project in TIBCO BusinessEvents Studio.
2. In BusinessEvents Studio Explorer, right-click the project name and click **Properties** (or press Alt-Enter, or select **Project > Properties**). You see the properties dialog for the project.
3. In the left panel, select **Build Path** and then select the **Project Libraries** tab.
4. Browse to and select the desired project library (`.projlib`) file.
5. Click **Remove Library.**

   Note: A project library can be removed by deleting the dependency entry from the project pom.xml file.

Changing XPath Version of The Project

If you want to change the default XPath version of the new project to XPath 1.0, or if you want to change the XPath version of an older project to XPath 2.0, you can do that in Studio.

Procedure

1. In BusinessEvents Studio, select the project for which you want to change the XPath version.
2. Right-click the project and select **Properties**, or click **File > Properties**.

   The property window for the project is displayed.
3. Select **Build Path** and select the appropriate XPath version in the **XPath Version** field.

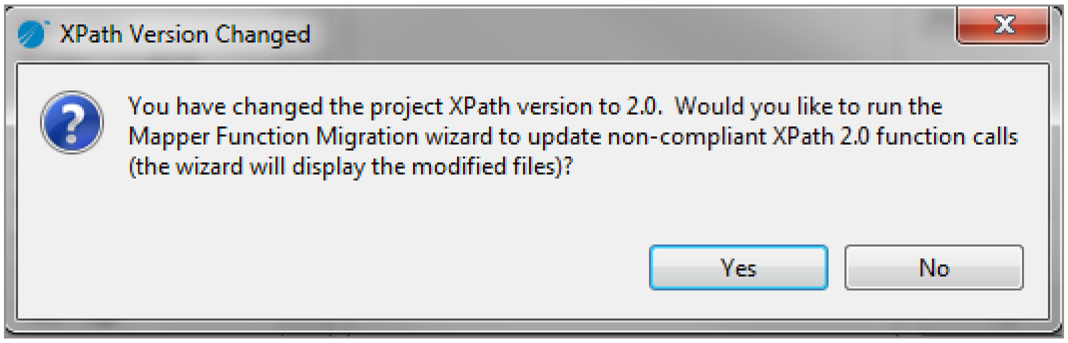
4. Click **OK**

   Alternatively, you can edit the `.beproject` file and update the value of the `xpathVersion` property to 1.0 or 2.0.

   If you have changed the XPath version from 1.0 to 2.0, the XPath Version Changed confirmation dialog is displayed. Click **Yes** to run the Mapper Function Migration wizard to fix common mapping issues. See Migrating Mapper Functions from XPath 1.0 to XPath 2.0 for more details. Click **No** if you wish to fix all the mapping issues manually.

   Important: Always back up your project before running the Mapper Function Migration wizard.

   Figure 1: Mapper Function Migration Wizard Confirmation



XPath Version Changed

You have changed the project XPath version to 2.0. Would you like to run the Mapper Function Migration wizard to update non-compliant XPath 2.0 function calls (the wizard will display the modified files)?

   The project now uses the XSLT mapper based on the new XPath version.

What to do next

After the change in the compatible XPath version, check the XSLT mappings for the validation errors due to the new XPath version. Use the autofix option in the Check and Repair dialog to correct the typecasting and function errors. See Mapping and Transforming Data for more details.

Disabling Caching for Xpath Expressions

While running an XPath expression, the expression string needs to be parsed and compiled. The result of this compiled object is cached so that repeated expressions do not need to be compiled again. By default, caching is enabled.

In some cases, XPath expressions are dynamic in nature and every single one of them does not need to be cached to avoid excessive memory usage. You can disable caching of XPath expressions by setting the `com.tibco.xpath.cache.disable` property to `true` in the `be-engine.tra` file located at *`BE_HOME`*`/bin`.

Working with Global Variables

Global variables provide an easy way to set defaults for use throughout your project. When the project is deployed, all occurrences of the global variable name are replaced with the provided global variable value or a deploy-time override.

Tip:

- The datatype of the global variable must match the datatype accepted in the field where you use it. If the global variable is of a different type, runtime errors result.
- An exception to the above allows flexibility in numeric fields: global variables used in numeric fields can be of any type, as long as the substituted value of the field is numeric.
- A project folder called `defaultVars` is available but not exposed in TIBCO BusinessEvents Studio Explorer, so that you can share the global variables using source control software. It is not used for other purposes.

Setting and Overriding Global Variables from a Project Library

You can export global variables to a project library and import the library into other projects (see Working with Project Libraries).

When multiple global variables have the same name, one overrides the rest and is used in the project. The name, datatype, and default value of the overriding global variable are used. The following is the list of override order:

- If multiple project libraries contain a global variable with the same name, the one in the project library at the top of the **Build Path** list takes precedence. The global variable of the top-listed project library overrides the global variable of all other project libraries listed below it.
  The **Build Path** list is in sync with the pom.xml file. So, the global variable's priority is determined by its project library's position in the runtime pom.xml file.
- If a global variable in a local project shares a name with a global variable in a project library, then the local project's global variable takes priority and overrides the one in the project library.

The Global Variables view presents a merged list. A blue upward pointing arrow in the row for a global variable (⇧) indicates that this global variable overrides another global variable. The source of the global variable is shown in the Project Source column to the right of the arrow.

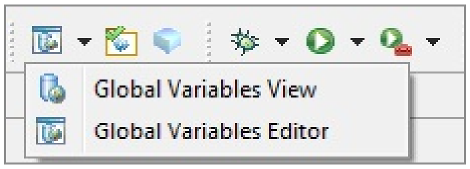
To see all global variables, including overridden ones, open the global variable editor and expand **Project Dependencies** and project libraries. Global variables from project libraries are not editable.

Adding and Managing Global Variables

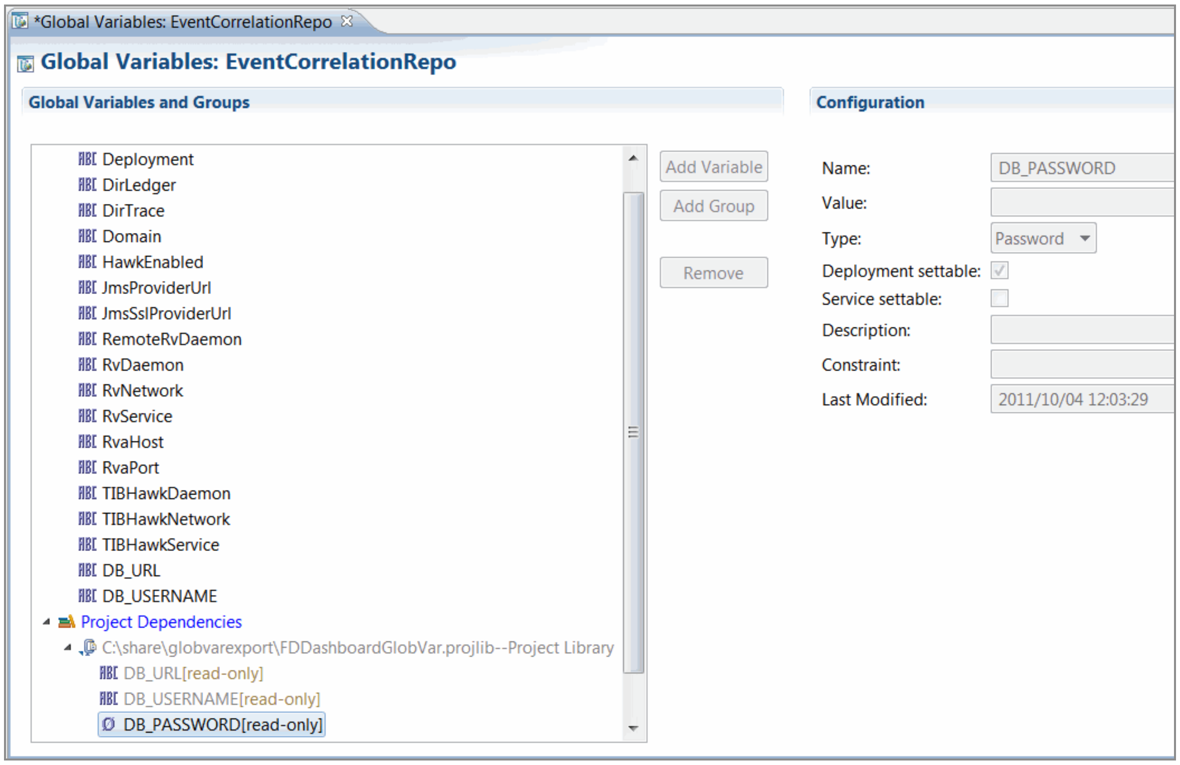
In a new TIBCO BusinessEvents Studio project no global variables are predefined. When you import a project from an earlier version of TIBCO BusinessEvents, however, you see predefined global variables as well as any others defined in the project.

You may also see global variables in project libraries. All global variables are visible in the editor including overridden ones. To see the merged list of global variables that are used in the project, open the Global Variables view in one of the following ways:

- Click the **Global Variables View** button from the toolbar.



- From the top menu, select **Window > Show View > Other > TIBCO BusinessEvents > Global Variables.**


*Global Variables: EventCorrelationRepo
Global Variables: EventCorrelationRepo
Global Variables and Groups
Configuration
Add Variable
Add Group
Remove

Deployment
DirLedger
DirTrace
Domain
HawkEnabled
JmsProviderUrl
JmsSslProviderUrl
RemoteRvDaemon
RvDaemon
RvNetwork
RvService
RvaHost
RvaPort
TIBHawkDaemon
TIBHawkNetwork
TIBHawkService
DB_URL
DB_USERNAME
Project Dependencies
C:\share\globvarexport\FDDashboardGlobVar.projlib--Project Library
DB_URL[read-only]
DB_USERNAME[read-only]
DB_PASSWORD[read-only]

Name:	DB_PASSWORD
Value:	
Type:	Password
Deployment settable:	✓
Service settable:	
Description:	
Constraint:	
Last Modified:	2011/10/04 12:03:29

Add and Manage Global Variables

Groups are used for organizing variables. Variable groups are especially useful if multiple developers share a project using a version control system. When referencing a variable that is in a group, use the complete path, for example `%%mygroup/mysubgroup/myvariable%%`. Since the complete path is used, the name of a variable in a group can be the same as the name of a variable in a different group.

Procedure

1. Open the project in TIBCO BusinessEvents Studio.
2. Open **Global Variables Editor** in one of the following ways:
   - From the toolbar, click the **Global Variables Editor** button.
   - From the top menus, select **Project > Edit Global Variables**.

     You see the global variables editor listing the variables available, if any.

3. Perform any of the following (for details about the Global Variable fields, see Global Variable Reference):
   - To add a variable, click **Add Variable** and complete the fields.
   - To edit a variable, select the variable and update the fields.
   - To add a variable group, click **Add Group**.
   - To add a variable to a group, first select a group, then click **Add Variable**.
   - To remove a variable or a group, highlight it and click **Remove**.

   Note: You must add at least one variable to a group, or the group is not saved. If you delete all global variables in a global variable group, the group itself is also automatically deleted.

4. Save the resource. Groups and references to the `defaultVars.substvar` file appear in the **defaultVars** project folder.

Using Global Variables

To Use Global Variables in TIBCO BusinessEvents Studio Project Fields

To use a global variable as the value for a project setting, drag it from the list of variables into the text box for the setting, or enter it manually. Use the following syntax:

`%%Variable_Group/Variable_Name%%`

You must include the global variable group hierarchy, if one exists.

For example, to use a global variable in a **File Path** field, you might enter the following:

```
%%filePathVars/certificateFilePath%%
```

To Use Global Variables in the Rule Editor

To use a global variable in the rule editor, use one of the `System.getGlobalVariableAs*` functions. For example:

```
System.getGlobalVariableAsString("myvars/Hostname", "Localhost")
```

Where, `myvars/Hostname` is the name of the variable group and variable, and `Localhost` is an optional literal value to use if the variable is not found.

To Use Global Variables in Debugger

To use a global variable in Debugger, add it as a VM argument. Prefix the variable with –V, as shown:

–V *Variable_Group/Variable_Name=*

Global Variable Reference

Global Variable Reference

Field	Description
Name	The variable name. Different groups can have the same-named variable, because the group name is included when you use the global variable.
Value	The variable value. Varies according to type.
Type	Data type of the global variable. The values are: • String • Integer • Boolean • Long • Password The type must match the type of the field where the global variable is used, or errors result.
Deployment Settable	**For deployment with TIBCO Administrator** If selected, the variable is visible and can be set when deploying using TIBCO Administrator. The values set at that time are saved in the project that the TIBCO Administrator creates from the provided EAR file. If the checkbox is not selected, the variable is not visible in the TIBCO Administrator. It is selected by default.
Service	**For deployment with TIBCO BusinessEvents Enterprise Administrator**

Field	Description
Settable	**Agent** You can override global variables in the TIBCO BusinessEvents Enterprise Administrator Agent when the **Service Settable** checkbox is selected. **For deployment with TIBCO Administrator** If both Deployment Settable and Service Settable are selected, the value of the global variable can be set differently for each deployable instance. **Note:** Even if **Service Settable** is selected, the variable is included in the EAR only when the **Include all service level global variables** option is selected when building the enterprise archive file. In all deployment scenarios, values set at the service instance level are passed to the engine at runtime in the format tibco.clientVar.*VariableName*=value. The checkbox is not selected by default.
Description	A helpful description, as needed.
Constraint	Optional. For String and Integer types, using this you can provide a range of allowed values. The constraint field for Strings is an enumeration, for example, `one, two, three`. The constraint field for Integers is for a range, for example, `1-100`. **Note:** Constraints for Integers are currently not implemented in TIBCO Administrator.
Last Modified	Non-editable field that records the date and time this variable was last modified.

Overriding Global Variables at Deploy Time

You can override default values by setting global variable values in one of these ways, depending on how you deploy:

- For starting at the command line:
  - In the design time CDD file.
  - At the command line, using the `--propVar` option, or using the `-p` option to specify a property file where the override properties are defined. See *TIBCO BusinessEvents Administration*.
- For deployment using TIBCO Administrator, set overrides in the TIBCO Administrator UI. You can override variables at the deployment level or at the service level. Values set at the service level are used for the specific engine you are deploying. Service settable global variables are only available if the **Include All Service Level Global Variables** checkbox in the Build Enterprise Archive dialog is selected (see *TIBCO BusinessEvents Administration*).

Order of Precedence of Global Variable Overrides

Global variable values are selected at runtime using values set in the following ways, shown in order of precedence, highest to lowest:

1. Command-line arguments at engine startup (highest priority).
2. Property files specified at command-line engine startup
3. TIBCO Administrator
4. CDD file (ignored by TIBCO Administrator)
5. TIBCO BusinessEvents Studio Global Variable Editor (lowest priority)

Storing Trusted Certificates Outside of Your Project

Trusted certificates are used when you configure SSL, such as in an HTTP Connection Reference, JMS Connection Reference, or JDBC Connection Reference.

Trusted certificates can be used to ensure that remote servers are who they claim to be and to ensure that TIBCO BusinessEvents can identify itself as a valid client when connecting to a server.

You can store the certificates within a project folder, or you can use a special global variable, `BE_GLOBAL_TRUSTED_CA_STORE`, to specify the location of an external directory that contains all the certificates known to TIBCO BusinessEvents.

When you store the certificates within a project folder, then when a certificate changes or expires, you must import any new certificates or certificate chains into the project, rebuild the EAR file, and re-deploy your project.

Using the global variable, however, avoids this problem. When you use the global variable to specify the external location of certificates, then when certificates change or expire, replace certificates or add new certificates and then restart the engine to load the changes.

You can set the global variable value and then use the variable in the usual ways, as described in this chapter. For example, you could use the global variable as follows:

```
tibco.clientVar.BE_GLOBAL_TRUSTED_CA_STORE=file:///somePath/myGTCAFolder
```

Store Trusted Certificates Outside of the Project

Procedure

1. Create a directory where you want to store the trusted certificates. You must copy this directory to each machine where the engines are deployed. Alternatively, the location can be a shared network area accessible by all process engines.
2. Create a global variable named `BE_GLOBAL_TRUSTED_CA_STORE`. See Global Variable Reference for more information.
3. Set the value of `BE_GLOBAL_TRUSTED_CA_STORE` to the location of the trusted certificates folder on your file system. The value must be a file URL, for example `file:///c:/tibco/certs`.

   The location can be the same for all deployed engines (that is, you copied it to the same location on each machine or it is a shared network drive). Alternatively you can change the value of the global variable as needed when you deploy the project
4. Specify a value in the Trusted Certificates field in the SSL Configuration dialog. When the project runs, the value of BE_GLOBAL_TRUSTED_CA_STORE is used, and not the value you specify in the Trusted Certificates field.
5. Save the resource.

Remote Terminal

TIBCO BusinessEvents Studio provides a terminal option that can be used to perform command-line operations (such as, studio-tools) from within BusinessEvents Studio. The terminal also provides shell access to remote systems from within BusinessEvents Studio.

You can open the terminal in BusinessEvents Studio in either of the following ways:

- Press the shortcut keys **Ctrl+Shift+Alt+T**.
- Click **Open a Terminal** icon in the toolbar.
- Select **Window > Show View > Other** from menu to open Show View window. In the Show View window, select **Terminal > Terminal** and click **OK**.

You can refer to the *Eclipse Documentation* for more details about the terminal.

TIBCO BusinessEvents Studio Tools Utility

TIBCO BusinessEvents Studio Tools is a command-line utility with various operations (tools) you can use to automate common procedures. This chapter documents the tools that can be useful while developing applications in TIBCO BusinessEvents Studio. Other tools in this suite are documented in *TIBCO BusinessEvents® Decision Manager User Guide* and TIBCO BusinessEvents Installation

Note: All arguments of a command-line utility must use only ASCII characters.

For full details about deployment of an EAR file, see TIBCO BusinessEvents Administration. The `buildear` operation within the `studio-tools` utility is useful for automation purposes, for example, in testing environments.

By default, the EAR files are built-in memory. The compiler does not use the file system during code generation. Instead, the Studio JVM is used to load all the Java classes and resources into memory until the build process is completed. You can choose to use the file-system based compiler to build EAR files by setting the appropriate options.

Validating a Project from the TIBCO BusinessEvents CLI

Before building a deployable project, validate the project to ensure that all the resource requirements for the project are met.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.

3. On the command prompt, run the following command:

   ```
   studio-tools -core buildEar -p studioProjectDir [-v] [-noBuild]
   ```

   For example:

   ```
   studio-tools -core buildEar -o c:\FD.ear -p
   D:\Workspace\FraudDetection -v -noBuild
   ```

   See Building an EAR File from the CLI for more information on the options used in this command.

Building an EAR File from the CLI

Note: When building an EAR file in memory for a large project, the JVM may run out of `PermGenSpace` and/or heap space. In such cases, edit the *BE-HOME*`/studio/eclipse/studio.ini` and *BE-HOME*`/studio/bin/studio-tools.tra` file to set appropriate values for the JVM settings. By default the heap size is set to `-XX:MaxPermSize=256m`.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.

3. On the command prompt, run the following command with the format (all on one line):

```
studio-tools -core buildEar [-h] [-x] [-lc] [-jc] [-v] [-noBuild]
[-showWarnings] [-o <outputArchiveFile>] -p <projectDir> -
pl<project lib path><path separator><project lib path> -cp<extended
classpath> -cd<temporary code gen/compilation directory>
```

For example:

```
studio-tools -core buildEar -o c:\FD.ear -p
D:\Workspace\FraudDetection
```

The following table provides detailed information about the options.

TIBCO BusinessEvents Studio Tools Options for Building an EAR File

Option	**Description**
`-core buildEar`	Specifies the `buildear` operation for building EAR files.
`-h`	(*Optional*) Displays help.
`-x`	(*Optional*) Overwrites the specified output file if it exists.
`-lc`	(*Optional*) Specifies that the file-based legacy compiler must be used to build the EAR file. By default, the EAR files are built-in memory.
`-jc`	(*Optional*) Includes the JAR files specified in the Java classpath while building the EAR file.
`-o`	(*Optional*) Specifies the filename for the output EAR file. If not specified the EAR file is the same as the final (leaf) directory name in the *`projectDir`* path.
`-p`	Absolute path to the TIBCO BusinessEvents Studio project directory. The EAR file is built using this project.
`-pl`	(*Optional*) Specifies a list of project library file paths to be used,

Option	Description
	separated by a path separator.
`-cp`	(*Optional*) Specifies the extended classpath to be used.
`- v`	(*Optional*) Performs validation checks to ensure that all the resource requirements are met. For example, it checks if the required fields are completed, names are valid, the syntax in the rules is correct and no unknown functions are called. For more information, see Validating a Project or Project Resource
`-showWarnings`	(*Optional*) Displays warnings found during the validation process. **Default:** False
`- noBuild`	(*Optional*) Use this option along with the validate option (`-v`) to validate a project without building an EAR for it.
`-cd`	(*Optional*) Absolute path to a temporary code generation directory. You can use this option while building an EAR for a project when you do not have write permission to the TIBCO BusinessEvents installation on the system.

Importing an Existing Project from the CLI

You can import projects from earlier versions or the same version using the `studio-tools` command-line utility.

After importing, using the command-line utility, an additional procedure is required before you can work with the project in TIBCO BusinessEvents Studio.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.

3. On the command prompt, run the following command:

```
studio-tools -core importExistingProject [-h] -p studioProjDir [-o
targetProjDir] [-c CDDprojectPath] [-u PUNameFromCDD] [-xp 1.0|2.0]
```

For example:

```
studio-tools -core importExistingProject -p C:\FT\SomeProj -o
c:\MyWorkspace\SomeProj -c COM.cdd -u Invproc -xp 2.0
```

Note: When you import a BusinessEvents project from 6.3.1 pr previous versions, then by default `pom.xml` is generated. If the project `pom.xml` already exists, then it creates a backup copy, `projectname_backup_pom.xml`. Eventually, there will be a single pom.xml for the specified project.

Options

The following table describes the command parameters:

importExistingProject Options

Option	Description
`-core importExistingProject`	Specifies the `importExistingProject` operation for importing a TIBCO BusinessEvents Studio project into the workspace.
`-h`	(*Optional*). Displays help.
`-p`	Source project: Specifies the absolute path to the project directory of the TIBCO BusinessEvents Studio project to be imported.
`-o`	(*Optional*). Specifies the absolute path to the target project directory, where the project is imported to. The target project directory is the source project directory name specified as the last element in the path. But if you specify a different directory as the last element in the path, the directory

Option	Description
	is created if it does not exist. The source project directory is imported within the specified target directory. The original project contents are updated if a target project directory is not specified. If the project to be imported is an older TIBCO BusinessEvents project, it is no longer compatible with the older version after the import. If the target location points to an existing project, the import does not proceed and this message displays: `The specified target location already exists and cannot be used.`
`-c`	(*Optional*). The CDD to use for migration actions. Project path of the CDD (path relative to the root directory of the source project).
`-u`	(*Optional*). If specified, then the `-c` option must also be specified. Specifies the name of the PU (within the specified CDD) that contains settings to be migrated. HTTP channel settings from this PU are migrated to all HTTP channel resources in the project.
`-xp`	(*Optional*). The XPath version to be compatible with the project. The values are: • (default) 1.0 • 2.0

What to do next

Open the Imported Project in TIBCO BusinessEvents Studio

Open the Imported Project in TIBCO BusinessEvents Studio

To open a project imported at the command line, you must add it as a new project.

Procedure

1. Start TIBCO BusinessEvents Studio. In Windows, click **Start > All Programs > TIBCO <*YourEnvironment*> > TIBCO BusinessEvents <*version_number*> > TIBCO BusinessEvents Studio**.
2. From the **File** menu, select **New > Project**. You see the **New Project > Select a Wizard** dialog.
3. Select **TIBCO BusinessEvents > Studio Project** and click **Next**.
4. In the **Project Name** field, enter the directory name where the imported project is located. This is used as the project name.
5. If you have imported the project to a directory in your default workspace, skip this step. If the project directory is located outside the default workspace, uncheck the Use default location checkbox and browse to the *parent* directory of the project imported at the command line.
6. Click **Finish**. The project folders appear in the Studio Explorer view.

Working With Project Libraries at the Command Line

See Working with Project Libraries for an introduction to project libraries and procedures for working with them in TIBCO BusinessEvents Studio.

This section explains how to use a command-line utility to complete the following tasks:

- Create a project library (and optionally overwrite any existing library at the same location with the same name). This action creates the `.projlib` file. The corresponding action in TIBCO BusinessEvents Studio is exporting a project library.

- Add an existing project library to a TIBCO BusinessEvents Studio project. The corresponding action in TIBCO BusinessEvents Studio is importing a project library (or adding it to the build path property page).
- Remove an existing project library from a TIBCO BusinessEvents Studio project. The corresponding TIBCO BusinessEvents Studio action is removing the library from the Build Path property page.

Work with Project Libraries at the Command Line

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.
3. On the command prompt, run the following command with the format (all on one line). Arguments for creating a project library are shown separately for clarity.

   To create a project library:

   ```
   studio-tools -core buildLibrary [-h] -p studioProjDir [-a] [-r] [-x] -n
   projectLib [-f resources]
   ```

   To add or remove an existing project directory in a TIBCO BusinessEvents Studio project:

   ```
   studio-tools -core buildLibrary [-h] -p studioProjDir [-a] [-r] [-x] -n
   projectLib [-f resources]
   ```

Examples of Creating a Project Library

Example 1

The following command creates a project library using the specified project's contents:

```
studio-tools -core buildLibrary -p C:\workspace\MyProj -n
C:\test\myproj.projlib
```

Example 2

The following command creates a project library at the location specified, and overwrites any project library already at that location. From the specified project, the project library uses only the resources specified in the -f argument: the `Concepts` folder, the `MyRule.rule` and all dependent resources referred to by these resources.

```
studio-tools -core buildLibrary -p C:\workspace\MyProj -x -n
C:\test\myproj.projlib -f Concepts,Rules\MyRule.rule
```

Examples for Adding and Removing a Project Library

When a project library exists, built either using TIBCO BusinessEvents Studio or using the command-line options shown in Examples of Creating a Project Library, you can use the `buildLibrary` operation to add a project library to or remove a project library from a TIBCO BusinessEvents Studio project.

Example 3

The following command adds the `myproj.projlib` project library to the TIBCO BusinessEvents Studio project at `C:\workspace\MyProj`.

```
studio-tools -core buildLibrary -p C:\workspace\MyProj -a -n
C:\test\myproj.projlib
```

Example 4

The following example removes the `myproj.projlib` project library from the TIBCO BusinessEvents Studio project at `C:\workspace\MyProj`.

```
studio-tools -core buildLibrary -p C:\workspace\MyProj -r -n
C:\test\myproj.projlib
```

The following table provides detailed information about the options.

TIBCO BusinessEvents Studio Tools Options for Working with Project Libraries

Option	**Description**
`-core`	Specifies the `buildLibrary` tool for adding, removing, and creating (overwriting as needed) project libraries.

Option	Description
`buildLibrary`	
`-h`	*Optional*. Displays help.
`-p`	File path to the TIBCO BusinessEvents Studio project. Resources from this project are used to create the project library. This filepath is also used to identify the project to which you want to add a project library.
`[-r | -a]`	Use one of these arguments as needed. `-r`: removes the specified project library from the project specified by `-p`. `-a`: adds the specified project library to the specified project.
`-x`	*Optional*. Overwrites any existing project library at the location specified by -n.
`-n`	The absolute path of the project library to be created or added or removed.
`-f`	*Optional*. A comma-separated list of resources to include in the project library, using a path relative to the `projectDir`. If not specified all resources of the specified project are used.

Studio Tools Commands Reference

You can perform a variety of tasks from the command line using studio tools commands and their options. To use the commands mentioned below, navigate to `BE_HOME/studio/bin/`. Open the `studio-tools` utility. On the command prompt, run the required command.

generateDTClass

You can use the following `generateDTClass` command to generate class files for decision tables:

```
studio-tools -dt generateDTClass [-h]-d Decision Table Path -p Project
Path -o <Output Directory for generated classes> -x -e EAR Path -pl
<project lib path><path separator><project lib path> -cp <Optional
extended classpath>
```

Option	Description
`-dt generateDTClass`	Specifies the `generateDTClass` operation for generating class files for decision tables.
`-h`	(*Optional*) Prints usage help.
`-d`	Specifies the path to the decision table.
`-p`	Specifies the path of the project to which the decision table belongs.
`-o`	(*Optional*) Specifies the output folder for the generated class files.
`-x`	(*Optional*) Overwrites the specified output class file if it exists.
`-e`	Specifies the path to the ear of the project to which the decision table belongs.
`-pl`	(*Optional*) Specifies the list of project library file path separated by a path separator.
`-cp`	(*Optional*) Specifies the extended classpath.
`-lc`	Use this option if you want to use the legacy compiler.

importExcel

You can use the following `importExcel` command to import a Microsoft Excel file to create a decision table:

```
studio-tools -dt importExcel [-h]-studioProjPath <Studio Project Path> -
excelPath <Excel File Path> -dtName -folderPath <Path relative to
project path> -vrfPath <Path relative to project path> -wsPath <Existing
Studio workspace folder>
```

importExcel Options

Option	Description
`-dt importExcel`	Specifies the `importExcel` operation for importing a Microsoft Excel file to create a decision table.
`-h`	(*Optional*) Prints usage help.
`-studioProjPath`	Specifies the path to the TIBCO BusinessEvents Studio project to which you want to import the file.
`-excelPath`	Specifies the path to the Microsoft Excel (`.xls` or `.xlsx`) file that you want to import to TIBCO BusinessEvents Studio.
`-dtName`	Specifies name of the decision table that you want to create using the Microsoft Excel file.
`-folderPath`	Specifies the output folder for the created decision table. Specify the output folder path relative to the path of the TIBCO BusinessEvents Studio project path.
`-vrfPath`	Specifies the path of the virtual rule function that the imported decision table uses. Specify the virtual rule function path relative to the path of the TIBCO BusinessEvents Studio project path and exclude the `.vrf` extension in it.
`-wsPath`	(*Optional*) Use this option if you want to specify the location of the existing TIBCO BusinessEvents Studio workspace folder to which the decision table is added. The best practice is to leave this option blank so that the project is not stale.

convertCSV2Excel

You can use the following `convertCSV2Excel` command to convert a CSV file to a Microsoft Excel file:

```
studio-tools -dt convertCSV2Excel [-h]-csvPath <CSV File Path> -
excelPath <Excel File Path> -s <Column Separator>
```

Option	Description
`-dt convertCSV2Excel`	Specifies the `convertCSV2Excel` operation for converting CSV files to Microsoft Excel files.
`-h`	(*Optional*) Prints usage help.
`-csvPath`	Specifies the absolute path to the source CSV file.
`-excelPath`	Specifies the absolute path to the Microsoft Excel file.
`-s`	(*Optional*) Specifies the output folder for the generated class files.

validateTable

You can use the following `validateTable` command to validate a decision table using the command line:

```
studio-tools -dt validateTable [-h] -studioProjPath <Studio Project
Path> -dtPath <Path relative to project path> -wsPath <Existing Studio
workspace folder>
```

validateTable Options

Option	Description
`-dt validateTable`	Specifies the `validateTable` operation for generating class files for decision tables.
`-h`	(*Optional*) Prints usage help.
`-studioProjPath`	Specifies the path to the TIBCO BusinessEvents Studio project which contains the decision table that you want to validate.

Option	Description
`-dtPath`	Specifies path of the decision table that you want to validate. Specify this path relative to the path of the TIBCO BusinessEvents Studio project path and exclude the `.dt` extension in it.
`-wsPath`	(*Optional*) Use this option if you want to specify the location of the existing TIBCO BusinessEvents Studio workspace folder. The best practice is to leave this option blank so that the project is not stale.

validateWSTable

You can use the following `validateWSTable` command to generate class files for decision tables:

```
studio-tools -dt validateWSTable [-h] -r <Path of SCS> -p <project Name>
-e <EAR archive path> -t <DecisionTable Temp File Path>
```

Option	Description
`-dt validateWSTable`	Specifies the `generateDTClass` operation for generating class files for decision tables.
`-h`	(*Optional*) Prints usage help.
`-r`	Specifies a path to the decision table.
`-p`	Specifies a path of the project to which the decision table belongs.
`-e`	(*Optional*) Specifies the output folder for the generated class files.
`-t`	(*Optional*) Overwrites the specified output class file if it exists.

exportToExcel

You can use the following `exportToExcel` command to export decision tables as Microsoft Excel files(`.xlsx`):

```
 studio-tools -dt exportToExcel [-h] -studioProjPath <Studio Project
Path> -dtPath <Decision Table Path(File/Folder)> -excelPath <Exported
Excel Path)> -useColumnAlias <Export Column Alias> [-legacy]  -e <Ear
path>
```

Option	Description
`-h`	(*Optional*) Prints usage help.
`-studioProjPath`	Specifies the path to the TIBCO BusinessEvents Studio project that contains the decision table that you want to export.
`-dtPath`	Specifies the path of the decision tables that you want to export. Specify this path relative to the path of the TIBCO BusinessEvents Studio project path and exclude the `.dt` extension in it. If you want to export multiple decision tables, specify the path to the folder containing them.
`-excelPath`	Specifies the path of the Microsoft Excel file where you want to export the decision tables.
`-useColumnAlias`	(*Optional*) Use this option if you want to print column alias as column names in the decision table.
`-legacy`	(*Optional*) Use this option to export a decision table in the `.xls` format instead of the default `.xlsx` format.
`-e`	(optional) Specifies the path of the EAR file of the project.

importXSDs

You can use the following `importXSDs` command to generate concept definitions based on XSD schema elements:

```
 studio-tools -core importXSDs [-h] -s <xsdSourceDir> -o <outputDir> -p
<projectName> [-x]
```

Option	Description
`-h`	(*Optional*) Prints usage help.
`-s`	Specifies the path to the source directory that contains the XSD files.
`-o`	Specifies the absolute path to the output directory of the converted concepts and events.
`-p`	(*Optional*) Specifies the target TIBCO BusinessEvents project name for the converted concepts and events. If omitted, the target directory is assumed to be the TIBCO BusinessEvents Studio project, and the project name is derived from the output directory name.
`-x`	(*Optional*) Use this option to overwrite the specified output files if they exist.

Migrating Mapper Functions to XPath 2.0 Using the CLI

After the project is migrated to the XPath 2.0, you can use the `migrateMapperFunctions` operation of the `studio-tools` utility to fix common Mapper Function issues.

This command fixes the typecast errors, if the data type can be converted using the XSD: constructor functions. It also fixes the issues that displays the error: `XSLT is out of sync with schema component properties.`

Important: Always back up your project before running the Mapper Function Migration wizard.

Procedure

1. Navigate to *BE_HOME*`/studio/bin/`.
2. Open the `studio-tools` utility.

3. On the command prompt, run the following command:

```
studio-tools -core migrateMapperFunctions -p <studioProjDir>
```

Options

The following table describes the command parameters:

Studio Tools Options for Migrating Mapper Functions

Option	Description
`-core migrateMapperFunctions`	Specifies the `migrateMapperFunction` command. The command attempts to fix all SLT and XPath mappings in the project to be XPath 2.0 compliant.
`-p`	The file path to the TIBCO BusinessEvents Studio project to be migrated.

Note: You cannot undo the changes after running this command.

For example:

```
studio-tools -core migrateMapperFunctions -p
c:\MyWorkspace\SomeProj
```

Generating All Project Class Files from the Command Line

You can generate all class files in a project at the command line. Although this is a core component, the class files are generally used within the context of TIBCO BusinessEvents Decision Manager, where decision table class files can be separately deployed.

Generate Class Files at the Command Line

Procedure

1. Navigate to *BE_HOME*/studio/bin/.
2. Open the `studio-tools` utility.
3. On the command prompt, run the following command:

   ```
   studio-tools -core generateClass [-h] -p studioProjectDir [-n
   studioProjectName] -o outputPath [-x {true | false}] [-lc] [-pl
   projectLibraryFilePath] [-cp extendedClasspath]
   ```

   For example:

   ```
   studio-tools -core generateClass -p D:\Workspace\FraudDetection -o
   c:\temp -x true -cp c:\tibco\be\<version>\lib\myjar.jar
   ```

   You see a success message if the files were generated successfully.

TIBCO BusinessEvents Studio Tools for Generating Class Files

This table provides detailed information about the options for generating the class files.

TIBCO BusinessEvents Studio Tools for Generating Class Files

Option	Description
`-core generateClass`	Within the `core` category of operations, specifies the `generateClass` operation used to generate a project's class files.
`-h`	(*Optional*). Displays help.
`-p`	Specifies the absolute path to the TIBCO BusinessEvents Studio project directory.
`-n`	(*Optional*). Specifies the name of the TIBCO BusinessEvents Studio project

Option	Description
	whose class files are to be generated. If not specified, the final (leaf) directory name in the path specified for the `-p` option is used as the project name.
`-o`	Specifies the output directory for generated classes. If you do not specify an output directory, the files are placed in a user temporary directory. For example, in Windows files are stored in temporary directory at the following location: `C:\Documents and Settings\User\Local Settings\Temp\BE_1322046141896`
`-x`	(*Optional*). If `true`, overwrites any existing class file with the same name.
`-lc`	(*Optional*). Specifies that the file-based legacy compiler must be used to build the EAR file. By default, the EAR files are built-in memory.
`-pl`	(*Optional*). Specifies a list of project library file paths to be used, separated by a path separator.
`-cp`	(*Optional*). Extended classpath. Use as needed. Provide separate JAR file paths for each JAR file required for project compilation. For example, additional classpath information is needed if the decision table uses custom functions or third-party JAR files. Separate entries by the appropriate path separator. For example, if the separator is semicolon (;) you can add the following: `C:\customjars\custom.jar;C:\customjars\custom2.jar`

Generating Encrypted Passwords

TIBCO BusinessEvents Studio Tools utility provides a command to generate encrypted passwords.

Syntax

```
studio-tools -crypto encrypt [-h] [-i inputText]
```

Definition

The encrypted text is written to STDOUT. The `exit` status of the command indicates whether the generation was a success or a failure.

An `exit` status of **0** indicates success, and **-1** indicates an error.

Options

The following table describes the command parameters:

TIBCO BusinessEvents Studio Tools Options for Generating Encrypted Passwords

Option	Description
`-h`	(*Optional*). Displays help.
`-i`	Input text that needs to be encrypted.

Running the Tester from the Command Line

Running the Tester on the command line allows you to create test data templates, populate the test data files, and run tests.

To run the Tester on the command line, follow these steps:

Procedure

1. Run the following command.

   ```
   -tester generate
   ```

   This generates the test data template files, which are used for concepts, scorecards, and events for a given project directory.

   The test data template files are either in XLS or CSV format, as desired by the user, and are used to enter the test data. They have a project name, element relative path, external ID, column names, and so on.

   Note: Blank columns should be filled with NA.

   To generate a file in a specific format, see Command Line Options .

2. Enter the data manually in the generated template files.

3. Run the following command.

   ```
   -tester import
   ```

   This imports the data entered into test data files.

   `*.concepttestdata`

   and

   `*.eventtestdata`.

   These files are similar to the files saved by the Studio Tester when selecting Create Test Data.

4. Run the command

   ```
   -tester assert
   ```

   This starts an instance of the TIBCO BusinessEvents engine and asserts the test data generated in step 2 to the instance.

   After the assertion, the result data files are saved in an output directory that has to be provided in the user command-line options, and in one of these two formats:

   - **Default Format:**

     It is either the XML format (default) or the HTML format (defined through the command-line option).

   - **Custom Format:**

     Output is based on Types or Operation (such as Created, Modified, and Deleted).

     The default output is generated based on Operation.

     Command Line Options

Result

The `-tester` operation provides for the following commands:

- The Generate Command
- The Import Command
- The Assert Command

The Generate Command

The command `generate` reads concepts, scorecards, and events from a project and creates test data template files in comma-separated format. The output files can be edited by the user to add test data and then can be used as an input for the `import` command to import the test data.

Syntax

```
studio-tools -tester generate [-h] [-x] -p <projectDir [-t < xls | csv]
[-s <separator] [-o <outputDirectory]
```

Options

The following options are used with the command `generate`.

Option	Description
-h	*Optional*. Prints usage help.
-o	*Optional*. Specifies the output directory. If not specified, the output directory is `ProjectDirectory/TestDataTemplates`.
-p	`Required`. Specifies the project path.
-t	*Optional*. Specifies the output file type. Supported values are `xls` and `csv`. If this option is not specified, the output is in CSV format with a comma (**,**) as a column separator.
-s	*Optional*. Use only when the output file type is in CSV format. To specify a column separator, the supported value is a comma `[,]`. If not specified, the default column separator is comma (**,**).
-x	*Optional*. Overwrites the output directory if it exists.

The Import Command

The command `import` reads the CSV data files and converts them to the appropriate test data format (such as `.concepttestdata`, `.eventtestdata`).

Syntax

```
 studio-tools -tester import [-h] [-x] -i <inputFile -p
<projectDirectory[-o <outputDirectory] [-t < xls | xlsx | csv] [-s
<separator]
```

Options

The following options are used with the command `import`:

Option	Description
-i	*Required*. Sets the input path, which is a path to a directory containing test data template files, a single test data template file, or a path to `Listfile` containing paths of files that must be included for test data generation.
-h	*Optional*. Prints usage help.
-o	*Optional*. Specifies the output path.
-p	*Required*. Specifies the project path.
-t	*Optional*. Specifies the output file type. Supported values are `xls`, `xlsx`, and `csv`. If this option is not specified, the output is in CSV format with a comma (**,**) as a column separator.
-s	*Optional*. Used only when the output file type is CSV to specify a column separator. The supported values are `[,]`. If not specified, the default column separator is comma (**,**).
-a	*Optional*. Appends test data to existing test data with the same name (if present).
-x	*Optional*. Removes the `TestData` root from under the output directory. If it exists, it creates a one.

The Assert Command

The command `assert` is used to assert the test data on a running TIBCO BusinessEvents engine instance.

Syntax

```
studio-tools -tester assert [-h] -p <projectDir -c <cddFile -r <earFile
-w <workingDir[-u <processingUnit][-f <outputFormatType][-o
<outputDirectory][-n <filename][-i <input path][-x]
```

Options

The following options are used with the command `assert`:

Option	Description
-c	*Required*. Specifies the CDD file path.
-f	*Optional*. Specifies the output format type. Acceptable values are `ops` and `type`. If this option is not specified, the default output is generated.
-h	*Optional*. Prints usage help.
-i	*Optional*. Specifies the input directory path. If not specified, it will attempt to use the `TestData` directory under the project directory.
-n	*Optional*. Name of the `resultFile`. If not specified, it creates a file with the name `Run-#.resultdata`.
-o	*Optional*. Specifies the output directory. If not specified, the output directory will be `ProjectDirectory/results.`
-p	*Required*. Specifies the project path.
-r	*Required*. Specifies the EAR file path.
-u	*Optional*. Specifies the processing unit name. If not specified, it will use the

Option	Description
	default processing unit.
-w	*Required*. Specifies the working directory path.
-x	*Optional*. Overwrites the output file if it already exists.

Apache Maven Integration

Apache Maven is an open-source tool that helps you to build, test, and deploy TIBCO BusinessEvents applications like other Java projects in your organization. It creates a uniform build system with Apache Maven across the organization. This provides a standardized and efficient way to manage the entire life cycle of a software project. It enhances compatibility with different IDEs, such as Visual Studio Code, Eclipse, and so on. All the required dependencies are managed in the pom.xml file that streamlines the build process and provides consistent build across the environments.

You can use Maven to get the artifacts from different repositories and share them across different locations. If you are already using Apache Maven for other Java projects, then similar commands can be used for TIBCO BusinessEvents projects.

In this version of 6.3.1, all the TIBCO BusinessEvents projects are native Maven projects by default. The TIBCO BusinessEvents version 6.3.1 installer installs the BusinessEvents Maven plug-in as a built-in component. It is not installed manually as done in earlier versions. However, the prerequisite is that Apache Maven should be installed. If Apache Maven is not installed, then during TIBCO BusinessEvents installation, it displays an error message in the installer log file that some of the projects may not work as the BusinessEvents Maven plug-in is undetected. In such a case, the BusinessEvents Maven plug-in is installed explicitly:

- In Windows from `%BE_HOME%/maven/bin/install-be-maven-plugin.bat`.
- In Unix, from `%BE_HOME%/maven/bin/install-be-maven-plugin.sh`.

In this version, the performance has improved significantly due to lack of duplication. All the project build information is consolidated in a single source file called the Maven pom.xml file. A parent pom.xml is built in with the BusinessEvents 6.3.1 installer in BE_HOME. This parent pom.xml inherits all the dependencies in BE_HOME.

BusinessEvents Maven Plug-in Files and Utilities

In this version 6.3.1, TIBCO BusinessEvents provides a built-in TIBCO BusinessEvents Maven plug-in to integrate Apache Maven with TIBCO BusinessEvents. The JAR file for the plug-in and the utilities, which helps in the integration of Apache Maven with TIBCO BusinessEvents, is provided at `*BE_HOME*/maven/bin`. For more information on these files, see the following table.

BusinessEvents Maven Plug-in Files and Utilities

Files	Description
`be-maven-plugin-<version>.jar`	The BusinessEvents Maven plug-in JAR file must be installed to your local Maven (`.m2`) repository to set up the BusinessEvents Maven plug-in. You can use the `install-be-maven-plugin` utility to do that. See Setting Up TIBCO BusinessEvents Maven Plug-in for more information on how to install the BusinessEvents Maven plug-in. **Note:** In the version 6.3.1, the BusinessEvents Maven plug-in JAR file (`be-maven-plugin-<version>.jar`) is not run manually.
`install-be-maven-plugin.bat`	The utility installs the BusinessEvents Maven plug-in JAR file (`be-maven-plugin-<version>.jar`) to the local `.m2` repository in the Windows platform. See Setting Up TIBCO BusinessEvents Maven Plug-in for more information on how to install the BusinessEvents Maven plug-in.
`install-be-maven-plugin.sh`	The utility installs the BusinessEvents Maven plug-in JAR file (`be-maven-plugin-<version>.jar`) to the local `.m2` repository in the Unix platform. See Setting Up TIBCO BusinessEvents Maven Plug-in for more information on how to install the BusinessEvents Maven plug-in.
`install-projlib.bat`	The utility installs the existing project library (`.projlib`) for the BusinessEvents project to the local `.m2` repository in the Windows platform. See Installing The Project Library to Maven Repository for more information on how to install the project library.
`install-projlib.sh`	The utility installs the existing project library (`.projlib`) for the BusinessEvents project to the local `.m2` repository in the Unix platform. See Installing The Project Library to Maven Repository for more information on how to install the project library.
`pom.xml`	The POM file required for setting up the BusinessEvents Maven plug-in. The `install-be-maven-plugin` utility uses this file.

TIBCO BusinessEvents Maven Plug-in Phases

TIBCO BusinessEvents Maven plug-in overrides the Apache Maven phases with respect to TIBCO BusinessEvents. Thus, running the same Maven phases, you can build or deploy BusinessEvents applications. The BusinessEvents Maven plug-in also adds a `buildProjectLib` build lifecycle for building and installing the project library.

The Maven phases refer to the pom.xml file of the BusinessEvents application for the required arguments. See BusinessEvents Application POM File for more details about the BusinessEvents application pom.xml file.

You can call to run a Maven phase in the command prompt using the following syntax:

```
mvn <phase>
```

In a build lifecycle, the phases are run sequentially to complete the build lifecycle. It means that whenever you call a phase to run in the lifecycle, all the previous phases are completed in sequence. Thereafter, the phase that you have called is run. For example, you have called the install phase:

```
mvn install
```

In such cases, first the `compile` phase is run, then the `test` phase, then the `package` phase, and finally the `install` phase is run.

The following tables list (in sequence) all the phases available in the BusinessEvents Maven plug-in.

BusinessEvents Maven Plug-in Phases for Maven `default` Lifecycle

Phases	Description
`compile`	Compiles the BusinessEvents project. The `useLegacy` tag in the pom.xml file specifies whether to use Legacy Id or key-based lookup. The default value is `false` and key-based lookup is enabled for the project. Some of the previous projects that are imported in the version 6.3.1, may have the Legacy Id set to `true`. In such cases, the value in the pom.xml file for `useLegacy` tag is `true`.
`test`	Runs BusinessEvents JUnit tests in the BusinessEvents project under the `JavaSrc`

Phases	Description
	folder. The test-compile phase that runs before the test phase checks whether the project contains test resources existing in the testSource directory `JavaSrc` folder. This directory can be overridden by setting the testSourceDirectory property. It searches for the BusinessEvents JUnit tests with the following file names: • `**/Test*.java` • `**/*Test.java` It compiles the test resources and runs the JUnit test suite. **Note:** Ensure that the EAR file location in the JUnit test suite file is the same as the EAR file location specified in the pom.xml file.
`package`	Compiles the BusinessEvents project and builds the EAR file at the location specified in the pom.xml file. The `<earLocation>` tag in the pom.xml file for the BusinessEvents application specifies the location to store the generated EAR file.
`install`	Installs the BusinessEvents project's EAR file and pom.xml file into your Maven repository.
`deploy`	Deploys the BusinessEvents project EAR and the project library into the remote repository. To deploy the project library into a remote repository, you must first build the project library at the location specified by the `<projLibLocation>` tag in the pom.xml file. Specify the remote repository details in the project's pom.xml file and create *`$USER_HOME`*`/.m2/settings.xml` a file for repository authorization.

BusinessEvents Maven Plug-in Phases for `buildProjectLib` *Lifecycle*

Phases	Description
`build-project-lib`	Builds the project library (`.projlib`) for the BusinessEvents project. The `<projLibLocation>` tag in the pom.xml file, for the BusinessEvents

Phases	Description
	application, specifies the location to store the generated project library (`.projlib`) file.
`install-project-lib`	Installs the project library (`.projlib`) for the BusinessEvents project to your Maven repository.

BusinessEvents Maven Plug-in Phases for Hot Deploy in TIBCO Enterprise Administrator

Phases	Description

`hot-deploy`	Deploys the generated EAR file for the BusinessEvents application deployed on TIBCO Enterprise Administration Agent. All the earlier phases until the `install` phases are run and the generated EAR file is deployed on the TIBCO Enterprise Administrator server specified in the `pom.xml` file.
	
	`<teaConfig>`
	`        <teaUrl>http://localhost:8777</teaUrl>`
	`        <applicationName>FraudDetection</applicationName>`
	`        <username>admin</username>`
	`        <password>admin</password>`
	`</teaConfig>`
	
	You can skip the earlier phases by using the property `disablePhases`. You can add this property in the project `pom.xml` file inside the tag `beProjectDetails`. The value must be a comma-separated list of all the phases needed to be skipped. The phase that should be skipped must have `be` as a prefix.
	
	Example:
	
	`<disablePhases>bebuildear</disablePhases>`
	
	Note:
	• The TIBCO Enterprise Administrator server and TIBCO Enterprise Administration Agent must be running and the respective BusinessEvents application must be deployed on it.
	• The **Hot Deploy** option in CDD must be selected for hot deployment to work.

BusinessEvents Maven Plug-in Phases for Docker

Phases	Description
`build-app-image`	Builds the Docker image for the TIBCO BusinessEvents application. You can specify the following arguments that are required for the Docker image generation in the `pom.xml` file.

```
<appImageConfig>
       <scriptPath/>
       <docker_dir/>
       <image_type/>
       <app_location/>
       <source/>
       <config_provider/>
        <enable_openjdk/>
       <enable_optimize/>
       <build_tool/>
       <modules/>
        <help/>
       <disable_tests/>
       <be_home/>
       <tag/>
       <dockerfile/>
</appImageConfig>
```

Following are the descriptions for the tags used:

Tag	Description
scriptPath	Path to the script to be run.
docker_dir	Path to the directory where the Docker scripts are located.
image_type	Type of the image to build (`"app"|"rms"|"teagent"|"s2ibuilder"`).
app_location	Path to the TIBCO BusinessEvents application where the

Phases	Description

Tag	Description
	project CDD, EAR, and optional supporting JAR files are located. This is a required tag if the `image-type` is app.
source	Path to *BE_HOME* or TIBCO installers.
config_provider	*(Optional)* Name of the config provider to be included in the image ("gvconsul"|"gvhttp"|"gvcyberark"|"cmcncf"|"custom").
enable_openjdk	*(Optional)* Specifies to use OpenJDK instead of the bundled Oracle JDK. Place the OpenJDK installer archive in the same folder on the file system where TIBCO installers are placed.
enable_optimize	*(Optional)* Optimizes the container image creation. If available, it automatically retrieves the required modules from the project CDD or EAR file.
build_tool	Build the container image using build tools ("docker"|"buildah"). The default is "docker".
modules	*(Optional)* Accepts additional module names as a comma-separated string. For example, "HTTP, kafka".
help	*(Optional)* Print the usage of the script.
disable_tests	*(Optional)* Disables docker unit tests on created image (applicable only for "app" and "s2ibuilder" image types).
tag	*(Optional)* Name and optionally a tag in the 'name:tag'

Phases	Description
	<table><tr><th>Tag</th><th>Description</th></tr><tr><td></td><td>format.</td></tr><tr><td>dockerfile</td><td>*(Optional)* Dockerfile to be used for generating the image.</td></tr></table> You can build the Docker image either by using an existing TIBCO BusinessEvents installation (*BE_HOME*) from your computer or using the software installer of TIBCO BusinessEvents and other required products. The Docker image generated by using software installers is of smaller size in comparison to the Docker image generated using your TIBCO BusinessEvents installation. • Docker image by using software installers - Set <useBEHome> to false and provide the location of the TIBCO BusinessEvents, TIBCO ActiveSpaces, and other add-ons installers in <installers_location>. • Docker image by using an existing TIBCO BusinessEvents installation - Set <useBEHome> to true and provide a BE_HOME path in <be_home>. See "Building BusinessEvents Application Docker Image" section in the *TIBCO BusinessEvents Cloud Deployment Guide* for more details about the arguments required for Docker. **Note:** In the macOS platform, ensure to provide the system `PATH` as an environment variable in the Run configuration. In TIBCO BusinessEvents Studio, right-click the project name and select **Run As > Run Configurations > <*Target_run_configuration*>**. In the **Environment** tab, add a variable `PATH` and provide its value the same as $PATH environment variable value from the system.
`docker-install`	Runs all the earlier phases for the Dockerfile and installs the BusinessEvents project's EAR file and pom.xml file into your Maven (`.m2`) repository.
`docker-deploy`	Creates a tag for the generated application image and deploys the tagged image to the specified Docker registry. All the previous phases are run and then the `docker-deploy` phase is run.

Phases	Description

	Note: For deploying to Amazon Web Services (AWS), you must have the AWS Command Line Interface (CLI) installed and configured on the system.
	
	You can specify the following arguments that are required for deploying the application Docker image in the `pom.xml` file. Specify the repository name and URL required for the deployment.
	
	`<dockerRegistryConfig>`
	`       <repository>Repository Name</repository>`
	`       <Url>Repository Url</Url>`
	`        <tagImage>RepositoryName:ImageName</tagImage>`
	`       </dockerRegistryConfig>`

Setting Up TIBCO BusinessEvents Maven Plug-in

TIBCO BusinessEvents Maven plug-in is packaged with TIBCO BusinessEvents installation. From version 6.3.1 onwards, manual installation of the TIBCO BusinessEvents Maven plug-in is not required. It is a built-in component packaged with TIBCO BusinessEvents installer. The only pre-condition is that Apache Maven should be already installed. If Apache Maven is not available, then the installation does not fail. However, it displays an error message in the installer log that some of the projects may not work as the BusinessEvents Maven plug-in is undetected.

In such situations to build TIBCO BusinessEvents Maven projects, install Apache Maven and then you can setup TIBCO BusinessEvents Maven plug-in manually.

To run the Maven commands, you must install the TIBCO BusinessEvents Maven plug-in to your Maven (`.m2`) repository. This is a one-time activity.

Before you begin

- JRE 1.8 or later (containing `tools.jar`) is installed and its location is added to the `PATH` environment variable. This value could be a path of the JRE provided with the TIBCO BusinessEvents installation or an external JDK 1.8 location. Ensure that the `tools.jar` file is present in the JDK or JRE path that you have provided in the environment variable.
- Apache Maven is installed and its location is added to the `PATH` environment variable
- TIBCO BusinessEvents 5.4.1 or later is installed.
- A working internet connection for downloading Maven dependencies.

Procedure

1. Navigate to the Maven folder in the TIBCO BusinessEvents installation at *BE_HOME*`/maven/bin`.
2. Run the following command:
   - For Windows:

     ```
     install-be-maven-plugin.bat
     ```

   - For Unix:

     ```
     install-be-maven-plugin.sh
     ```

   On the successful run of the command, the BusinessEvents Maven plug-in JAR file (*BE_HOME*`/maven/bin/be-maven-plugin-version-number.jar`) is installed to your Maven (`.m2`) repository.

Generating POM File Using BusinessEvents Studio Tools Utility

TIBCO BusinessEvents provides the `convertToMaven` utility that converts a BE project to a native Maven project, generating the pom.xml file in the process. This pom.xml file is saved at the BusinessEvents project root level.

You can generate the pom.xml file from the studio-tools utility in the following situations:

- To convert an existing BusinessEvents non-Maven project from 6.3.1 or older versions to a Maven project.
- To re-generate pom.xml if the existing pom.xml is accidentally deleted or corrupted.
- To generate the new structured pom.xml for 6.3.1 or older versions of BusinessEvents Maven projects that have the old structured pom.xml.

Note:
- If the pom.xml file is deleted, then the BusinessEvents Project Nature and Builder are removed and the corresponding BusinessEvents features are disabled for the project.
- The project effectively removes the BusinessEvents functionality from the project. However, to add the BusinessEvents Studio nature, see Adding Studio Nature to the Project.

Before you begin

TIBCO BusinessEvents Maven plug-in is set up in your repository. See Setting Up TIBCO BusinessEvents Maven Plug-in for more details.

The following procedure explains the steps to generate pom.xml using the `convertToMaven` utility.

Procedure

1. In the command line, navigate to the studio tools `bin` folder in the TIBCO BusinessEvents installation at BE_HOME/studio/bin.

2. Run the following command:

```
- studio-tools -core convertToMaven -p <studioProjDir> [-g
<groupId>] [-m] [-r] [-x]
```

The `convertToMaven` Utility Parameters

Options	Description
`-p`	Absolute path to the TIBCO BusinessEvents Studio project directory to be converted, or the root directory of all Studio projects if -r is specified. The pom.xml file is generated for this project.
`-g`	*Optional*. A unique group ID for the converted Maven project amongst an organization. If not specified, the default group ID `com.tibco.be` is used.
`-m`	*Optional* The option allows to create a multi-module project. Specifying this tag converts all Studio projects found in the provided directory and creates a multi-module Maven project.
`-r`	*Optional*. It processes all Studio projects recursively in the provided path.
`-x`	*Optional*. Overwrites any existing pom.xml files.

For example, to generate a pom.xml file for the FraudDetection example, run the following command:

```
- studio-tools -core convertToMaven -p BE_
HOME/examples/standard/FraudDetection/FraudDetection -g com.tibco.be
```

Result

A `pom.xml` file is generated for the FraudDetection project at the `BE_HOME/examples/standard/FraudDetection/FraudDetection` location.

Adding Studio Nature to the Project

If a pom.xml is deleted, you can re-generate it for the project. When you select a Studio project nature, you effectively add the TIBCO BusinessEvents functionalities to the project.

The BusinessEvents Studio nature is added to the project by performing the following steps:

Procedure

1. Select the project and navigate to **Project Properties > Project Nature**.
2. To add the Studio project nature to the project, select **Studio Project Nature**.
3. Click **Apply and Close**.

Note:
- Select **Maven Nature** to add Maven nature to the project.
- Add the dependencies in pom.xml, if required, after you add **Studio Project Nature** to the project.

Generating the BusinessEvents Project POM File Using BusinessEvents Studio

TIBCO BusinessEvents® Studio provides the option to generate the pom.xml file for a BusinessEvents project. The generated pom.xml file is saved at the BusinessEvents project's location.

Before you begin

TIBCO BusinessEvents Maven plug-in is set up in your repository. See Setting Up TIBCO BusinessEvents Maven Plug-in for more details.

Procedure

1. In the BusinessEvents studio, open the BusinessEvents project.
2. In the Project Explorer pane, perform either of the following actions:
   - Right-click on the project name and select **Generate Maven POM File**.
   - Select the project and click from the menu - **Project > Generate Maven POM File**.

   The Generate POM for Application window is displayed.

3. In the Generate POM for Application window, enter the details required for the project pom.xml file and click **Apply** to save the pom.xml configuration.

Field	Description
Group ID	A unique group ID amongst an organization. For example, all core Maven artifacts are under the group ID org.apache.maven. This is added in the `pom.xml` file with the `<groupId>` tag. For example, `<groupId>com.tibco.be</groupId>`
Artifact ID	A unique artifact ID that identifies the project or artifact in the group. This is added in the `pom.xml` file with the `<artifactId>` tag. For example, `<artifactId>maven-core</artifactId>`
Version	Version number of the artifact in the group. It is the version of the project name. This is added in the `pom.xml` file with the `<version>` tag. For example, `<version>6.3.1</version>`

4. Click **OK** to generate the pom.xml file for the BusinessEvents project.

Result

The `pom.xml` is created under the BusinessEvents project. Open the `pom.xml` file in BusinessEvents studio or any other XML editor to enter further details such as, properties in the pom.xml file (if required).
Refer to the Apache Maven documentation at https://maven.apache.org for more detailed information on pom.xml.

What to do next

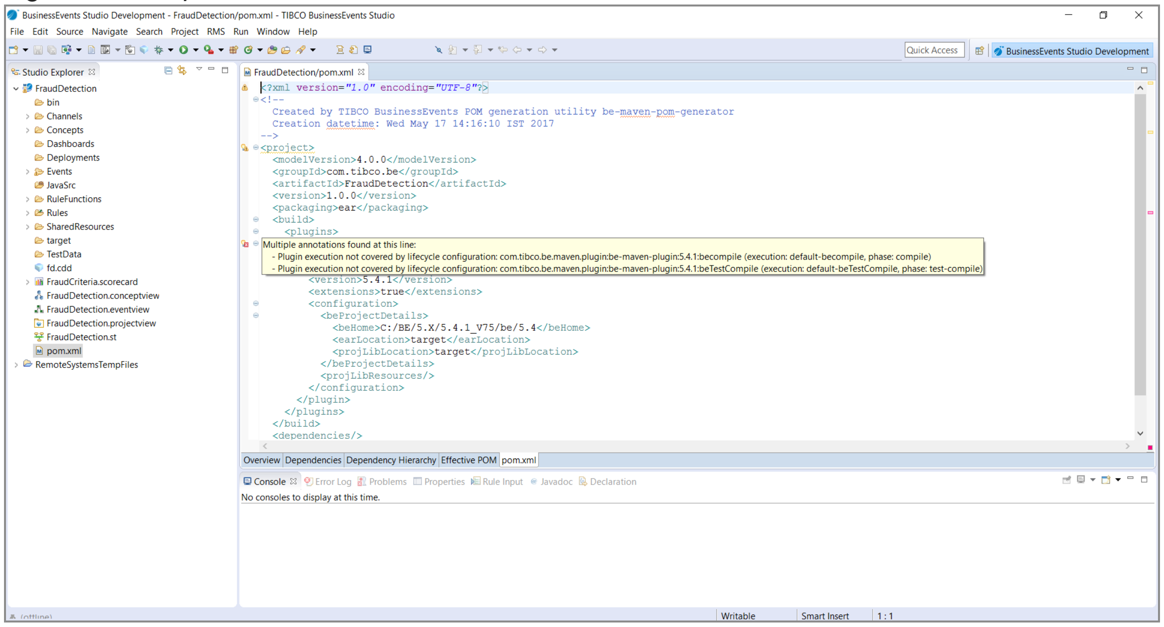
Open the generated pom.xml file in the studio, if you see the `Plugin execution not covered by lifecycle configuration` errors in the **Problems** view, you can use the *Quick Fix* feature to resolve them. The figure Generating the BusinessEvents Project POM File

Using BusinessEvents Studio displays a sample error that you might encounter when you open the pom.xml file for the first time in TIBCO BusinessEvents Studio.

In the **Problems** view, right-click on the error and select **Quick Fix**.

In the Quick Fix wizard, select the problems to fix and select one of the suggested fixes, and click **Finish**.

Figure 2: Sample Error for the POM File in BusinessEvents Studio



```
<?xml version="1.0" encoding="UTF-8"?>
<!--
  Created by TIBCO BusinessEvents POM generation utility be-maven-pom-generator
  Creation datetime: Wed May 17 14:16:10 IST 2017
-->
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.tibco.be</groupId>
  <artifactId>FraudDetection</artifactId>
  <version>1.0.0</version>
  <packaging>ear</packaging>
  <build>
    <plugins>
Multiple annotations found at this line:
 - Plugin execution not covered by lifecycle configuration: com.tibco.be.maven.plugin:be-maven-plugin:5.4.1:becompile (execution: default-becompile, phase: compile)
 - Plugin execution not covered by lifecycle configuration: com.tibco.be.maven.plugin:be-maven-plugin:5.4.1:beTestCompile (execution: default-beTestCompile, phase: test-compile)
        <version>5.4.1</version>
        <extensions>true</extensions>
        <configuration>
          <beProjectDetails>
            <beHome>C:/BE/5.X/5.4.1_V75/be/5.4</beHome>
            <earLocation>target</earLocation>
            <projLibLocation>target</projLibLocation>
          </beProjectDetails>
          <projLibResources/>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies/>
```

No consoles to display at this time.

BusinessEvents Application POM File

In this version of 6.3.1, the performance has improved significantly due to lack of duplication. All the project build information is consolidated in a single source file called the Maven `pom.xml` file. A parent pom.xml is built-in with the BusinessEvents 6.3.1 installer in BE_HOME. This parent pom.xml inherits all the dependencies in `BE_HOME/lib`. Hence, every BusinessEvents project inherits the dependencies in BE_HOME.

Note: Additionally, you can add Maven dependencies from the parent pom.xml, which is at BE_HOME.

The parent pom.xml specifies its parent as the following:

```
<parent>
       <groupId>com.tibco.cep.be</groupId>
       <artifactId>runtime</artifactId>
       <version>6.3.1</version>
</parent>
```

In runtime pom.xml, the project property BE_HOME is set in the `Properties` section to resolve the file system paths. BE_HOME setup is not specific to any project and can be overridden by passing it as part of the mvn command. In this version of 6.3.1, the runtime pom.xml has all the build information with empty values. The project-specific pom.xml overrides the empty values with the properties relevant for a specific project. For example, if there is no Docker configuration for a BE project, then the Docker registry configuration information is empty.

Therefore, in a CI/CD, the projects are built without changing the individual pom.xml.
You can edit the generated pom.xml file with the required parameters for building and deploying the BusinessEvents application.

Third-Party JAR File Dependencies

During the BusinessEvents application pom.xml file generation, the third-party Java libraries found in the project are placed as standard Maven dependencies in the pom.xml file. One dependency entry is created for each JAR file found.

For example, if `JUnit.jar` is found in the project's build path, the following dependency is added to the pom.xml file:

```
<dependency>
  <groupId>com.tibco.be</groupId>
  <artifactId>junit</artifactId>
  <version>0.0.1</version>
</dependency>
```

You can edit these dependencies depending on whether the JAR file is a standard JAR file or not. In the case of the standard JAR file, you can replace the dependency with a standard dependency or continue to use the generated dependency. For example:

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
</dependency>
```

In the case of a custom JAR file, in addition to the dependency entry in the pom.xml file, install it to the local Maven (`.m2`) repository using the standard Maven command. For example, for a custom JAR file `myprojectjar.jar` calls the `install` phase and ensure that the `groupId`, `artifactId` and `version` specified in the `install` command are the same as the dependency in the pom.xml file or conversely.

```
mvn install:install-file -Dfile=/path/to/myprojectjar.jar -
DgroupId=com.tibco.be -DartifactId=myprojectjar -Dversion=0.0.1 -
Dpackaging=jar
```

In case the third-party Java library is not included in the project's build path, you manually add the dependency details (the `<dependency>` tag) for the JAR file in the pom.xml file. After adding the dependency details for the JAR file in the pom.xml file, you must install the JAR file to the local `.m2` repository by using the Maven `install` command. For example, for the JUnit test suit to work in the Maven `test` phase, install the `junit.jar` and the latest `org.hamcrest.core` JAR files to the local `.m2` repository.

JDK Installation Location

If needed, you can also use your own JDK (or JRE containing `tools.jar`) installation for Maven phase execution. By default, Maven uses the JRE installation path added to the system `PATH` environment variable. You can override that path using the `pom.xml` file. In the `pom.xml` file, specify the JDK (or JRE containing `tools.jar`) path in the `<jdkhome>` element. The system then uses the `tools.jar` file at the `<jdkhome>` location for running Maven phases.

Project Libraries Dependencies

During the BusinessEvents application pom.xml file generation, the project libraries (`.projlib`) found in the BusinessEvents project build path are placed as standard Maven dependencies in the pom.xml file. One dependency entry is created for each project library file found. The `<type>` tag specifies that this dependency is a project library.

For example:

```
<dependency>
  <groupId>com.tibco.be</groupId>
  <artifactId>myprojlib</artifactId>
  <version>0.0.1</version>
  <type>projlib</type>
</dependency>
```

In addition to the dependency addition of the project library in the pom.xml file, install the project library in your Maven repository. You can use the `*BE_HOME*/maven/bin/install-projlib` utility to install the existing project library for the BusinessEvents project to your repository. See Installing The Project Library to Maven Repository for more details.

EAR File Location

To run the `compile` phase, specify the location to store the generated EAR file for the BusinessEvents project using the `<earLocation>` tag. The specified location is either a relative or an absolute path on the file system where the generated EAR file is placed. If the path starts with a forward slash (/) or a Windows drive (on the Windows platform), it is considered to be an absolute path. Otherwise, the path is relative to the folder containing the pom.xml file. In 6.3.1, the build process is independent of studio tools. This enables parallel execution of multiple projects, boosting the performance.

Note: Ensure that you provide only the location in the `<earLocation>` tag and not the custom EAR file name. The plug-in uses the combination of `artifactId` and `version` for the EAR file name.

Remote Deployment Repository

To run the `deploy` phase, specify the remote deployment repository location in the pom.xml file using the `<distributionManagement>` and `<repository>` tags.

For example,

```
 <distributionManagement>
          <repository>
                   <id>central</id>
                   <name>be-releases</name>
                   <url>http://artifacts.tibco.com:8081/artifactory/be-
```

```
releases</url>
         </repository>
</distributionManagement>
```

Also, create the *$USER_HOME*/.m2/settings.xml file for repository location and authorization. The content of a sample settings.xml file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
http://maven.apache.org/xsd/settings-1.1.0.xsd"
xmlns="http://maven.apache.org/SETTINGS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <servers>
       <server>
         <username>username</username>
         <password>password</password>
         <id>central</id>
       </server>
    </servers>
    <profiles>
       <profile>
          <repositories>
             <repository>
                <id>central</id>
                <name>be-releases</name>
                <url>http://artifacts.tibco.com:8081/artifactory/be-
releases</url>
            </repository>
        </repositories>
        <id>artifactory</id>
      </profile>
    </profiles>
    <activeProfiles>
      <activeProfile>artifactory</activeProfile>
    </activeProfiles>
</settings>
```

Project Library Build Location

To run the build-project-lib phase, specify the location to store the generated project library (.projlib) file for the BusinessEvents project using the <projLibLocation> tag in the pom.xml file. The specified location is either a relative or an absolute path on the file system where the generated project library (.projlib) file is placed. If the path starts with a forward slash (/) or a Windows drive (on the Windows platform), it is considered to be an absolute path. If not, the path is relative to the folder containing the pom.xml file.

Project Library Build Resources

If you require to include only certain project resources in the project library (`.projlib`) of your BusinessEvents project, specify those resources under the `<projLibResources>` tag. Thus, when the `build-project-lib` phase is run, only the project resources included in the `<projLibResources>` tag are picked building the project library. Add one entry for each project resource that needs to be included in the project library. For example:

```
<projLibResources>
  <projLibResource>Concepts/Customer.concept</projLibResource>
  <projLibResource>Rules/getCustomer.rule</projLibResource>
</projLibResources>
```

Installing The Project Library to Maven Repository

If your BusinessEvents project contains the project library, then you can use the `install-projlib` utility to install the project library in your repository.
The pom.xml file generator utility automatically adds an entry for each of the project library, found in the BusinessEvents application, in the generated `pom.xml` file as a dependency. To resolve this dependency, you need to install the project library (`.projlib`) to your repository. The parameters (`groupId`, `atifactId`, and `version`) for this utility must match the dependency parameters specified in the pom.xml file or conversely.

Internally, the utility uses the `mvn install` command to install the project library to your repository.

Before you begin

TIBCO BusinessEvents Maven plug-in is set up in your repository. See Setting Up TIBCO BusinessEvents Maven Plug-in for more details.

Procedure

1. In the command line, navigate to the Maven folder in the TIBCO BusinessEvents installation at `BE_HOME/maven/bin`.
2. Run the following command:

```
install-projlib -p <projlib> -g <groupid> -a <artifactid> -v <version>
```

The `install-projlib` Utility Parameters

Options	Description
`-p`	Absolute path to the project library (`.projlib`) file for the BusinessEvents project.
`-g`	A unique group ID amongst an organization. For example, all core Maven artifacts do live under the group ID org.apache.maven. This is added in the pom.xml file with the `<groupId>` tag. For example, `<groupId>com.tibco.be</groupId>`
`-a`	A unique artifact ID that identifies the project or artifact in the group. This is added in the pom.xml file with the `<artifactId>` tag. For example, `<artifactId>myprojlib</artifactId>`
`-v`	Version number of the artifact in the group. This is added in the pom.xml file with the `<version>` tag. For example, `<version>0.0.1</version>`

Using OpenJDK with TIBCO BusinessEvents

Based on your requirements, you can choose to use OpenJDK with TIBCO BusinessEvents artifacts instead of using the bundled Oracle JDK.

TIBCO BusinessEvents supports the Amazon Corretto and Oracle distributions of OpenJDK for the platforms, Windows, Linux, and macOS.

Before you begin

Download the OpenJDK installation archive and extract the contents to the folder where you want to install OpenJDK. Refer to this path as `OPEN_JDK_HOME`.

Procedure

1. Update the following variables in the `studio.ini` file:

studio.ini file for Windows

Existing variable and value	Change to
-vm TIBCO_HOME/tibcojre64/*version*/bin/server	-vm *OPEN_JDK_HOME*/bin/server **Example:** -vm `C:/openjdk/java/jdk-11.0.2/bin/server`
-DJDK_LIB=TIBCO_HOME/tibcojre64/*version*/lib	-DJDK_LIB=*OPEN_JDK_HOME*/lib

studio.ini file for Linux

Existing variable and value	Change to
-vm *TIBCO_HOME*/tibcojre64/*version*/lib/server	-vm *OPEN_JDK_HOME*/lib/server **Example:** -vm `/home/apps/Installations/Java/openjdk11-11.0.2/lib/server`

DJDK_LIB=*TIBCO_HOME*/tibcojre64/*version*/lib	DJDK_LIB=*OPEN_JDK_HOME*/lib
	
	Example:
	
	`DJDK_LIB=/home/apps/Installations/Java/openjdk11-11.0.2/lib`

studio.ini file for macOS

Existing variable and value	Change to
-vm *TIBCO_ HOME* /tibcojre64/ *version* /Contents/Home/lib/jli/libjli.dylib	-vm *OPEN_JDK_HOME*/Contents/Home/lib/jli/libjli.dylib **Example:** -vm `/Users/apple/Installations/Java/openjdk11-11.0.2/Contents/Home/lib/jli/libjli.dylib`

2. Update the `.tra` files for the `BE Engine` and `Studio Tools` as follows:

be-engine.tra for Windows

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*
tibco.env.JVM_LIB_PATH=*TIBCO_HOME*/tibcojre64/*version*/bin/server/jvm.dll	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/bin/server/jvm.dll
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/bin/server	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/bin/server

be-engine.tra for Linux

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*
tibco.env.JVM_LIB_PATH=*TIBCO_HOME*/tibcojre64/*version*/lib/server/libjvm.so	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/lib/server/libjvm.so
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/lib	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/lib

be-engine.tra for macOS

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*/Contents/Home	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*/Contents/Home
tibco.env.JVM_LIB_PATH=*TIBCO_HOME* /tibcojre64/	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/Contents/Home/lib/jli/libjli.dylib

Existing variable and value	Change to
version/Contents/Home/lib/jli/libjli.dylib	
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/ Contents/Home/lib/	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/Contents/Home/lib

studio-tools.tra for Windows

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=TIBCO_HOME/tibcojre64/*version*	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*
tibco.env.JVM_LIB_PATH=TIBCO_HOME/tibcojre64/*version*/bin/server/jvm.dll	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/bin/server/jvm.dll
tibco.env.JVM_LIB_DIR=TIBCO_HOME/tibcojre64/*version*/bin/server	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/bin/server
tibco.env.LIBPATH TIBCO_HOME/tibcojre64/*version*/bin/server	tibco.env.LIBPATH *OPEN_JDK_HOME*/bin/server

studio-tools.tra for Linux

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*
tibco.env.JVM_LIB_PATH=*TIBCO_HOME*/tibcojre64/*version*/lib/server/libjvm.so	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/lib/server/libjvm.so
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/lib	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/lib

Existing variable and value	Change to
tibco.env.LIBPATH *TIBCO_HOME*/tibcojre64/*version*/lib	tibco.env.LIBPATH *OPEN_JDK_HOME*/lib

studio-tools.tra for macOS

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*/Contents/Home	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*/Contents/Home
tibco.env.JVM_LIB_PATH=*TIBCO_HOME* /tibcojre64/ *version*/Contents/Home/lib/jli/libjli.dylib	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/Contents/Home/lib/jli/libjli.dylib
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/Contents/Home/lib/	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/Contents/Home/lib
tibco.env.LIBPATH *TIBCO_HOME*/tibcojre64/*version*/Contents/Home/lib/	tibco.env.LIBPATH *OPEN_JDK_HOME*/Contents/Home/lib/

3. (Optional) To use RMS or TIBCO BusinessEvents Enterprise Administrator Agent, update the appropriate .tra files, (`be-rms.tra` or `be-teagent.tra`) as follows:

   be-rms.tra and be-teagent.tra for Windows

Existing variable and value	**Change to**
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*
tibco.env.JVM_LIB_PATH=*TIBCO_HOME*/tibcojre64/*version*/bin/server/jvm.dll	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/bin/server/jvm.dll
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/bin/server	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/bin/server
(for rms.tra only) tibco.env.JDK_LIB=*TIBCO_HOME*/tibcojre64/*version*/lib	tibco.env.JDK_LIB=*OPEN_JDK_HOME*/lib

   be-rms.tra and be-teagent.tra for Linux

Existing variable and value	**Change to**
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*
tibco.env.JVM_LIB_PATH=*TIBCO_HOME*/tibcojre64/*version*/lib/server/libjvm.so	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME*/lib/server/libjvm.so
tibco.env.JVM_LIB_DIR=*TIBCO_HOME*/tibcojre64/*version*/lib	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/lib
(for rms.tra only) tibco.env.JDK_LIB=*TIBCO_HOME*/tibcojre64/*version*/lib	tibco.env.JDK_LIB=*OPEN_JDK_HOME*/lib

be-rms.tra and be-teagent.tra for macOS

Existing variable and value	Change to
tibco.env.TIB_JAVA_HOME=*TIBCO_HOME*/tibcojre64/*version*/Contents/Home	tibco.env.TIB_JAVA_HOME=*OPEN_JDK_HOME*/Contents/Home
tibco.env.JVM_LIB_PATH=*TIBCO_HOME* /tibcojre64/ *version* /Contents/Home/lib/server/libjvm.dylib	tibco.env.JVM_LIB_PATH=*OPEN_JDK_HOME* /Contents/Home/lib/server/libjvm.dylib
tibco.env.JVM_LIB_DIR=*TIBCO_HOME* /tibcojre64/*version*/Contents/Home/lib/	tibco.env.JVM_LIB_DIR=*OPEN_JDK_HOME*/Contents/Home/lib/
(for rms.tra only) tibco.env.JDK_LIB=*TIBCO_HOME* /tibcojre64/*version*/Contents/Home/lib	tibco.env.JDK_LIB=*OPEN_JDK_HOME*/Contents/Home/lib

4. To view the changes that you make in the `.tra` files, restart TIBCO BusinessEvents Studio and other components such as RMS and TIBCO BusinessEvents Enterprise Administrator Agent.

Element Refactoring Operations

When you change a project element, all references to that element must be updated accordingly. If the change affects only the structure and not the behavior of the project, this operation is known as *project refactoring*. This chapter explains how to use the refactoring features, and some related features to do with copy-paste operations.

Renaming Moving Deleting and Copy-Pasting Elements

Changes that affect the structure of a project, but not its behavior, are known as *project refactoring* changes. Refactoring ensures that the project structure remains self-consistent.

Copy-paste operations are not strictly speaking refactoring operations. However, some refactoring is also done to support these operations, so they are included here.

Note: You can copy items from one project to other projects in the workspace. Ensure that the items are suitable for their destination projects.

Moving, renaming, deleting, or copy-pasting project elements are changes that often affect other parts of a TIBCO BusinessEvents Studio project. Names of elements, element properties, and element locations, are referenced in various parts of a project such as rules, rule functions, and concept relationship properties.

When you change a project element, all references to that element must be updated accordingly. The refactoring wizard has a preview page that enables you to review all these changes (see Working with the Preview Page).

This section explains the refactoring procedures. See Automatic Refactoring Actions and Limitations to understand what TIBCO BusinessEvents does for each type of refactoring operation.

Updating All References

To ensure the integrity of the project, ensure that you make all changes to all locations where a renamed or moved item is referenced. Disable such changes only if you are certain that either there are no references to the element or there are unusual circumstances that justify such action.

Tip: Eclipse Tips

To Undo Changes

You can undo additions or deletions in a project. Click **Edit > Undo** or press **Ctrl+Z**.

To Revert to an Earlier Version or Restore a Deleted Resource

Using Eclipse you can compare your work with local history and to revert to any earlier saved version (not just the last saved version). Right-click a resource in BusinessEvents Studio Explorer and select **Compare With > Local History** or **Replace With > Local History**. Also, if you right-click a folder and select **Restore From Local History** you can restore items deleted from that folder.

Project Level Actions

You can rename a project (select **File > Rename**), copy and paste a project. However, you cannot move a project.

Renaming Moving and Deleting Elements

Renaming, moving, and deleting are refactoring actions that can have an effect on other parts of the project where element names and locations are referenced.

Renaming a Project Element

Procedure

1. Rename the element using one of the following methods:
    - Right-click a project element in BusinessEvents Studio Explorer and select **Refactor > Rename**.
    - Highlight the element, and then select **File > Rename**.
    - Highlight the element, and then press **F2**.

    The first page of the Rename Element wizard appears.
2. In the **New Name** field, type the new name.
3. If you do not want to rename the element in places where it is referenced, clear the **Update References** checkbox. However, you should update all references to ensure the integrity of the project (see Updating All References).
4. Click **Preview**. One of the following occurs:
    - The preview page displays so you can examine the effect of this change. See Working with the Preview Page for details.
    - The problems page displays if renaming cannot be done. For example, because a new element name is used by an existing element. Click **Back** to fix the problem, or **Cancel** to cancel the rename.
5. In the preview page, clear checkboxes if you do not want the change to be made in some referenced locations. However, you should update all references to ensure the integrity of the project (see Updating All References).
6. To complete the change, click **OK**.

Moving an Element to a Different Project Folder

Procedure

1. Move the element using one of the following methods:
    - Right-click a project element name in BusinessEvents Studio Explorer and select **Refactor > Move**.
    - Highlight the element, then select **File > Move**.
    - Drag the element to the target folder (go to *Step 3*).
2. If you opened the Move Element wizard using menus, navigate the project tree in the first page of the wizard to select a destination folder, then click **Preview**.

3. One of the following occurs:
    - The preview page displays so you can examine the effect of this change. See Working with the Preview Page for details.
    - The problem page displays if the move cannot be made. Click **Back** to fix the problem, or **Cancel** to cancel the move.
4. In the preview page, clear checkboxes if you do not want the change to be made in some referenced locations. However, you should update all references to ensure the integrity of the project (see Updating All References).
5. To complete the change, click **OK**.

Delete an Element or Folder

When you delete a folder, all the elements within that folder are also deleted.

Procedure

1. Perform one of the following actions.
    - Right-click a project element or folder name in BusinessEvents Studio Explorer and select **Delete**
    - Highlight the element, then select **Edit > Delete**
    - Highlight the element, then press the **Delete** key
2. At the Delete Resources page, click **OK** to delete without previewing, or click **Preview** to preview the effect of the deletion.
3. If you click Preview, one of the following occurs:
    - The preview page is displayed so you can examine the effect of this change. For deletions, there is generally no information. Click **OK** to delete.
    - The problems page is displayed if there is a problem with the deletion. Click **Back** to fix the problem, or **Continue** to force the deletion, or **Cancel** to cancel the deletion.

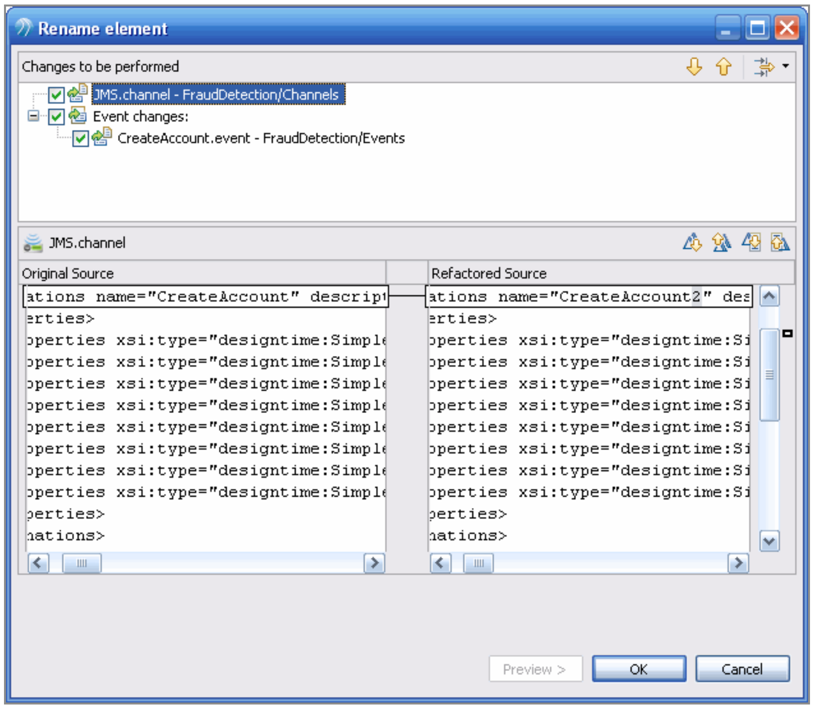
Deleting a Project

Procedure

1. Perform either of the following:
    - Right-click the project name in BusinessEvents Studio Explorer and select **Delete**.
    - Highlight the project name, then select **Edit > Delete**.
    - Highlight the project name, then press the **Delete** key.
2. If you want to remove the project contents on disk, select the **Delete Project Contents on Disk** checkbox. If you do not select this checkbox, then the project is removed from BusinessEvents Studio Explorer, but the project contents remain on disk.
3. Click **OK** to delete without previewing, or click **Preview** to preview the effect of the deletion.
4. If you click Preview, one of the following occurs:
    - The preview page is displayed so you can examine the effect of this change. For deletions, there is generally no information. Click **OK** to delete.
    - The problems page is displayed if there is a problem with the deletion. Click **Back** to fix the problem, or **Continue** to force the deletion, or **Cancel** to cancel the deletion.

Working with the Preview Page

During refactoring operations, you can click **Preview** to examine the effects of the operation on the project. If a pre-check finds issues that may prevent the operation from completing successfully, a problem page appears instead.



```
ations name="CreateAccount" descrip
erties>
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
operties xsi:type="designtime:Simple
erties>
ations>
```

```
ations name="CreateAccount2" des
erties>
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
operties xsi:type="designtime:Si
erties>
ations>
```

Checks in the upper panel indicate that the elements change as a result of a refactoring operation. Expand to take a closer look at individual folders and elements. All elements that appear are affected by the change. When you highlight an element, details of the change to be made are shown in the lower panel.

The Original Source panel on the left and the Refactored Source on the right use the persisted format of the element. Change bars indicate changed areas.

Do any of the following as needed to examine the changes and select a subset of the changes to be done on clicking OK:

- In the upper panel, use the arrows to navigate up and down a long list of changes.
- Click the **Filter Changes** button and select **Hide Derived Resources**. For example, a diagram is a derived resource. Diagrams are not persisted. You can easily recreate them. So you may not be interested in seeing those changes.

- If you want to apply the change to only some of the project elements, clear the checkboxes next to the elements as desired. For example, after you have completed the refactoring operation, you may wish to replace or delete an element. You do not need to apply the change to that element.

  Note: Accept all changes to be performed

  Your project can become corrupted if you do not make the changes throughout the project. Only deselect changes if you have a specific, valid reason to do so.

Copy-Pasting an Element

Copy-pasting is not a true refactoring operation, because it does change the behavior of the TIBCO BusinessEvents Studio project.

You would generally make manual changes to your project to use the newly created project element. However, some limited refactoring is done for your convenience. For example, the definition of a rule or rule function begins with its fully qualified name, such as the following:

```
rule SomeRules.Application_Rule
```

If you copy and paste the above rule into the folder `OtherRules`, the definition of the rule automatically changes to:

```
rule OtherRules.Application_Rule
```

Procedure

1. In BusinessEvents Studio Explorer, highlight the element (or folder) and perform one of the following actions.
    - Right-click the element and select **Copy**.
    - Press **Ctrl+C**.
    - Select **Edit > Copy**.

2. Highlight the project folder where you want to paste the element, and perform one of the following actions.
    - Right-click the folder and select **Paste**.
    - Press **Ctrl+V**.
    - Select **Edit > Paste**.

Result

If you are pasting an element to the same folder that you copied it from, a dialog appears enabling you to provide a different name. The default value is CopyOf *`oldname`*.

Automatic Refactoring Actions and Limitations

This section explains what is done for each kind of refactoring operation. It also lists some limitations.

Refactoring Limitations

Changes made to certain items are not refactored in this release. You must handle the reference updates manually.

References to moved or changed folders in strings

References to changed or moved folders are not updated in strings, including CDD and XSLT strings. CDD strings are used in the Cluster Definition Descriptor editor to point to project resources using their project path. XSLT strings are used in mapper functions, which are completed using the Function Argument Mapper. (Such references are, however, updated for entity refactoring operations).

Copy-paste of folders

Elements inside the pasted folder are not updated.

References to global variables and shared resources

Refactoring does not handle changes to global variables and shared resources. You must manually update references to global variables and shared resources that you change.

Note: TIBCO BusinessEvents destinations and property definitions cannot be moved.

Refactoring for Move and Rename Operations

Move and rename refactoring operations change only the structure of a project. For example, when you change a concept name, that name must change everywhere the concept is referenced in the project. If the element has its own file, the file must also be renamed.

References to the changed or moved element are handled as shown in the following table. Projects can be complex. This list covers the main cases.

Refactoring for Move and Rename Operations

Renaming this...	Updates reference in these places...
Concept	Concepts that inherit from this one Event expiry actions Property definitions for contained or referenced concepts State models Rules and rule functions
Event	Events that inherit from this one Event expiry actions Destination (Default Event) State models Rules and rule functions
Property of a concept or an event	Event expiry actions State models

Renaming this...	Updates reference in these places...
	Rules and rule functions
Domain Model	Domain models that inherit from this one Associated properties
Channel	Event default destination paths
Destination	Event default destination paths
Rule	The rule source Rules and rule functions
Rule Function	Event expiry actions State models The rule function source Rules and rule functions
Folder	All locations that this folder is used in a path. For example, the path to a default destination in an event, property definitions for contained or referenced concepts, and in rules and rule functions. For folder refactoring limitations, See Refactoring Limitations .

Refactoring for Delete Operations

Element deletion can affect project behavior. Ensure that your project behavior is as desired after the deletion refactoring.

Deletion removes all references to the deleted object only in specific cases:

- References to a deleted domain model are removed from the concepts that refer to it.
- References to a deleted state model are removed from the owning concept.

TIBCO ActiveSpaces Support

TIBCO BusinessEvents integrates with TIBCO ActiveSpaces using various methods. You can use the Legacy ActiveSpaces embedded cache, integrate with ActiveSpaces using catalog functions. Use ActiveSpaces as a persistence data store. You can also communicate with TIBCO ActiveSpaces using the channels such as, Legacy ActiveSpaces Channel or ActiveSpaces Channel.

TIBCO ActiveSpaces® software is a distributed in-memory data grid product. ActiveSpaces® features familiar database concepts (such as tables, rows, and columns), high I/O capacity, and network scalability. TIBCO ActiveSpaces uses TIBCO FTL messaging software for communication between an application program and data grid, for internal communication.

TIBCO FTL should be installed in your system as a prerequisite for TIBCO ActiveSpaces to communicate with the ActiveSpaces store. In the `be-engine.tra` file, the `ACTIVESPACES_HOME` must be set to TIBCO ActiveSpaces and `FTL_HOME` must be set to a TIBCO FTL installation location.

Supported ActiveSpaces versions

Compared to Legacy ActiveSpaces, the latest ActiveSpaces software is completely redesigned and reimplemented to make it more user-friendly for both end users and administrators. See Comparison Matrix for more information about the differences between the ActiveSpaces versions.

The following table lists which TIBCO ActiveSpaces versions are supported by the various methods that integrate TIBCO BusinessEvents with ActiveSpaces.

ActiveSpaces Versions Support

Integration Method	Legacy ActiveSpaces	ActiveSpaces
Embedded You can use ActiveSpaces as a TIBCO BusinessEvents cache server.	Yes	No

Integration Method	Legacy ActiveSpaces	ActiveSpaces
Data Store You can use ActiveSpaces as a TIBCO BusinessEvents persistence data store.	No	Yes
External You can integrate TIBCO BusinessEvents with TIBCO ActiveSpaces using catalog functions.	Yes See Built-in Functions for more information.	Yes See Data Type Mapping and CEP Store Functions for more information.
Channel You can use the channels to notify TIBCO BusinessEvents of changes made to ActiveSpaces.	Yes See Legacy ActiveSpaces Channels for more information.	Yes See ActiveSpaces Channel for more information.

Key Concepts

Store

Store in BusinessEvents is synonymous with Data Grid in TIBCO ActiveSpaces.

Container

A container in BusinessEvents is synonymous with Tables in TIBCO ActiveSpaces.

Items

Items in TIBCO BusinessEvents are synonymous to Rows in TIBCO ActiveSpaces. TIBCO BusinessEvents applications can insert, delete, and retrieve individual rows. TIBCO BusinessEvents applications can query for rows that match a specified query filter.

To know more about the key concepts and advanced configuration of TIBCO ActiveSpaces, refer to the *TIBCO ActiveSpaces documentation*.

Data Type Mapping

As TIBCO ActiveSpaces does not support TIBCO BusinessEvents primitive data types such as integer and boolean, they can be stored in long data type in TIBCO ActiveSpaces. Also, TIBCO BusinessEvents arrays of primitive type can be mapped to an opaque data type in

TIBCO ActiveSpaces. To know more about TIBCO ActiveSpaces data types, refer to the TIBCO ActiveSpaces documentation.

Note: TIBCO ActiveSpaces is case insensitive, thus, for entities properties, do not use reserved words in either lowercase or uppercase. See Keywords and Other Reserved Words for a complete list.

The following table lists the data type mapping between TIBCO BusinessEvents and TIBCO ActiveSpaces.

TIBCO ActiveSpaces to TIBCO BusinessEvents Data Type Mapping

ActiveSpaces Data Type	TIBCO BusinessEvents Data Type
Long	Long/Integer/Boolean/Float
Double	Double
String	String
Opaque	String[]/Integer[]/Long[]/Double[]/Boolean[]/DateTime[]
DateTime	DateTime

BusinessEvents to ActiveSpaces Data Type Mapping

TIBCO BusinessEvents Data Type	ActiveSpaces Data Type
String/String[]	String/Opaque
Integer/Integer[]	Long/Opaque
Long/Long[]	Long/Opaque
Double/Double[]	Double/Opaque
Boolean/Boolean[]	Long/Opaque
DateTime/DateTime[]	DateTime/Opaque

TIBCO BusinessEvents Data Type	ActiveSpaces Data Type
Float/Float[]	Long/Opaque

Example Project

TIBCO BusinessEvents provides a working example project for TIBCO ActiveSpaces connection present at `*BE_HOME*/examples/standard/ActiveSpaces`. Use the `readme.html` present at the same location to run the example. You can also import the example project in the TIBCO BusinessEvents Studio to understand more on how to integrate the project with TIBCO ActiveSpaces. For example, in the ActiveSpaces example, the `StartUp.rulefunction` uses `Store.ConnectionInfo.create` and `Store.connect` catalog functions to create a connection to TIBCO ActiveSpaces data store.

See CEP Store Functions for details on the catalog functions to integrate with ActiveSpaces.

Channels and Destinations

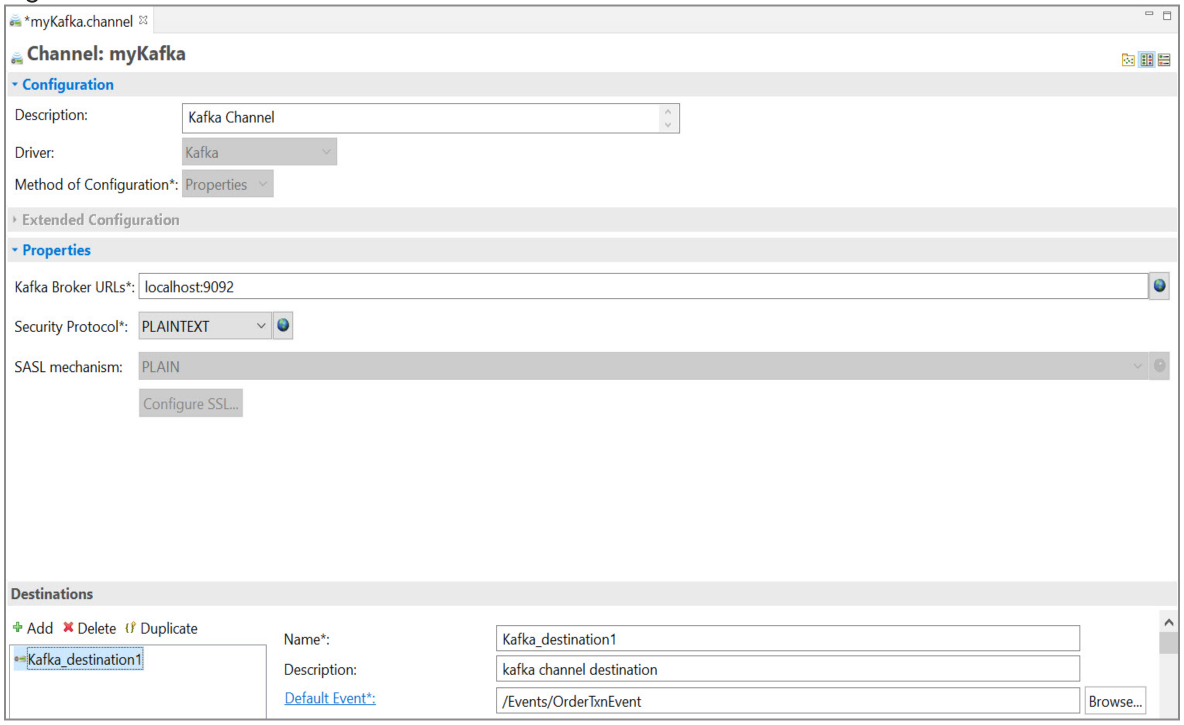
One project can have multiple channels of different types with multiple destinations as needed.

See *TIBCO BusinessEvents Architect Guide* for more information.

- For TRA file updates required for JMS channels that use TIBCO Enterprise Message Service, see *TIBCO BusinessEvents Administration*.
- For configuring SSL over HTTP, when the Identity Resource is of the type Certificate/KeyURL, you need to set up native Tomcat libraries. See *TIBCO BusinessEvents Installation guide*.

The general procedure for creating channels and destinations of all types is the same, though the configuration options are different. A TIBCO Kafka channel is shown as an example.

Figure 3: TIBCO Kafka Channel



*myKafka.channel

Channel: myKafka

Configuration

Description: Kafka Channel

Driver: Kafka

Method of Configuration*: Properties

Extended Configuration

Properties

Kafka Broker URLs*: localhost:9092

Security Protocol*: PLAINTEXT

SASL mechanism: PLAIN

Configure SSL...

Destinations

Add Delete Duplicate

Kafka_destination1

Name*: Kafka_destination1

Description: kafka channel destination

Default Event*: /Events/OrderTxnEvent

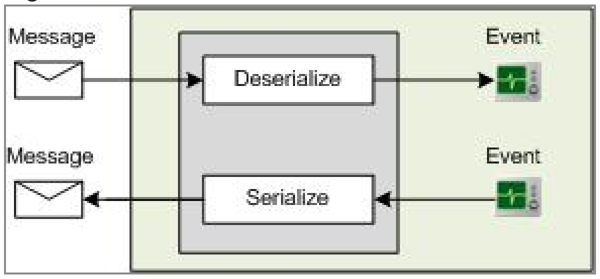
Browse...

Channel Serializers

For each type of channel (except local channels), TIBCO BusinessEvents uses a serializer to convert events to messages and a deserializer to convert incoming messages to events.

Local channels do not require serializers. HTTP channels use action rule functions on the message instead of converting messages to event using the deserializer.

Figure 4: Serializer and Deserializer Behavior



When you configure a destination, you select the appropriate serializer. (It actually includes the serializer and deserializer).

Mapping Incoming Messages to Non-default Events

Incoming messages can be mapped to default events or to specified event types.

The fields in a message header instruct TIBCO BusinessEvents to map the incoming message to a specified event type:

- The field named `_ns_` takes a namespace as a value. The namespace points to the event type, for example, `www.tibco.com/be/ontology/Events/MyEvent`
- The field named `_nm_` takes the name of the event, for example, `NewMyEvent`
- The field named `_extid_` takes the unique external id of the event.

These fields are added and filled automatically for events created using TIBCO BusinessEvents rules. You can also add these fields to the incoming messages from other sources if you have control of those messages. You can also use the **Include Event Type** field in the destination of the channel to suppress the original behavior of including _ns_ and _nm_ fields during serialization and deserialization.

Working with Local Channels

Local channels are used in rules or rule functions to route events to an appropriate agent running in the same engine (processing unit).

Local channels are useful in two cases:

- For applications using In Memory object management (generally used only for testing)
- For certain scenarios where an inference agent is co-deployed with a query agent. See *TIBCO BusinessEvents Event Stream Processing Query Developer Guide*.

Local channels pass the same event object between the agents. Consuming the event in one agent does not affect other agents that also received the event over a local channel. A use count is maintained for each event to track how many agents have received the event but not consumed it. The use count of the event is incremented depending on the number of agents that it is routed to. When an event is consumed in one agent, TIBCO BusinessEvents deletes the reference to the event in that agent, and it decrements the use count of the original event instance.

Local channel destinations can use an event preprocessor and have no default events. To route an event to a local channel's destination, use the `Event.routeTo()` function. (You can use this function for other purposes too.)

In the provided example, `*BE_HOME*/Examples/MultipleSessionsAndLocalChannel`, events containing small orders are sent to the agent that deals with small orders as follows:

```
Event.routeTo(order, "/Channels/Local/toSmall", "");
```

The signature of this function is as follows:

```
SimpleEvent routeTo(SimpleEvent event, String destinationPath, String properties)
```

In TIBCO BusinessEvents, you can add the following channels:

- Local
- HTTP
- JMS
- Kafka

- Legacy ActiveSpaces
- FTL
- ActiveSpaces
- Hawk

Adding a Channel

Channels (except for local channels) represent physical connections to a resource, such as a JMS server, HTTP server or client, Hawk domain, or a metaspace in Legacy ActiveSpaces.

Procedure

1. In BusinessEvents Studio Explorer, right-click the folder where you want to store the channel and select **New > Channel**. You see the New Channel Wizard.
2. In the **Channel name** field, type a name for the channel. In the **Description** field, type a description.

   Note: You cannot change the channel name in the channel editor. To change the name of any channel, right-click the element in Studio Explorer and select **Refactor** > **Rename**. See Element Refactoring Operations for more details.

3. In the **Driver type** field, select the appropriate driver.
4. Click **Finish**. You see the Channel editor.

What to do next

Use the channel editor to configure the channel and destination properties. See Editing a Channel.

Editing a Channel

After adding a channel, you can edit it using the Channel Editor.

Procedure

1. Edit **Description** as needed.
2. In the **Driver type** field, the driver you selected in the wizard is selected. If you want to change to a different type of channel, you can do so now. Select one of the available drivers.
3. For all channel types (except Local and HTTP), from the **Method of Configuration** dropdown list, select one of the following:
   - **Properties** - Select `Properties` to configure this channel resource using properties. When you select Properties, a Properties section displays appropriate fields for the type of channel. See Channel Resource Reference for help with entering the correct values.

     Note: For HTTP channels only, in the **serverType** field of the Extended Configuration section, TOMCAT is preselected as the server type and cannot be changed.

   - **Resource** - Select `Resource` if you have a shared resource in your project whose properties you want to reuse for this channel. For HTTP channels, the Resource is preselected and cannot be changed.

     Note: The path to the resource and the resource name should not contain any of the keywords or other reserved words listed in the *TIBCO BusinessEvents Architect Guide*.

4. If you selected Resource in the **Method of Configuration** field, in the **Resource** field click **Browse** and select the shared resource you want to use.

   First create a shared resource for the selected driver type. See Adding a Shared Resource for more information about how to create a shared resource.
5. If you are configuring an HTTP channel, click **Advanced** and configure tuning properties.
6. Continue to the section Adding a Destination to a Channel and add destinations to the channel as needed.

Adding a Destination to a Channel

Destinations in a channel represent listeners to messages from that channel. They can also send messages to the resource. One channel can have multiple destinations. Each is shown in the Destinations section of the Channel editor.

Procedure

1. If it is not already open, open the editor for the channel to which you want to add a destination. To open the channel editor, double-click the channel name in the BusinessEvents Studio Explorer view.
2. In the Destinations section, click **Add**.
3. Enter a **Name** and **Description** for the destination.
4. In the **Default Event** field, browse to and select the event to be created (by default) from incoming messages received by this destination. If you have not yet created the event, you can select the default event later.
5. In the **Serializer/Deserializer** field, select the appropriate class. See the following sections for more details:
    - For JMS destinations, see the *TIBCO BusinessEvents Architect Guide*.
    - For HTTP and SOAP destinations, Adding a Destination (Destination configuration is the same for SOAP and HTTP destinations). See Manually Creating Resources to Work with SOAP Services and Creating Resources Using the WSDL Import Utility for more information about creating SOAP channels and destinations).
    - For Hawk destinations, the Hawk channel allows TIBCO BusinessEvents to receive events from the Hawk monitor and transform them to events in TIBCO BusinessEvents.
    - See Kafka Channel Serializers for Kafka destinations.
    - See Working with Legacy ActiveSpaces Channels for Legacy ActiveSpaces destinations.

      Various fields appear, depending on your selection.
    - For FTL destinations, select com.tibco.cep.driver.ftl.serializer.FTLSerializer
    - For ActiveSpaces destinations, see ActiveSpaces Channel

6. *(For JMS, and HTTP only)* In **Include Event Type**, select when to use the `_ns_` and `_nm_` fields from the following options:
   - When serializing and deserializing
   - Only when serializing
   - Only when deserializing
   - Never

   See Destination Resource Common Properties for more details about this property.
7. Complete the rest of the properties for the type of destination that you are creating. See Destination Resource Reference for details.
8. Save the project.

Channel Resource Reference

Channels allow TIBCO BusinessEvents to listen to and send out messages. Channels contain destinations.

You can configure channels of different types, using the appropriate driver. See *TIBCO BusinessEvents Architect Guide*.

To configure an HTTP channel resource, select an HTTP connection resource. No other channel resource fields require configuration.

Note: Local channels are in memory. Information in a local channel could be lost if the TIBCO BusinessEvents engine fails.

Channel Configuration Reference

Field	Global Variable	Description
Name	No	(Shown in the wizard and then in the editor title only.) The name to appear as the label for the resource. Names follow

Field	Global Variable	Description
		Java variable naming restrictions. Do not use any reserved words.
Description	No	Short description of the resource.
Driver	No	Displays the type of the channel that you are configuring. You can change the channel type only if there are no destinations defined in the channel. The following are the available channel types: • Legacy ActiveSpaces • ActiveSpaces • FTL • Hawk • HTTP • JMS • Kafka • Local
Method of Configuration	No	Select the method that you want to use for channel configuration. You can either configure the channel properties directly or use a shared resource.

Field	Global Variable	Description
		• **Properties** - Select Properties to configure the channel resource using properties. The configuration properties are displayed based on the **Driver** value. See the following sections for the configuration of different channel types: ◦ Legacy ActiveSpaces Channel Configuration Properties ◦ ActiveSpaces Channel Configuration Properties ◦ FTL Channel Configuration Properties ◦ JMS Channel Configuration Properties ◦ Kafka Channel Configuration Properties ◦ Pulsar Channel Configuration Properties • **Resource** - Select Resource if you have a shared resource in your project whose properties you want to reuse for this channel. See Adding a Shared Resource.
Resource	No	If you choose Resource as the method of configuration, the **Resource** field appears. Browse to and select the resource you want to use. For convenience, you can open the selected resource by clicking the underlined label. **Note:** The path to the resource and the resource name cannot contain any of the words listed in Keywords and Other Reserved Words.

Legacy ActiveSpaces Channel Configuration Properties

Legacy ActiveSpaces Channel Properties

Field	Global Variable	Description
Metaspace Name	No	The name of a particular metaspace instance in Legacy

Field	Global Variable	Description
		ActiveSpaces that the channel must connect to. The metaspace must be created and initialized before the channel can use it at run time.
Discovery URL	No	Specifies how a metaspace instance discovers the current metaspace members. Multicast discovery can use the PGM - Pragmatic General Multicast protocol. If using the PGM protocol, the multicast URL is expressed in the following format: `tibpgm://`*`destination port`*`/`*`interface`*`;`*`discovery group address`*`/`*`optional transport arguments`*, where • *`destination port`* specifies the destination port used by the PGM transport. If not specified, the default value of 7888 is used. • *`interface`*`;`*`discovery group address`* specifies the address of the interface to be used for sending discovery packets, and the discovery group address to be used. If not specified, it defaults to the default interface and discovery address, 0.0.0.0;239.8.8.8. • *`optional transport arguments`* specifies a semicolon-separated list of optional PGM transport arguments. By default, the PGM transport is tuned to provide the best performance according to the most common deployment architectures. Any inappropriate values in the optional arguments could easily result in degraded performance of the product. So, the values of those optional arguments should be changed only when it is necessary and with care.
ListenUrl	No	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. This

Field	Global Variable	Description
		address and port are specified by the member's Listen URL. If not specified, the discovery process uses the default IP address and the first free TCP port that can be acquired from the operating system (starting 5000 and above).
RemoteListenUrl	No	This field is used to configure a Legacy ActiveSpaces Channel as a remote-discovery proxy. In this case, any remote client can connect to a Legacy ActiveSpaces metaspace through the Legacy ActiveSpaces Channel node.
EnableSecurity	No	Enables security for the Legacy ActiveSpaces Channel when selected. **Note:** Some fields are activated only for the specific security role or authorization policy.
SecurityRole	No	Security role of a node for the secure Legacy ActiveSpaces Channel in the metaspace. The values are: • Controller: It is dedicated to enforcing a security domain's defined security behavior for a metaspace associated with the security domain. Security domain controllers are the only discovery nodes in a metaspace. • Requestor: It just requires access to the data in the data grid, such as a seeder or a leech, and which need to be authorized by a controller. Requesters can never be used by a discovery node. The controller nodes are configured with a security policy file. The requester nodes provide a token file and additional credentials to the controller for authentication. The controller performs authentication as defined in its policy file and the credentials provided by the requester.

Field	Global Variable	Description
		If the Controller option is selected, then the following fields become active: • **Identity Password** • **PolicyFile** If the Requestor option is selected, then the following fields become active: • **Identity Password** • **TokenFile** • **Credential**
Identity Password	No	The password for the identity key is in the security policy file.
PolicyFile	No	Absolute path to the policy file that contains the security settings that the controller node enforces. It is generated using the as-admin utility.
TokenFile	No	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.
Credential	No	Authentication policy to be used for authentication as specified in the policy file. The values are: • USERPWD - A username and password-based authentication is used. It activates the following fields: 　◦ **Domain** 　◦ **Username** 　◦ **Password**

Field	**Global Variable**	**Description**
		• X509V3 - The authentication source is an LDAP configured with certificate based authentication. It activates the following fields: ◦ **KeyFile** ◦ **PrivateKey**
Domain	No	Domain name for system-based user authentication.
Username	No	Username for LDAP and system-based authentication.
Password	No	Password for LDAP and system-based authentication.
KeyFile	No	The absolute path for a file containing the key to use for LDAP with the certificate based authentication.
PrivateKey	No	The password for the identity key in the LDAP identity file is specified in **KeyFile**.

ActiveSpaces Channel Configuration Properties

ActiveSpaces Channel Properties

Field	**Global Variable**	**Description**
Realm Server URL	Yes	The URL at which the TIBCO BusinessEvents can connect to the TIBCO FTL realm server. **Default URL:** http://localhost:8080
Grid Name	Yes	The name of the data grid to which the ActiveSpaces channel connects.
Username	Yes	A valid username for the TIBCO FTL realm server.

Field	Global Variable	Description
Password	Yes	The password is assigned to the username specified in **UserName**, for accessing the TIBCO FTL realm server.
Use SSL	No	Check this box to use a Secure Socket Layer (SSL) protocol while connecting to the TIBCO FTL realm server. If checked, the **Trust Type** and **Identity** fields are activated.
Trust Type	Yes	Specifies whether you want to use a trust file for SSL authentication.
Identity	No	The **Identity** field is activated if you have selected the **Trust File** option in the **Trust Type** dropdown. Browse to select the trust file from your system.

FTL Channel Configuration Properties

FTL Channel Properties

Field	Global Variable	Description
RealmServer	Yes	The URL at which BusinessEvents can connect to the FTL realm server. The default value is: http://localhost:8080
Username	Yes	A valid username for the FTL realm server. **Note:** This field is required only when the basic authentication is enabled at the realm server.
Password	Yes	The password is assigned to the username specified in **UserName**, for accessing the FTL realm server. **Note:** This field is required only when the basic authentication is enabled at the realm server.

JMS Channel Configuration Properties

JMS Channel Properties

Field	**Global Variable**	**Description**
ProviderURL	Yes	The URL at which TIBCO BusinessEvents can contact the Enterprise Message Service server. Example: `tcp://localhost:7222`
Username	Yes	A valid username for the Enterprise Message Service server.
Password	Yes	The password is assigned to the username specified in **UserName**, for accessing the Enterprise Message Service server.
IsTransacted	Yes	Accepts `true` or `false`. Specify `true` if the session has transaction semantics. Specify `false` if it has non-transaction semantics. See TIBCO Enterprise Message Service documentation for more information about the `IsTransacted` property.
ClientID	Yes	The unique client ID of the connection.

Kafka Channel Configuration Properties

Kafka Channel Properties

Field	**Global Variable**	**Description**
Kafka Broker URLs	Yes	List of URLs (host and port pairs) that are used for establishing the initial connection to the Kafka cluster. The format of the URL is *`host1:port1,host:port2,....`* For more details, refer to the setting bootstrap servers topic in the Kafka Documentation (https://kafka.apache.org/documentation/).

Field	Global Variable	Description
Poll Previous Messages	Yes	Enables to consume the messages based on the value in `Polling Timestamp` when the checkbox is selected. The checkbox is not selected by default.
Polling Timestamp	Yes	The messages are consumed from the specified timestamp. The format of the timestamp is `YYYY-MM-DD HH:MM:SS`. The `Polling Timestamp` is editable when the `Poll Previous Messages` checkbox is selected.
Security Protocol	Yes	The security protocol is implemented in the Kafka broker. This protocol must match with the security protocol configured in the Kafka broker. For more details about how to configure security for a Kafka broker, refer to the *Kafka Documentation* at https://kafka.apache.org/documentation/#security. You can select the following security protocol values: • `PLAINTEXT` - No authentication or encryption mechanism is implemented for Kafka communication. • `SASL_PLAINTEXT` - SASL authentication is implemented without SSL for Kafka communication. • `SSL` - SSL based connection between Kafka clients and brokers. • `SASL_SSL` - SASL authentication is implemented with SSL for Kafka communication. The default value is `PLAINTEXT`.
SASL Mechanism	Yes	Type of SASL mechanism is implemented on a Kafka broker. This field is active only for `SASL_PLAINTEXT` and `SASL_SSL` security protocol. The following SASL mechanisms are supported for the Kafka channel: • `GSSAPI` (Kerberos) • `PLAIN`

Field	Global Variable	Description
		• `SCRAM-SHA-256` • `SCRAM-SHA-512`
Configure SSL	N/A	Click the **Configure SSL** button to open the SSL configuration window and configure SSL details. This button is active only for `SSL` and `SASL_SSL` security protocol. The following fields are available on the SSL configuration window: **Trusted Certificates Folder** Location of the trusted certificates on the client machine. The trusted certificates are a collection of certificates from servers with whom you establish connections. If the server you wish to establish a connection with, presents a certificate that does not match one of your trusted certificates, the connection is refused. Trusted certificates must be imported into a folder, and then you can select the folder in this field. **Identity** The location of the identity shared resource file that contains the information to authenticate BusinessEvents client identity. For more information on identity resources, see Identity Resource Reference. **Trust Store Password** Password to access the trust store file.

Pulsar Channel Configuration Properties

Pulsar Channel Properties

Field	Global Variable	Description
Pulsar URL	Yes	The URL of the Pulsar broker service. The format of the URL is `pulsar://host:port` If there are multiple hosts, the URL format must be: `pulsar://host1:port,host2:port`
Pulsar Namespace	Yes	The namespace to be used with all destinations of this channel.
Connection Timeout (ms)	Yes	Duration to wait for a connection to a Pulsar broker to be established
Security Protocol	Yes	The security protocol is implemented in the Pulsar broker. This protocol must match with the security protocol configured in the Pulsar broker. For more details about how to configure security for Pulsar broker, refer to the *Pulsar Documentation* at https://pulsar.apache.org/docs/. You can select one of the following security protocol values: • `PLAINTEXT`: No authentication or encryption mechanism is implemented for Pulsar communication. • `mTLS:` For mTLS authentication, the server uses the trust certificate to verify that the client has a key pair that the certificate authority signed. • `JWT`: Pulsar supports authenticating clients using security tokens based on JSON Web Tokens. • `OAuth` 2: Pulsar supports authenticating clients using OAuth 2 access tokens. • HTTP basic: Basic authentication is a simple authentication scheme built into the HTTP protocol, which uses base64-encoded username and password pairs as credentials.

Field	Global Variable	Description
		The default value is `PLAINTEXT`. See Setting Up Authentication for Pulsar for more information about setting up authentication.
Configure	NA	Click the **Configure** button to open the authentication configuration dialog. The following fields are available on the configuration window: If you select the **mTLS** protocol, the following fields are displayed: **File Type** Select the keystore format: PEM or JKS. **Keystore File** Location of the keystore file for the BusinessEvents application. Create Identity Resource for KeyStore file. For PEM extension, the file must be `client.cert.pem` and for a JKS file must be `client.keystore.jks`. **Trust Store File** Browse and select the Identity Resource that contains the information to authenticate BusinessEvents client identity. If not already present, first create an Identity Resource. In the Identity Resource, the Identity file must be `client.truststore.jks`. **Client Key Certificate** Location of the client certificate file in the `.pem` format. This is applicable for PEM file format. Create Identity Resource for key file. The file extension must be `client.key-pk8.pem`. **CA Certificate** Browse the CA certificate. A CA certificate is a digital

Field	Global Variable	Description
		certificate issued by a certificate authority (CA). A CA certificate is required if the File Type is PEM. If you select the **JWT** protocol, the following field is displayed: **Authentication Token** Specify the JSON Web Token for authentication. If you select the **OAUTH 2** protocol, the following fields are available: **Authentication Config File** Provide a JSON configuration file path that includes type, client_id, client_secret, client_email, issuer_url parameter details, Issuer, and Audience URL. **Issuer URL** The URL of the authentication provider. An OAuth provider validates the request for a token for the given client ID and returns JWT as a response. **Audience URL** URL of the Resource server. Resource server is the server that hosts the protected resources. If you select the **HTTP_BASIC** protocol, the following fields are available: **Authentication Identity** The location of the identity shared resource file that contains the information to authenticate BusinessEvents client identity. For more information on identity resources, see Identity Resource Reference.

Destination Resource Reference

Within each channel, destinations direct the incoming and outgoing information. A channel resource is not ready for use until it has at least one destination.

Destinations of each channel have some common properties and configuration properties specific to the channel type. The following sections provide information about these configuration properties:

- Destination Resource Common Properties
- HTTP Destination Configuration Properties
- ActiveSpaces Destination Configuration Properties
- TIBCO FTL Destination Configuration Properties
- JMS Destination Configuration Properties
- Kafka Destination Configuration Properties
- Pulsar Destination Configuration Properties
- Local Destination Configuration Properties

Destination Resource Common Properties

The Destinations section of a channel has the following fields.

Destination Resource Common Properties

Field	Global Variable	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words.
Description	No	Short description of the resource.
Default Event	No	The event is to be created from incoming messages unless otherwise specified. For convenience, you can open the selected event resource by clicking the underlined label. This field is optional if you specify an event type in the incoming message. The property is not used for the local channel.

Field	Global Variable	Description
Serializer/Deserializer	No	Specify a serializer class to convert messages to simple events and simple events to messages.
Include Event Type	Yes	*(Only for HTTP, JMS, and Kafka channels)* Specifies when to suppress the original behavior of including `_ns_` and `_nm_` fields during serialization and deserialization. For more details on `_ns_` and `_nm_` fields, see Mapping Incoming Messages to Non-default Events. The values are: • When Serializing and Deserializing: always include `_ns_` and `_nm_` fields. • Only when Serializing: for outgoing JMS messages, `_ns_` and `_nm_` fields are set in the JMS message headers. For incoming JMS messages, `_ns_` and `_nm_` fields are ignored. • Only when Deserializing: the `_ns_` and `_nm_` fields are suppressed in the outgoing message but are used in incoming messages. • Never: the `_ns_` and `_nm_` fields are suppressed in incoming as well as outgoing messages.

HTTP Destination Configuration Properties

HTTP Destinations Configuration Properties

Field	Global Variable	Description
Reply Event	No	Optional. Defines a representative response event
Is JSON Payload	No	Specifies whether the payload is JSON

Field	Global Variable	Description
Type	No	Specifies the type of the HTTP destination. The following are the values of the Type field: • Default - Specify the Default Event and **Serializer/Deserializer** fields for the destination. • PageFlow - Enables Action Rule Function based approach. If selected, the system disables the **Default Event** and **Serializer/Deserializer** fields for input. • WebSocket - If the selected, the destination uses the WebSocket protocol and the com.tibco.cep.driver.http.serializer.WebSocketMessageSerializer is selected for the **Serializer/Deserializer** field.
Context Path	No	Context URI for the web application. The field is active for input if the value of the **Type** field is PageFlow.
Action Rule Function	No	The Action Rule function to be run when an HTTP message arrives at the context URI for the web application. The field is active for input if the value of the **Type** field is PageFlow.

ActiveSpaces Destination Configuration Properties

ActiveSpaces Channel Destination Configuration Properties

Field	Global Variable	Description
Table Name	Yes	The name of the table to which the ActiveSpaces channel connects.
Filter	Yes	String specified to evaluate rows and refine the set of rows to work. A filter string can be seen as what would follow the *where* clause in a `select * from where` statement. See Filters for more information.

Field	Global Variable	Description
Consumption Mode	No	Specifies the consumption mode for the ActiveSpaces event as one of: • Event Listener - listens for specific events to occur, and invokes a callback function from TIBCO ActiveSpaces.
PutEvent	No	When selected, the Event Listener listens for any Put events on the table and invokes a callback function when such an event occurs.
DeleteEvent	No	When selected, the Event Listener listens for any Delete events on the table and invokes a callback function when such an event occurs
ExpireEvent	No	When selected, the Event Listener listens for any Expire events on the table and invokes a callback function when such an event occurs.

TIBCO FTL Destination Configuration Properties

TIBCO FTL Destinations Configuration Properties

Field	Global Variable	Description
Application Name	Yes	The name of the application that communicates using TIBCO FTL software.
Endpoint Name	Yes	Name of endpoint for the application.
Instance Name	Yes	Name of the application instance. The default value is default.
Format Name	Yes	The format name is defined in the TIBCO FTL software.

Field	Global Variable	Description
Content Matcher	No	The query clause to filter messages. The syntax of the query is as follows: `{"FIELD_NAME1" : FIELD_VALUE1, "FIELD_NAME2" : FIELD_VALUE2 }`
Durable Name	Yes	Durable name of the FTL subscriber

JMS Destination Configuration Properties

See TIBCO Enterprise Message Service documentation for more details on these settings.

JMS Destinations Configuration Properties

Field	Global Variable	Description
Is JSON Payload	No	Specifies whether the payload is JSON
Queue	Yes	Specifies whether the destination is a queue or a topic. Select the checkbox if the destination is a queue. If the destination is a topic, do not select it.
Name	Yes	Required. The name of the queue or topic. TIBCO BusinessEvents ignores JMS destinations with null or empty-string queue or topic names. It logs an error message for the ignored destinations. If a JMS message is sent out through an ignored destination, TIBCO BusinessEvents generates an exception, and the message is not sent out. TIBCO BusinessEvents also does not receive JMS messages (events) through these ignored destinations.
Selector	Yes	Specifies a filter to pick up messages from the destination. This is a standard JMS selector based on SQL92 semantics.

Field	Global Variable	Description
DeliveryMode	No	The delivery mode property instructs the server concerning persistent storage for the message. Select one of the following: • PERSISTENT (default)—In JMS message headers this is represented by the code 2. • NON-PERSISTENT — In JMS message headers this is represented by the code 1. • RELIABLE — This value is an extension to the standard used in the TIBCO Enterprise Message Service. In message headers, this is represented by the code 22. You can also set a delivery mode in an event.
AckMode	Yes	The acknowledgment mode. See JMS Message Acknowledgement Modes for more details about the various modes. The default value is `EXPLICIT_CLIENT_ACKNOWLEDGE`.
Priority	No	The message is priority. Takes a numerical value between 0 and 9. Larger numbers represent a higher priority. You can also set a priority in an event. The default value is 4.
TTL	Yes	The length of time that the message lives (in milliseconds) before expiration. If set to 0, the message does not expire. You can also set a TTL (`JMSExpiration`) in an event. The default value is 0.
DurableSubscriberName	Yes	For destinations that are JMS Topics, if you provide a DurableSubscriber Name, the destination becomes a JMS

Field	Global Variable	Description
		durable topic subscriber with the specified name. If you do not provide a value, the destination becomes a non-durable topic subscriber. The value of this property can be any unique string and can include any global variables. TIBCO BusinessEvents provides a set of case-sensitive variables that produce a unique DurableSubscriberName string. The default value is: `%%EngineName%%:%%SessionName%%:%%ChannelURI%%:%%DestinationName%%`.
SharedSubscriptionName	Yes	Specify the shared subscription name for the consumers. Among the consumers having the same **SharedSubscriptionName** only one receive the message. Thus, an application can have multiple consumers performing a job, and these consumers can share the subscription without each one receiving the same copy of the message. The shareable and durable nature of a subscription can be used in different combinations. The following are the possible combinations: • *non-shared and non-durable subscription* - The subscription can have only one consumer and messages sent during the offline time are lost. • *non-shared and durable subscription* - The subscription can have only one consumer and can receive the messages that were sent when it was offline. • *shared and non-durable subscription* - The subscription can have many consumers and if they all went offline, the messages sent during the offline time are lost.

Field	Global Variable	Description
		• *shared and durable subscription* - The subscription can have many consumers. If all of them are offline, then the messages during the offline time are received by one of the consumers that comes online.

Kafka Destination Configuration Properties

Kafka Channel Destination Configuration Properties

Field	Global Variable	Description
Topic Name	Yes	Name of the Kafka topic.
Schema Registry Platform	Yes	Select the Schema Registry that you want to connect to. You can use the Confluent Schema Registry or the TIBCO Schema Registry. In the **Serializer/Deserializer** field, to use the schema registry, you must select the KafkaAvroSerializer for Avro messages and KafkaProtobufSerializer for Protobuf messages. See Confluent documentation for more information about the concepts and installation of the Confluent Schema Registry. See TIBCO® Messaging - Schema Repository for Apache Kafka - Enterprise Edition documentation for more information about the concepts and installation of the TIBCO Schema Registry.
Schema Registry URL	Yes	The REST API URL of the schema registry. The URL must be in the following format: `http://<hostname>:<port>`
Schema Subject	Yes	Subject name of Schema Registry. See Subject Naming

Field	Global Variable	Description
		Strategies for more information about subject name strategies.
Schema Registry Authentication	Yes	Authentication method for the Schema Registry: • Basic Auth: This authentication method uses a valid username and password to establish a secure communication with the schema registry. • One-Way SSL: Establishes an encrypted connection to the schema registry using a truststore file and truststore password. • Mutual Auth: Establishes an encrypted connection to the schema registry using truststore file, truststore password, keystore file, and keystore password. • No Auth Click **Configure Auth** to configure authentication properties for the Schema Registry. See Kafka Schema Registry Security Configuration Properties for more information about authentication properties. **Note:** For TIBCO Schema registry, One Way SSL and No Authentication are only supported.
Update Schema in Schema Registry	Yes	If this checkbox is selected, TIBCO BusinessEvents creates the schema by using event properties and payload associated with the destination and registers it as per the subject name strategy. If there are any changes in event properties or concepts, you can select this checkbox to update the schema in the schema registry.
Schema	No	Specify the schema to be registered in the Schema Registry. Supports both Avro and Protobuf schema.
Group ID	Yes	A unique ID that identifies the consumer group that this

Field	Global Variable	Description
		consumer belongs to. Assign the same **Group ID** to the consumers that you need to logically group in the same consumer group.
Client ID	Yes	A unique ID for the consumer to identify the source of the request. If not specified, BusinessEvents auto generates a unique **Client ID**. The best practice is to leave it blank, so that BusinessEvents generates the unique ID automatically.
Consumer Threads	Yes	Number of Kafka consumer threads that BusinessEvents creates for the destination. You need to define multiple partitions for the Kafka topic so that each consumer thread is able to load messages. One partition can be assigned to one consumer thread at a time. To add more partitions to the topic, use a `kafka-topics` script. `kafka-topics.sh --zookeeper <zk_host:port> --age --topic <your_topic_name> --partitions <new_partition_count>`
Heartbeat Interval (msec)	Yes	Specifies the time interval in milliseconds in which the destination sends heartbeat messages to the broker. Heartbeats are used to ensure that the worker's session stays active and to facilitate rebalancing when new members join or leave the group. The value must be set lower than **Session Timeout**. The best practice is to set it to one-third or lower of the **Session Timeout** value.
Session Timeout	Yes	Specifies the time interval in milliseconds, which is used to

Field	Global Variable	Description
(msec)		detect worker failures. The worker sends the heartbeat messages in **Heartbeat Interval** to indicate their liveliness to the broker. If no heartbeat messages are received by the broker before the expiration of the **Session Timeout**, then the broker removes the worker from the group and initiates a rebalance.
Enable AutoCommit	Yes	Specifies if the offsets are auto committed or not. If selected (default), the consumer's offset is committed automatically in the time interval specified in **AutoCommit Interval**. If cleared, the message is acknowledged when the RTC cycle completes or when `Event.consumeEvent` is called. This works only with the "Caller's Thread" threading model type. When Cache is enabled, the Event is acknowledged after the RTC cycle completes. By default, `enableParallelOps` is enabled so that the Event is committed in a separate thread. Because of this, the worker thread loads and processes the same Event a second time. To avoid that the message is processed twice, acknowledge the Event in a Rule or RuleFunction with `Event.acknowledgeEvent (<event>)`. Alternatively, Disable parallelOps for the InferenceAgent. To disable it, add the following property in the CDD file: Property: `Agent.inference-class.enableParallelOps` Value: false
AutoCommit Interval (msec)	Yes	When **Enable AutoCommit** is checked, this field identifies the frequency in milliseconds that the consumer offsets are auto-acknowledged to Kafka.

Field	Global Variable	Description
Sync Sender	Yes	When checked, BusinessEvents waits for the message to be actually sent, instead of relying on the Kafka client library to send it asynchronously and assuming sent.
Sync Sender Max Wait (msec)	Yes	When **Sync Sender** is checked, this field identifies the maximum duration in milliseconds to wait for a send call to complete. If the client fails to send a message within this limit, an exception occurs and sendEvent() returns NULL.
Max Poll Interval (msec)	Yes	Specifies the maximum delay in milliseconds before invoking the poll(). If poll() is not called before expiration of this timeout, then the consumer is considered failed.
Include Kafka Header	Yes	If the checkbox is selected, then the event property is included in the Kafka Header and not consumed as a Kafka message. If the checkbox is not selected, then the event property is not included in the Kafka Header and is consumed as a Kafka message. **Note:** When you add a property into the Kafka header, convert an Object to Byte array. Similarly, when extracting a property from Kafka header, convert from Byte array to an Object.
Compression Type	Yes	Specifies the compression type for messages being sent for the destination. The valid values are: • GZIP • Snappy • LZ4 • ZSTD • None (default)

Field	Global Variable	Description
Send Message Key (RuleFunction)	Yes	This determines the Kafka partition to which the message is routed. Messages with the same keys are routed to the same partitions. Browse and select the RuleFunction that is used to compute the keys for the messages that are sent using this destination. The RuleFunction must have the expected Event in scope as the only parameter, and return a String value for the message key. When a RuleFunction URI is not specified, BusinessEvents sends a message without a key. In such a case, the message is routed to a partition as determined by Kafka.

Kafka Schema Registry Security Configuration Properties

When you select a **Schema Registry Authentication** method and click **Configure Auth**, the following fields are displayed.

Field	Global Variable	Description
FTL Realm Server	Yes	TIBCO FTL server URL. This field is enabled only for the TIBCO Schema registry.
FTL Username	Yes	Username to access TIBCO FTL server.
FTL Password	Yes	Password to access TIBCO FTL server.
Trusted Certificate File	No	Absolute path and file name of the SSL truststore file that contains the SSL certificate to connect to the schema registry.
Identity	No	Path and file name of the identity store.
Trust Store Password	Yes	Password for the truststore.

Pulsar Destination Configuration Properties

Field	Global Variable	Description
Topic Name	Yes	Name of the Pulsar topic.
Client ID	Yes	The client id to be used while subscribing to the Pulsar topic
Subscription Initial Position	Yes	The initial position of subscription. Select Latest to receive the messages that have been published after the subscriber has been connected. Select Earliest to receive all the stored and new messages. • Earliest: • Latest Default: Earliest
Subscription Type	Yes	Specifies the type of Subscription. • Exclusive: In Exclusive mode, only a single consumer is allowed to attach to the subscription. An error occurs if multiple consumers subscribe to a topic by using the same subscription. • Shared (default): In Shared mode, multiple consumers can attach to the same subscription. Messages are delivered in a round robin distribution across consumers, and any given message is delivered to only one consumer.

Field	Global Variable	Description
		• Failover: In Failover mode, multiple consumers can attach to the same subscription. When the master consumer disconnects, all (non-acknowledged and subsequent) messages are delivered to the next consumer in line. • Key_Shared: In KeyShared mode, multiple consumers can attach to the same subscription. Messages are delivered in a distribution across consumers. Messages with the same key or the same ordering key are delivered to only one consumer.
Is Source Cassandra	Yes	Set this checkbox to true when you want to use Cassandra as a channel and want to receive events from Cassandra.
Sync Sender	Yes	When checked, BusinessEvents waits for the message to be actually sent, instead of relying on the Pulsar client library to send it asynchronously and assuming sent. Default: False
Compression Type	Yes	Specifies the compression type for messages being sent using the particular destination. Valid values are None, LZ4, ZLIB, ZSTD, Snappy. Default: None (no compression)
Schema	No	Specify the schema to be registered in the Schema Registry. Only the Avro schema type is supported.
PulsarWebServiceURL		Specify the Pulsar web service URL for your broker cluster.

Field	Global Variable	Description
Update Schema in Schema Registry	Yes	If there are any changes in event properties or concepts, you can select this checkbox to update the schema in the schema registry.
Send Message Key (RuleFunction)	Yes	This determines the Pulsar partition to which the message is routed. Messages with the same keys are routed to the same partitions. Select the RuleFunction that is used to compute the keys for the messages that are sent using this destination. The RuleFunction must have the expected Event in scope as the only parameter, and return a String value for the message key. When a RuleFunction URI is not specified, BusinessEvents sends a message without the key. In this case, the message is routed to a partition as determined by Pulsar.

Local Destination Configuration Properties

Local destinations do not use serializers, deserializers, or default events.

Local Destinations Configuration Properties

Field	Global Variable	Description
Size	No	The maximum number of events to be held in the queue. The default is zero (0), which allows unlimited events in the queue.
TimeOut	No	Time to wait when sending an event to this local destination. The values are: • `-1` (Default) - Waits indefinitely • `0` - Does not wait • `>0` - Waits for the specified number of milliseconds

ActiveSpaces Channel

TIBCO ActiveSpaces® software is a distributed in-memory data grid product. It can notify applications like TIBCO BusinessEvents of changes to the rows stored in a table that can be transformed into TIBCO BusinessEvents events. For more information, see the *TIBCO ActiveSpaces documentation*.

TIBCO BusinessEvents integrates with TIBCO ActiveSpaces in various other methods as well. To know how the ActiveSpaces Channel is different in comparison to the other methods of integration, see TIBCO ActiveSpaces Support.

Catalog Functions

ActiveSpaces channel supports the sending of simple events using a `Event.sendEvent()` catalog function.

ActiveSpaces Channel Serializers

ActiveSpaces channel provides the following serializer to handle payloads:

AS3xSerializer

The AS3Serializer serializer (`com.tibco.cep.driver.as3.AS3Serializer`) is used to deserialize TIBCO ActiveSpaces row to a simple event in TIBCO BusinessEvents and to serialize an event in TIBCO BusinessEvents to TIBCO ActiveSpaces row.

ActiveSpaces Channel Example

A sample TIBCO BusinessEvents application that uses the ActiveSpaces channel is available under *BE_HOME*`/examples/standard/ActiveSpacesChannel`.

Adding an ActiveSpaces Channel in a TIBCO BusinessEvents Application

Configure channel properties and destination properties for the ActiveSpaces channel to add the channel successfully to the TIBCO BusinessEvents application.

Before you begin

- You must have installed TIBCO FTL [1]
- You must have installed TIBCO ActiveSpaces

Procedure

1. In TIBCO BusinessEvents Studio, add a channel with **Driver Type** as ActiveSpaces.

   See Adding a Channel for more details.

2. You can configure the ActiveSpaces channel by using a shared resource or by configuring the channel properties. See ActiveSpaces Connection Reference for using a shared resource. Otherwise, in the channel editor, enter the ActiveSpaces channel configuration properties and select **Method of Configuration** as Properties.

   See ActiveSpaces Channel Configuration Properties for more details on the ActiveSpaces channel configuration properties.

3. In the channel editor, add a destination for the ActiveSpaces channel for listening to the messages from the ActiveSpaces channel.

   See Adding a Destination to a Channel for information on how to do it. Also, see ActiveSpaces Destination Configuration Properties for more details on the configuration properties for the ActiveSpaces channel destination.

4. Save the new ActiveSpaces channel.

[1] 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.

Legacy ActiveSpaces Channels

This chapter provides additional information about working with the Legacy ActiveSpaces channel.

This section provides a brief introduction to the Legacy ActiveSpaces concepts and terminology that are useful when working with the Legacy ActiveSpaces channel. See the *TIBCO ActiveSpaces* documentation for more information.

TIBCO BusinessEvents integrates with TIBCO ActiveSpaces in various other methods as well. To know how the Legacy ActiveSpaces Channel is different in comparison to the other methods of integration, see TIBCO ActiveSpaces Support.

Metaspace

A *metaspace* is a virtual entity that contains spaces, which are containers that store the data used by applications.

A metaspace is a container for managing system spaces, user spaces, and a group of members that are working together in a cluster. The metaspace is the initial handle to Legacy ActiveSpaces. An application or member first joins a metaspace, and through it, gets access to other objects and functionality.

A metaspace is created when the first process connects to it, and disappears when the last process disconnects from it. The metaspace grows or shrinks automatically as members connect to it and disconnect from it.

Initially, a metaspace contains only system spaces. As users create spaces in the metaspace, the definition of those spaces (along with other administrative data) is stored in system spaces.

Multiple metaspaces may exist at the same time, each one containing a completely independent set of spaces. This means, for example, that changes to a space called **clients** in a metaspace named **Dev** have no impact on a space named **clients** in a metaspace named **Prod**. Since no single application can connect to two different metaspaces using the same metaspace name, metaspaces should always use different names.

Space

A space provides shared virtual storage for data. A space is:

- A container for a collection of entries that consist of a tuple and associated metadata.
- A *shared* entity. Many applications can access a space concurrently and each application has the same coherent view of the data contained in the space. The spaces in Legacy ActiveSpaces are called tuple spaces, and the items stored in them are called tuples.

  A space can proactively notify applications of changes in the data contained in the space as changes happen (push model), and can therefore be used as a coordination mechanism for building distributed systems.
- A *virtual* entity that is distributed and implemented collaboratively by a group of processes located on multiple hosts and communicating over the network.

Legacy ActiveSpaces handles changes in this set of processes automatically: processes might join or leave the group at any time without requiring any user intervention. A space automatically scales up as the number of processes in the group increases, and scales down when processes suddenly disappear from the group or network. There is no negative impact on the data contained in the space when processes leave the space.

System spaces are a set of administrative spaces that are created and maintained by Legacy ActiveSpaces and are used to describe the attributes of the spaces.

User spaces are spaces that are defined by the user.

Members and Member Roles

Entities that need access to a space join the space as *members*.

Members can join a space as a seeder or as a leech:

- *Seeders* play an active role in maintaining the space by providing CPU and RAM.
- *Leeches* play a passive role. They have access to space data but provide no resources.

Seeders

Seeder applications join the space and indicate their willingness to lend some of the resources of the host where they are deployed to scale the service provided by Legacy ActiveSpaces. In effect, the seeding applications have a portion of the Space embedded in their process.

Legacy ActiveSpaces distributes the data stored in the space evenly between all of the processes that have joined the space as seeders. Legacy ActiveSpaces is an elastic distributed system. Seeders can join and leave the space (effectively scaling it up or down) at any time without the need to restart or reconfigure any other participant in the space. When this happens, the distribution of entries is automatically rebalanced if necessary to maintain even distribution between the seeders.

Leeches

An application can also join a space as a *leech*, without contributing any of its host's resources to the scalability of the space, and without experiencing any limitation in functionality or in its ability to use the space. A leech (as opposed to a seeder) is a peer of the space that is not considered by the distribution algorithm and therefore can join or leave the space without causing redistribution of the data.

Legacy ActiveSpaces Serializer

The default serializer used to deserialize Legacy ActiveSpaces tuple to a simple event in TIBCO BusinessEvents and to serialize an event in TIBCO BusinessEvents to an Legacy ActiveSpaces tuple. You must configure the serializer (`com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer`) for every destination created on the Legacy ActiveSpaces channel.

Distribution role

The distribution role (Seeders or Leeches) is a level of participation of a space member and does not indicate any limitation on use. Leeches have access to the same set of space operations as seeders. The choice of distribution role must be made on a per space basis: the best solution might be to join some spaces as a seeder and others as a leech.

Consumption mode

Legacy ActiveSpaces can proactively notify applications of changes to the tuples stored in a space. You can choose one of the consumption modes: Event Listener, Entry Browser, or Router.

When the consumption mode of a destination is *event listener*, the destination behaves like a *subscriber* in a publish-subscribe messaging system. When certain data changes or certain events occur, a callback function is invoked. You can choose the type of events to listen to: Put Event, Take Event, and Expire Event.

When the consumption mode is *entry browser*, the destination can monitor the space for data changes or certain events to occur, and can retrieve the tuple from the space using the *browser type* - Get, Take, or Lock & Take.

When the consumption mode is *router*, the destination feeds the events to only the specified listener. This way, the application benefits from having multiple listener processes running.

Note: The `DistributionScope` as well as the `TimeScope` properties now have options enabled that can be specified for both EventListener and EntryBrowser.

Tuple

A *tuple* is similar to a row in a database table: it is a container for data. Specifically, it is a sequence of named elements called fields (similar to the columns in a database table) that contain values of a specific type. Each tuple in a space represents a set of related data.

Fields have a name and a type. A tuple can be seen as a kind of map on which fields can be 'put' or 'removed'. A tuple can also be seen as a self-describing message. Tuples are platform independent and can be serialized and deserialized.

Filters

Filters can be applied to both listeners and browsers, and also be used to evaluate a tuple against a filter. Filters allow your application to further refine the set of tuples it wants to work with using a space browser or event listener.

A filter string can be seen as what would follow the *where* clause in a `select * from Space where…` statement.

Examples

```
   field1 < (field2+field3)   state = "CA"
   name LIKE ".*John.*" //any name with John
```

Filters can make reference to any of the fields contained in the tuples. Filters do not provide any ordering or sorting of the entries stored in the space.

Operators Supported in Filters

The following table shows the operators that are supported in the Legacy ActiveSpaces filters:

Operators for Legacy ActiveSpaces Filters

Operator	Meaning
`>, >=`	`greater than`
`NOT or ! or <>`	`not`
`*`	`multiply`
`=`	`equal`
`!=`	`not equal`

Operator	Meaning
`ABS`	`absolute value`
`MOD or %`	`modulo, as in ( MOD (x,y) or x % y)`
`NOT BETWEEN`	`not between`
`BETWEEN`	`between`
`||`	`string concatenation`
`NOT LIKE`	`not like (regex)`
`LIKE`	`like (regex)`
`<, <=`	`less than`
`+`	`addition`
`OR`	`or`
`-`	`subtraction`

Operator	Meaning
AND	and
IN	range, as in “age in (1,3,5,7,11)"
NULL	does not exist
NOT NULL	exists
IS	only used with NULL or NOT NULL, as in “x IS NULL" or "x IS NOT NULL"
NOR	nor, as in “age NOT 30 NOR 40“
/*	comments
//	comments

Formats for Filter Values

The following table shows the formats for values used in filters.

Formats for Filter Values

Filter Value	Format
`octal value`	`\oXXX`
`hexadecimal value`	`\xXXX`
`exponents (as in 1E10 or 1E-10)`	`XXXEYY`
`date time`	`YYYY-MM-DDTHH:MM:SS`
`date`	`YYYY-MM-DD`
`time`	`HH:MM:SS:uuuu+/-XXXXGMT`
`true`	`TRUE`
`false`	`FALSE`

Working with Legacy ActiveSpaces Channels

In order to use the channel to monitor a space in Legacy ActiveSpaces, you need to add and configure one or more destinations for the channel.

Creating a Destination and Event for Legacy ActiveSpaces Channel Using the Wizard

TIBCO Channel Wizards provide an option to create a destination and event for a specific Legacy ActiveSpaces channel.

Note: The wizard requires the metaspace and spaces in Legacy ActiveSpaces to be created and initialized. Otherwise, no spaces are listed on the Select space dialog. You must create and initialize the metaspace and then click **Refresh** to refresh the list of spaces that the channel can connect to.

Procedure

1. In BusinessEvents Studio Explorer, select the Legacy ActiveSpaces channel for which you need to create a destination and event.
2. Go to **File > New > Other > TIBCO Channel Wizards > Legacy ActiveSpaces Destination and Event** and click **Next**.

   The Select Space dialog lists the available spaces for every Legacy ActiveSpaces channel in the project.

   Note: If no spaces are listed on the Select space dialog, check if the metaspace is created and initialized. Create and initialize the metaspace, and then click **Refresh** to refresh the list of spaces that the channel can connect to.
3. Select a space and click **Next**.

4. On the New SimpleEvent and Destination dialog, enter the values to configure the simple event and destination for the selected channel and space.

Configuring an Legacy ActiveSpaces Destination and Event

Field Name	Description
Simple Event	
Name	Name of the simple event to be created. The default name is of the format <*spaceName*>Event.
Destination	
Name	The name of the destination to be created. The default name on the screen is of the format <*spaceName*>Destination.
Default Event	The default event for the destination. This is set to the SimpleEvent specified to be created by the wizard.
Serializer/Deserializer	The serializer used to map tuples in Legacy ActiveSpaces to simple events in TIBCO BusinessEvents. The default value is `com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer`. See Legacy ActiveSpaces Serializer for more information.
Space Name	The name of the space that the destination connects to. This space is selected in the Select space dialog. See Space for more information about spaces in Legacy ActiveSpaces.
Distribution Role	The level of participation of the space member: seeder or leech. See Distribution role for more information.

Field Name	Description
Filter	String specified to evaluate tuples and refine the set of tuples to work. A filter string can be seen as what would follow the *where* clause in a `select * from Space where…` statement. See Filters for more information.
Consumption Mode	Specifies the consumption mode for the Legacy ActiveSpaces event as one of: • Event Listener - listens for specific events to occur, and invokes a callback function from Legacy ActiveSpaces. • Entry Browser - can listen and retrieve tuples from the space using the Get, Take, or Lock & Take methods. • Router - feeds the events to only the specified listener. This way, the application benefits from having multiple listener processes running. See Consumption mode for more information.
Browser Type	Available only when the consumption mode is Entry Browser. Specifies the browser type used to retrieve tuples from a space. You can choose either Get, Take, or Lock & Take. The difference between the Get and Take browsers is that Get retrieves a copy of the tuple from the space and Take retrieves the tuple from the space, and there is no trace of the tuple in the space after the Take event. Using the Lock & Take option, you can acquire a lock on the tuple key. The event is consumed or taken in post-RTC ensuring successful processing of the event. The Take action does not occur if abortRtc() is called in the preprocessor or rules.

Field Name	Description
	Note: The lock owner is the process. Make sure that the Lock Scope is defined as a process in the Legacy ActiveSpaces space definition.
DistributionScope	The distribution scope for the Legacy ActiveSpaces event. This field is active only when the **Consumption Mode** is Event Listener or Entry Browser. When **Consumption Mode** is Entry Browser, the field specifies the possible ways to browse for events in a Space based on where entries are stored in the space. The values are: • ALL - The browser iterates through all the events in a space regardless of where the entries for which those events occurred are stored. • SEEDED - The browser only iterate through the events in the space that occurred for entries stored on the member. The values when **Consumption Mode** is Event Listener are: • ALL - The listener listens to an event related to all the entries in space. • SEEDED - The listener listens only to events associated with the entries in Space that are seeded by the member (empty unless the member is a seeder on the space).
TimeScope	The time scope for the Legacy ActiveSpaces event. This field is active only when the **Consumption Mode** is Event Listener or Entry Browser. When **Consumption Mode** is Entry Browser, the field specifies the possible ways to browse for entries in a Space based on the time at which entries were stored in the space relative to the

Field Name	Description
	Browser creation time. The values are: • ALL - The browser iterates through all the entries stored in the space at creation time and continues to iterate as changes occur in the space. • NEW - The browser skips iterating through the entries stored in the space at creation time and only iterates through new changes that occur in the space. • SNAPSHOT - The browser only iterates through entries stored in the space at creation time. The values when **Consumption Mode** is Event Listener are: • ALL - The listener starts with all the tuples currently in the space at creation time, which is presented as an initial site of PUT events and then is continuously updated according to changes in the space. • NEW - The listener starts empty and is updated only with events related to new or updated tuples in the space. • SNAPSHOT - The listener contains only PUT events corresponding to the tuples stored in the space at creation time. • NEW_EVENTS - The listener starts empty and is updated with all the events that occur in the space after creation time.
Put Event	Available only when the consumption mode is Event Listener. When selected, the Event Listener listens for any Put events on the space and invokes a callback function when such an event occurs.
Take Event	Available only when the consumption mode is Event Listener.

Field Name	Description
	When selected, the Event Listener listens for any Take events on the space and invokes a callback function when such an event occurs.
Expire Event	Available only when the consumption mode is Event Listener. When selected, the Event Listener listens for any Expire events on the space and invokes a callback function when such an event occurs.
Prefetch	Available only when the consumption mode is Entry Browser. Set the value in the field to attain optimum performance. The default value is -1 (prefetch all).

5. Click **Finish** to create the destination and the simple event.

Result

The wizard adds the following properties to the simple event:

- **id** of type long
- **consumption_mode** of type string
- **browser_type** of type string
- **event_type** of type string

Configuring the Destination and Default Event Manually

You can add a destination to the Legacy ActiveSpaces channel manually, without using the TIBCO Channel wizards.

The following procedure describes the steps to add and configure a destination for the Legacy ActiveSpaces channel.

Procedure

1. In BusinessEvents Studio Explorer view, double-click the channel name to open the channel editor, if it is not already open.
2. In the Destinations section, click **Add**.
3. Enter a Name and Description for the destination.
4. In the **Default Event** field, browse to and select the event to be created (by default) from incoming messages received by this destination.

   If you have not yet created the event, you can select the default event later. See Assigning a Default Event to the Destination for details about the simple event.
5. Select the serializer/deserializer to be used: `com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer`.
6. Enter the values for the rest of the fields as described in Creating a Destination and Event for Legacy ActiveSpaces Channel Using the Wizard .

Assigning a Default Event to the Destination

When creating a simple event manually, ensure that you add the property `id` of type long.

Procedure

1. To assign a default event to a destination, open the channel editor in BusinessEvents Studio Explorer view.
2. In the **Default Event** field, browse to and select the simple event created earlier.

Configuring Events to Work with Legacy ActiveSpaces Channel

You can configure events to interact with external Legacy ActiveSpaces metaspace.

Event properties are mapped to Legacy ActiveSpaces tuple fields and are utilized to place and receive tuples to and from Legacy ActiveSpaces metaspaces.

For incoming events on Legacy ActiveSpaces destinations, event property values are set with the values coming from the Legacy ActiveSpaces tuple.

For outgoing events sent on Legacy ActiveSpaces destinations, event properties are copied into the Legacy ActiveSpaces tuple.

There is a special case to set and receive null tuple fields since null cannot be handled by event properties. Event payload is used instead of event properties.

Each child of the root element is mapped to a tuple field. For incoming events, if an element of the payload matches the name of a field in the Legacy ActiveSpaces space, that element is set with the value from the tuple. If the value of the field in the tuple is null, the attribute `xsi:nil` is set to true for that element.

For outgoing events, if the name of the element matches the name of a field in the Legacy ActiveSpaces space, that field is set with the value of the element. If the attribute `xsi:nil=true` is set in the element, the value of the field in the tuple is set with null.

Configuring Security and Authentication For Legacy ActiveSpaces Channel

You can configure security for the Legacy ActiveSpaces channel and use LDAP or system based authentication.

Procedure

1. Open the Legacy ActiveSpaces channel file in the channel editor.
2. Select Properties or Resources as **Method of Configuration**.

   The channel properties are displayed.

3. If Properties is selected, then enter the values for the following security fields (see Legacy ActiveSpaces Channel Configuration Properties for more details about the properties).
   - **EnableSecurity** (activates the other security fields)
   - **SecurityRole**
   - **Identity Password**
   - **PolicyFile**
   - **TokenFile**
   - **Credential**
   - **Domain**
   - **Username**
   - **Password**
   - **KeyFile**
   - **PrivateKey**

   Note: Some fields are activated only for the specific security role or authorization policy.

4. If Resources is selected, then configure the Legacy ActiveSpaces connection resource for the security. The security fields are the same as mentioned in Step 3. See Wizard and Configuration Tab for more details about the properties.

Custom Channel

If you want to use a channel other than the channels provided with TIBCO BusinessEvents, use the custom channel API to create the custom channel.

The Java API for custom channel (`com.tibco.be.custom.channel` package) is bundled with TIBCO BusinessEvents. The `com.tibco.be.custom.channel` package consists of interfaces and classes that you can implement and extend to create a custom channel. Implement these classes to initialize the channel, create serializer and deserializer for the custom channel, to store and access event data, and to perform various other operations. For more information about the classes and interfaces in the `com.tibco.be.custom.channel` package see *Java API Reference*.

TIBCO BusinessEvents bundles a sample Kafka channel and its source code to showcase the channel API usage. The sample Kafka channel implementation of custom channel API is located at `*BE_HOME*\api\channel-api\examples\StudioProjKafka` and the source code is located at `*BE_HOME*\api\channel-api\examples\kafka\src`.

Custom Channel Lifecycle

By using the Java API, you can create your own custom channel according to your requirement. While implementing a custom channel, you can extend custom channel classes and override custom channel methods that BusinessEvents invokes during startup.

Refer to *Java API Reference* for complete list of Java classes and interfaces for the custom channel. However, few of the key Java classes, interfaces and their key components are mentioned in the following sections identifying the channel lifecycle in BusinessEvents.

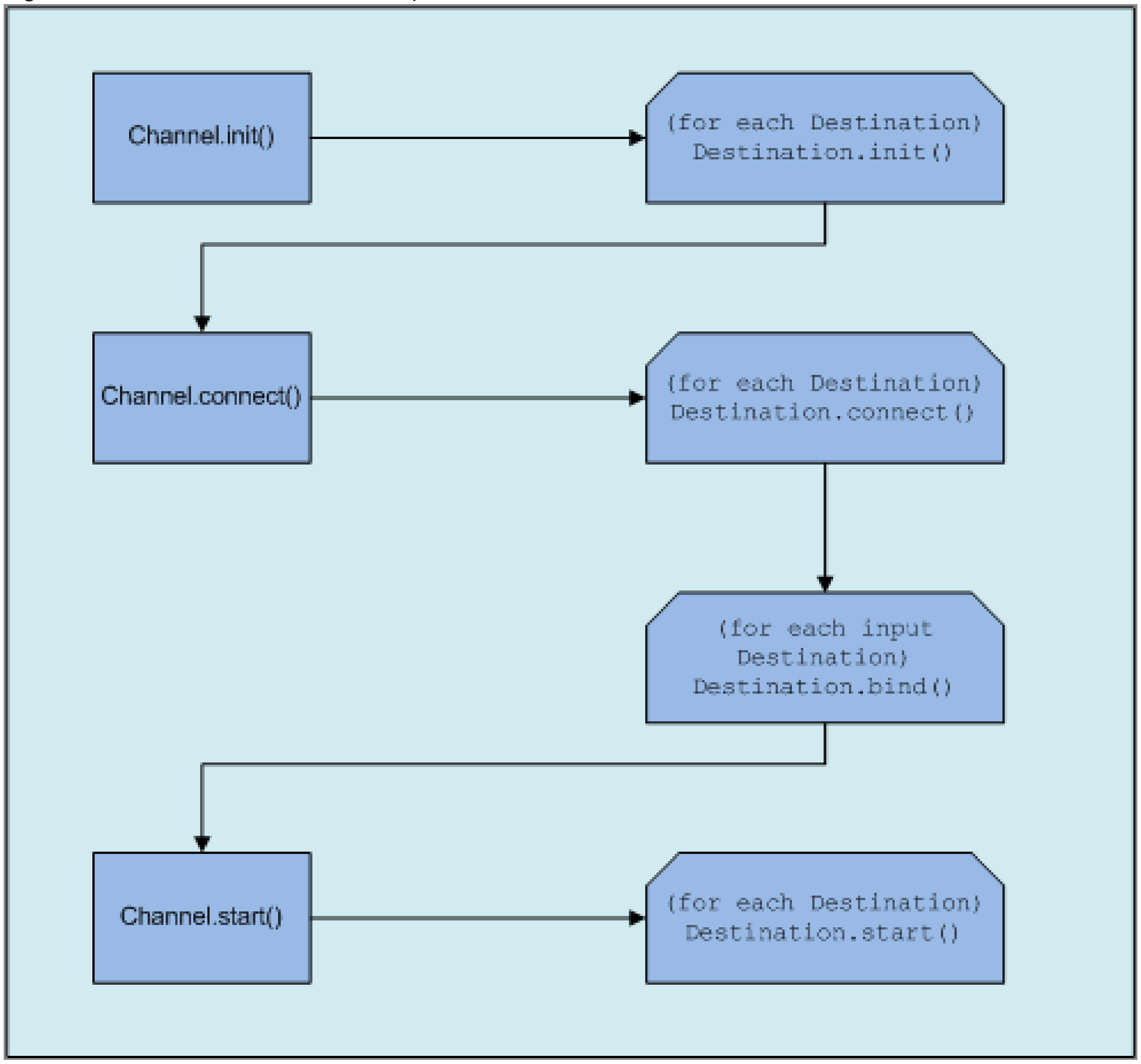
Channel and Destination Class Instantiation

Channel and destination class instances are created during the engine startup by using the information provided in the `drivers.xml` file and in the project. For each of your custom channel defined in the project, BusinessEvents creates one instance of the driver class as defined in `drivers.xml` and calls the driver class's `getChannel` method. Typically, the `getChannel` method must return a new channel instance of your channel class.

Similarly, for each of the destination of each of your custom channel found in the project, BusinessEvents calls `Driver.getDestination` method. Typically, the `getDestination` method return a new instance of your destination class.

During startup, BusinessEvents invokes the various lifecycle methods of your custom channel in the sequence specified in the following figure.

Figure 5: Channel Initialization Sequence


Channel.init()
(for each Destination) Destination.init()
Channel.connect()
(for each Destination) Destination.connect()
(for each input Destination) Destination.bind()
Channel.start()
(for each Destination) Destination.start()

Channel.init

Initialize your channel in this method. Extend the `BaseChannel` class to create your custom channel class and then you can override the `init` method to perform initialization of your channel. During initialization, your custom channel can access the channel properties that you have defined in `drivers.xml` using the `getChannelProperties` method and access the runtime TIBCO BusinessEvents properties using the `getBeProperties` method. For example, when you override the `Channel.init`method, you can set up initialization objects such as thread pools for internal use, connection objects for connecting to your channel endpoints, and so on.

If you override the `init` method, you must initialize each destination manually, or you call the `super.init` method. The `super.init` method initializes each of the channel's destinations by calling the `BaseDestination.init` method for each destination of the channel.

Destination.init

Extend the `BaseDestination` class to create your custom destination class. In case, the destination need initialization, override the `init` method to initialize your destination. You can access destination properties by using the `getDestinationProperties` method. You can access the parent channel instance by using the `getChannel` method and in turn channel and TIBCO BusinessEvents properties from the channel instance.

Channel.connect

For connection based transports, such as JMS or WebSphere MQ, override the `connect` method to initiate the connection to the underlying data provider or data source. By default, the `BaseChannel` implementation calls the `BaseDestination.connect` method for each of your destinations. In case your destination does not have a connection semantic, you can simply implement an empty method.

If you override the `connect` method, you must initiate connection for each destination manually, or you call the `super.connect` method. The `super.connect` method initiates connection for each of the channel's destinations by calling the `BaseDestination.connect` method for each destination of the channel.

Destination.bind

The next step in the TIBCO BusinessEvents startup sequence is the `BaseDestination.bind` method. Whenever an agent becomes *active*, TIBCO BusinessEvents calls the `BaseDestination.bind` method for each of the destination defined in CDD under **Input Destinations Collections** of the agent. The `bind` method binds the underlying RuleSession to a destination, that is, whenever a message arrives on that destination endpoint, the message is routed to the RuleSession associated with the destination. Thus, the message triggers the rules associated with its bounded RuleSession.

In the `bind` method, you must start your destination's message listeners and store or bind the EventProcessor instance, that is handed out by the framework to this listener. When a message arrives at this listener's endpoint, you must convert this message to an *event* by using the serializer for your destination, and then send that event to the TIBCO BusinessEvents framework by calling the `EventProcessor.processEvent` method.

Channel.start

The next step during TIBCO BusinessEvents startup sequence is the `BaseChannel.start` method. For each of the channels, TIBCO BusinessEvents calls the `start` method. By default, the `BaseChannel.start` method calls the `BaseDestination.start` method for each destination. Use this call to start dispatching messages from the underlying transport.

If you override the `start` method, you can call the `super.start` method to start each of the channel's destinations by calling the `BaseDestination.start` method.

If you override the `start` method, you must start each destination manually, or you call the `super.start` method. The `super.start` method starts each of the channel's destinations by calling the `BaseDestination.start` method for each destination of the channel.

Channel.suspend

TIBCO BusinessEvents invokes the `BaseChannel.suspend` method when you need to suspend the channel. Provide a meaningful implementation in this method to suspend receiving of messages for your destination.

If you override the `suspend` method, you need to suspend each destination manually, or you can call the `super.suspend` method. The `super.suspend` method suspends each of the channel's destinations by calling the `BaseDestination.suspend` method for each destination of the channel.

Channel.resume

TIBCO BusinessEvents invokes the `BaseChannel.resume` method when you need to resume the suspended channel. Provide a meaningful implementation in this method to resume receiving of messages for your destination.

If you override the `resume` method, you need to resume each destination manually, or you can call the `super.resume` method. The `super.resume` method resumes each of the channel's destinations by calling the `BaseDestination.resume` method for each destination of the channel.

Channel Messages to Event Conversion

The overall function of your channel is to receive messages from your data source, convert them to Event instances and call `EventProcess.processEvent` to hand off the events to BusinessEvents for further processing.

In order to serialize your messages to BusinessEvents events, you must use the serializer that is configured in your destination. You can extend the `BaseEventSerializer` class to create a serializer for your channel. Also, you need to implement the `Event` interface which holds the underlying event related data.

The serializer is available using the `getDestination` method of your destination. As part of converting your channel's messages to events, you can perform the following implementation in the `deserializeUserEvent` method:

- Create a new `Event` instance and populate event fields with `Event.setProperty`.
- Set the event's URI using `Event.setEventUri` and set it to the project's event type, for example, `/Events/MyEvent`.
- (Optional) Set an XML String payload using the `Event.setPayload` method.
- (Optional) Set the Event's extID using the `Event.setExtId` method.

Your event must return all those values that are set using corresponding getter methods. In particular, your event must properly return a list of all properties that are set using `getAllPropertyNames`. The framework uses various getter methods to create and populate a BusinessEvents event type as defined by `setEventUri`.

Catalog Function Implementation for Custom Channel

Your channel code must implement certain methods to provide proper semantics to the `Event` catalog functions.

Event.send

Override the `Destination.send` method and provide an implementation to send to your destination endpoint. This method is invoked when the BusinessEvents application code invokes `Event.send`.

Event.requestEvent

Override the `Destination.requestEvent` method and provide an implementation to send synchronously a message request to your destination endpoint. The implementation should return a response event to the caller. If your messaging system does not support the request-reply semantic, you can choose to generate an unsupported operation exception.

Event.routeTo

Override `Destination.send` and provide an implementation to route to your specified destination endpoint.

EventContext

An `EventContext` object holds the context of the incoming event. Your `Destination.getEventContext` method must return an implementation of your `EventContext`. In your event context, you can keep any state, such as, the inbound message received in your listener callback; a corresponding listener/session handles. Typically you would keep as much state in this object as required for providing proper behavior for the `EventContext.reply` and `EventContext.acknowledge` methods.

Event.replyTo

This catalog function maps to the `EventContext.reply` method. Override the `EventContext.reply` method as applicable for your project for a reply event semantics.

Event.consumeEvent

This catalog function maps to your `EventContext.acknowledge` method. For some messaging systems, you might need to indicate to your message broker to acknowledge delivery of the inbound message, so that the message is not redelivered. Override the `EventContext.acknowledge` method as applicable for your project.

Creating a New Custom TIBCO BusinessEvents Channel

You can use the custom channel API to create a custom channel according to your requirement for your project.
The Java API for a custom channel (`com.tibco.be.custom.channel` package) is bundled with TIBCO BusinessEvents. For more information about the classes in the `com.tibco.be.custom.channel` package, see *Java API Reference*.

Procedure

Define the drivers.xml file

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

2. In the XML file, define the drivers for the channel in the following format.

```
<?xml version="1.0" encoding="UTF-8"?>
<drivers>
    <driver>
        <type>CHANNEL_TYPE</type>
        <label>CHANNEL_LABEL</label>
        <class>DRIVER_CLASS </class>
        <description>CHANNEL_DESCRIPTION</description>
        <version>CHANNEL_VERSION</version>
        <properties>
            <!--Channel Properties -->
        </properties>
        <destinations>
            <!--Destination Properties -->
        </destinations>
        <serializers userdefined="true">
            <serializer type="SERIALIZER_TYPE" class="SERIALIZER_CLASS"/>
            <!--Additional Serializers -->
        </serializers>
    </driver>
</drivers>
```

Where,

- *CHANNEL_TYPE* - The channel type that you have defined. This is displayed in the BusinessEvents Studio under the **Driver Type** field in the New Channel Wizard.
- *CHANNEL_LABEL* - Display the name of the channel.
- *DRIVER_CLASS* - The URI of the BaseDriver class that you have implemented.
- *CHANNEL_DESCRIPTION* - Short description of the channel.
- *CHANNEL_VERSION* - Version number of the channel.
- *<!--Channel Properties -->* - Define the configuration properties for the channel. Specify the name, type, and default value of the properties. These properties are displayed as channel properties in BusinessEvents Studio.

- *<!--Destination Properties -->* - Define the configuration properties for the destination of the custom channel. Specify the name, type, and default value of the properties. These properties are displayed as destination properties in BusinessEvents Studio.
- *SERIALIZER_TYPE* - Type of the serializer for the channel. You can define multiple serializers.
- *SERIALIZER_CLASS* - Class of the serializer that you have defined. Specify at least one serializer for the channel. The class is listed in the BusinessEvents Studio under the **Serializer/Deserializer** field while adding a destination for the custom channel.

Create a JAR file

3. Create all the required Java class files using the Java API for a custom channel. Archive all the Java class files for the custom channel with the `drivers.xml` file as a JAR file.

   Refer to the Kafka channel example and its source code for the custom channel Java API usage. The Kafka channel example is at *BE_HOME*`\api\channel-api\examples\StudioProjKafka` and the source code is at *BE_HOME*`\api\channel-api\examples\kafka\src`. See the following topics for more understanding on how the custom channel API can be implemented to create the required Java files:

   - Custom Channel Lifecycle
   - Channel Messages to Event Conversion
   - Catalog Function Implementation for Custom Channel

   Refer to the *Java API Reference* for complete details about the Java API for a custom channel (`com.tibco.be.custom.channel` package).

Include the JAR file in BusinessEvents classpath

4. Copy this JAR file at *BE_HOME*`\lib\ext\tpcl\contrib` and restart BusinessEvents Studio.

   A new custom channel is displayed under the **Driver Type** in the New Channel Wizard.

Sample Kafka Channel Drivers.xml File

The following sample code is of the `drivers.xml` file for the Kafka channel example:

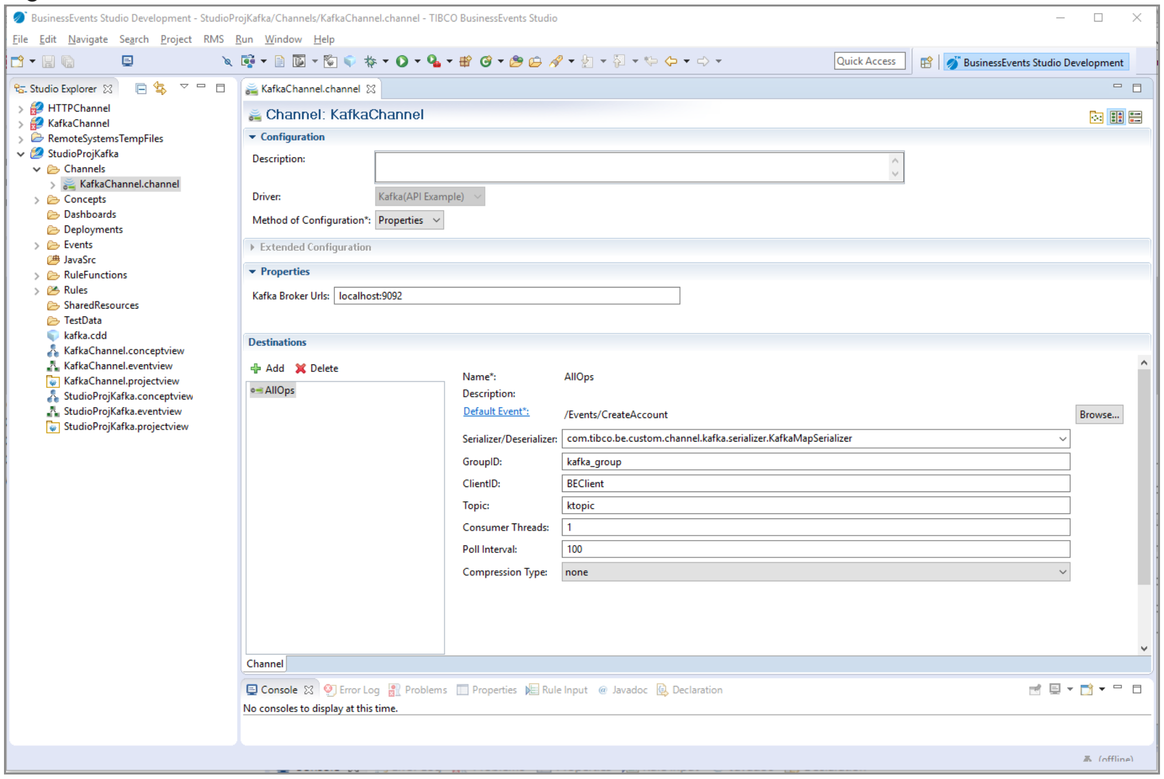
```
<?xml version="1.0" encoding="UTF-8"?>
<drivers>
       <driver>
        <type>Kafka(API Example)</type>
        <label>Kafka(API Example)</label>
        <class>com.tibco.be.custom.channel.kafka.KafkaDriver</class>
        <description>Apache Kafka is an open-source message broker
project developed by the Apache Software Foundation written in Scala.
The project aims to provide a unified, high-throughput, low-latency
platform for handling real-time data feeds.</description>
        <version>1.0.0.0</version>
        <properties>
            <property name="kafka.broker.urls" displayName="Kafka Broker
Urls" type="String" default="localhost:9092"/>
        </properties>
        <destinations>
               <property name="group.id" displayName="GroupID" type="String"
default="kafka_group"/>
            <property name="client.id" displayName="ClientID"
type="String" default="BEClient"/>
            <property name="topic" displayName="Topic" type="String"
default="ktopic"/>
            <property name="consumer.threads" displayName="Consumer
Threads" type="Integer" default="1"/>
            <property name="poll.interval" displayName="Poll Interval"
type="Integer" default="100"/>
            <property name="compression.type" displayName="Compression
Type" type="String" default="none"/>
        </destinations>
        <serializers userdefined="true">
            <serializer type="Map"
class="com.tibco.be.custom.channel.kafka.serializer.KafkaMapSerialize
r"/>
        </serializers>

        <!-- Define combo-boxes that are referred to in destination
properties. -->
        <configuration>
               <property>
                       <name>compression.type</name>
```

```
                        <parent>destination</parent>
                        <type>combo-box</type>
                        <choices>
                                <choice value="none" displayed="none"/>
                                <choice value="gzip" displayed="gzip"/>
                                <choice value="snappy" displayed="snappy"/>
                                <choice value="lz4" displayed="lz4"/>
                        </choices>
                </property>
        </configuration>
    </driver>
</drivers>
```

Sample Kafka Channel BusinessEvents Studio UI

The following screen shows the fields of the Kafka channel using a custom API in BusinessEvents Studio, according to the `drivers.xml` file.

Figure 6: Kafka Channel BusinessEvents Studio UI



Channel: KafkaChannel

Configuration

Description:

Driver: Kafka(API Example)

Method of Configuration*: Properties

Extended Configuration

Properties

Kafka Broker Urls: localhost:9092

Destinations

AllOps

Name*:	AllOps
Description:	
Default Event*:	/Events/CreateAccount
Serializer/Deserializer:	com.tibco.be.custom.channel.kafka.serializer.KafkaMapSerializer
GroupID:	kafka_group
ClientID:	BEClient
Topic:	ktopic
Consumer Threads:	1
Poll Interval:	100
Compression Type:	none

No consoles to display at this time.

FTL Channel

TIBCO FTL® messaging product is used for point-to-point (PTP) messaging between applications. TIBCO BusinessEvents can send and receive events by using TIBCO FTL as transport.

To know more about TIBCO FTL, you can refer TIBCO FTL documentation at https://www.tibco.com/products/tibco-ftl.

By using the FTL Channel, BusinessEvents can receive FTL message and transform them into BusinessEvents events.

Datatype Conversion

The following FTL datatypes are supported for the FTL message to BusinessEvents event conversions:

- String
- Long
- DateTime
- Double

To define property of any FTL subscriber, publisher, realm, or eventqueue, append the `be.channel.ftl.` prefix to the property name.

Syntax:

```
be.channel.ftl.<property_name>=<value>
```

To set the property for a particular destination, use the destination URI.

Syntax:

```
be.channel.ftl.<destinationURI>.<property_name>=<value>
```

To set the property for all destinations of a channel, use the channel URI.

Syntax:

```
be.channel.ftl.<channelURI>.<property_name>=<value>
```

Adding an FTL Channel in BusinessEvents Application

Using the FTL channel, TIBCO BusinessEvents transforms an incoming FTL message into a BusinessEvents Event and transforms a BusinessEvents Event into an FTL message.

Before you begin

TIBCO FTL should be installed on the system. For information on installation of the software, refer to the TIBCO FTL Documentation.

Procedure

1. In TIBCO BusinessEvents Studio, add a channel with **Driver Type** as `FTL`.

   See Adding a Channel for more details.

2. You can configure the FTL channel by using a shared resource or by configuring the channel properties. See FTL Connection Reference. for using a shared resource. Otherwise, in the channel editor, enter the FTL channel configuration properties and select **Method of Configuration** as Properties.

   See Channel Resource Reference for more details about the FTL channel configuration properties.

3. Now, in the channel editor, add a destination for the FTL channel for listening to the messages from the FTL channel by using either of the following methods:

   - Add a destination to the FTL channel following the procedure Adding a Destination to a Channel. Enter values for the destination configuration properties. See Destination Resource Reference for details about the FTL destination properties.
   - Add the destination and event for the FTL channel using a wizard, see Creating FTL Destination and Events Using BusinessEvents Studio Wizard

4. Save the new channel.

Creating FTL Destination and Events Using BusinessEvents Studio Wizard

By using the TIBCO BusinessEvents Studio, you can create destinations and events required for communication with TIBCO FTL. The FTL Destination and Event Wizard automatically analyze configuration data based on the configuration file generated by the FTL server to help you to generate FTL destinations and events.
For more information about the FTL entities involved in the creation of a FTL channel event and destination, refer *TIBCO FTL Documentation*.

Procedure

1. In BusinessEvents Studio Explorer, select **File > New > Other > TIBCO Channel Wizards > FTL Destination and Event** and click **Next**.
2. In the FTL Destination and Event window, select the FTL channel for the project from the FTL Channel list and click **Next**.
3. In the FTL Destination and Event window, click **Browse** and select the FTL server configuration JSON file. Click **Next**.

   This FTL server configuration JSON file is generated by FTL with all the required server configuration and FTL application details.
4. Select the FTL application for which you want to create the destination and click **Next**.
5. Select the details of the selected application and click **Next**.
   - **Endpoint List** - Select the endpoint for the application to be associated with the destination.
   - **Instance List** - Select the application instance for the instance.
   - **Format List** - Select the format name.
6. Select the FTL transport to be used for the application and click **Next**.
7. Enter the destination and events name. Also, specify the **Event Folder** and click **Next**.

   The FTL Destination and Event window opens the created destination name for confirmation.

8. (Optional) Click **Add more** to add more FTL destination and events. Repeat from Step 4 to Step 7.
9. (Optional) Click **Remove** to remove the selected destination.
10. Click **Finish** to confirm the creation of selected destination names.

Hawk Channels

The Hawk channel allows TIBCO BusinessEvents to receive events from the Hawk monitor and transform them to events in TIBCO BusinessEvents.

In order to use the channel to receive events from the TIBCO Hawk monitor, you must configure one or more destinations for the channel. Destinations represent different types of monitors listening to different Hawk events. The following monitor types are available:

AlertMonitor

AlertMonitor is used to listen on the events for any alerts being posted.

RuleBaseListMonitor

It is used to listen on the events for changes in the rulebase. For example, if some rulebases are added, updated, or removed.

Agent Monitor

It is used to listen on the events for any changes in the status of the agents. For example, the status may change from alive to expired.

MicroagentListMonitor

It is used to listen on the events for any microagents added or removed.

ErrorMonitor

It is used to listen on the event for errors.

WarningMonitor

It is used to listen on the event for warnings.

Microagent Method Subscription

It is used to subscribe to a microagent method at specified time intervals.

You need to create a channel with a driver type Hawk as described in Adding a Channel.

Working with Hawk Channels

Before proceeding to configure the Hawk channel, you must edit the configuration file `studio.tra` at *BE_HOME*`\studio\eclipse\configuration`.

Perform the following steps:

1. Add the following environment variables:

   ```
   tibco.env.HAWK_HOME=<absolute_path_where_TIBCO Hawk_is_installed>
   tibco.env.TRA_HOME=<absolute_path_where_TIBCO Rendezvous_is_installed>
   ```

   Depending on the transport used by the Hawk Connection shared resource, you must set one of the following:

   ```
   tibco.env.EMS_HOME=<absolute_path_where_TIBCO Enterprise Message Service_is_installed>
   ```

2. Edit the Studio extended classpath and append the following:

   ```
   ;%HAWK_HOME%/lib;%RV_HOME%/lib/tibrvj.jar;%RV_HOME%/lib/;%TRA_
   HOME%/lib;%EMS_HOME%/lib
   ```

3. Restart TIBCO BusinessEvents Studio to pick up the updated values if it is already started.

Setting Up the Runtime Configuration

Before using the Hawk channel at runtime, edit the configuration file `be-engine.tra` at *BE_HOME*`\bin` and set the following variables:

Procedure

1. Add or edit the environment variable *HAWK_HOME*:

   ```
   tibco.env.HAWK_HOME=<absolute_path_where_TIBCO Hawk_is_installed>
   ```

2. Depending on the transport used by the Hawk Connection shared resource, set one of the following:

```
tibco.env.RV_HOME=<absolute_path_where_TIBCO Rendezvous_is_installed>
tibco.env.EMS_HOME=<absolute_path_where_TIBCO Enterprise Message Service_is_installed>
```

Using the Hawk Destination and Event Wizard

The Hawk Destination and Event Wizard can be invoked from BusinessEvents Studio Explorer, select File > New > Other > TIBCO Channel Wizards > **Hawk Destination and Event**.

You can choose one of the wizard types:

- **Create a Destination and Event** Creates a destination and a simple event for the specified Hawk channel.
- **Create Event for Hawk Catalog Function** Creates a simple event for catalog functions that invoke a specified microagent method from a rule or rulefunction.

Creating a Destination and Event Using the Wizard

Procedure

1. Start the Hawk Destination and Event Wizard as described in Using the Hawk Destination and Event Wizard and select the wizard type Create Destination and Event. Click **Next**.
2. On the Select Channel dialog, select the Hawk channel from the list for which you need to create a destination and event. Click **Next**.
3. On the Select Monitor Type dialog, select the monitor type for the Hawk channel. The monitor type determines the type of Hawk events that are monitored. Click **Next**.

   If you selected MicroAgent Method Subscription, proceed to *Step 4*. Otherwise go to *Step 7*.
4. The wizard detects the available Hawk agents and displays them on the Select Hawk Agent dialog. Select a Hawk agent from the list and click **Next**.

5. The Select Hawk MicroAgent dialog is displayed. Select a microagent from the list of available Hawk microagents and click **Next**.
6. The Select Hawk MicroAgent Method dialog is displayed. Select a method from the list of available methods of the microagent. The tooltip displays the arguments for the selected method along with their datatypes.
7. The Set File and Folder name dialog is displayed. Enter the name of the destination to be created, the folder name where the event is to be stored, and the name of the event to be created.
8. Click **Finish**. A message ’Created the resources of Hawk Channel successfully.’ indicates that the Hawk destination and the event have been created.

   Navigate the project directory in BusinessEvents Studio Explorer to verify that the specified destination and event have been created. The return elements for the selected method are added to the event as properties.

Creating Event for Hawk Catalog Function Using Wizard

Procedure

1. Start the Hawk Destination and Event Wizard as described in Using the Hawk Destination and Event Wizard and select the wizard type Create Event for Hawk Catalog Function. Click **Next**.
2. On the Select Channel dialog, select the Hawk channel from the list for which you need to create an event. Click **Next**.
3. The wizard detects the available Hawk agents and displays them on the Select Hawk Agent dialog. Select a Hawk agent from the list and click **Next**.
4. The Select Hawk MicroAgent dialog is displayed. Select a microagent from the list of available Hawk microagents and click **Next**.
5. The Select Hawk MicroAgent Method dialog is displayed. Select a method from the list of available methods of the microagent.
6. The Set File and Folder name dialog is displayed. Enter the folder name where the event is to be stored and the name of the event to be created.

7. Click **Finish**. A message ’Created the resources of Hawk Channel successfully.’ indicates that the event has been created.

   Navigate the project directory in BusinessEvents Studio Explorer to verify that the specified event has been created in the specified folder. The return elements for the selected method are added to the event as properties.

   Hawk Channel Destination Reference

Field Name	**Description**
Name	Name of the destination to be created. The default name on the screen is of the format NewDestination_n.
Description	Description of the destination that is to be created.
Default Event	The default event for the destination. You can browse and select an existing event from the project.
Serializer/Deserializer	The default serializer used to deserialize a Hawk event to a simple event in TIBCO BusinessEvents and to serialize an event in TIBCO BusinessEvents to a Hawk event. You must configure the serializer (`com.tibco.cep.driver.hawk.serializer.HawkSerializer`) for every destination created on the Hawk channel.
Monitor Type	Type of monitor listening to different Hawk events. Choose one of: • AlertMonitor • RuleBaseListMonitor • AgentMonitor • MicroAgentListMonitor • ErrorMonitor • WarningMonitor • MicroAgentMethodSubscription

Field Name	Description
Subscription Method URI	URI for the Hawk microagent subscription method. When you use the TIBCO Channel Wizard to create a destination, you can select a subscription method from the list of available methods.
TimeInterval (seconds)	Time interval (specified in seconds) between successive calls to a subscription method.
Arguments	Optional. The subscription method selected determines if the arguments are required. Some methods do not require any arguments. Enter the arguments required by the selected subscription method, if any. If you create the destination using the TIBCO Channel wizard, the wizard automatically enters the arguments for the selected subscription method, if any.

HTTP Channels

Working with HTTP channels allows you to add an HTTP connection, add a destination, create events, configure rules, set up fault tolerance, and create resources to work with SOAP and a WSDL File.

Before you can configure TIBCO BusinessEvents to receive and send HTTP requests, ensure the following:

- Configure proxy for both SSL and non-SSL so that the end target servers can be routed through the proxy. Do not interpret that authentication at any level is supported.
- Use catalog functions ending with `ViaProxy`.
- Create a `connectioninfo` object and pass it through the HTTP function.

Note: Once `HTTP.ConnectionInfo` is created, pass the object through the `HTTP.sendRequest()` method.

SOAP Channels

An HTTP channel is an internal HTTP server. When the TIBCO BusinessEvents engine starts, it starts the internal HTTP server, which listens to the requests on the port specified in the HTTP Connection resource.

SOAP version 1.1 is supported. TIBCO BusinessEvents supports only document/literal type of encoding of SOAP requests.

A SOAP event is an extension of a SimpleEvent. To create a SOAP event, you create a SimpleEvent that inherits from a `SOAPEvent`. This creates a default schema in the event payload. Then you edit the schema and introduce header and body elements as necessary.

Using an HTTP channel and a destination configured to use the SOAP serializer and deserializer, TIBCO BusinessEvents can act as a web services platform that sends and receives SOAP requests, and performs whatever operations are provided by the web service.

TIBCO BusinessEvents can import a WSDL file and create the required project artifacts based on it, such as events, rules, rule functions, channels, and destinations. For more details, see Creating Resources Using the WSDL Import Utility in

TIBCO BusinessEvents can also export a SOAP-based rule function to a WSDL. The export utility scans the project for rule functions that take a SOAP event as the input, and generates a WSDL operation for each one. For more details, see *TIBCO BusinessEvents Developer Guide*, Exporting Resources as a WSDL File.

For more details, see Configuring TIBCO BusinessEvents as a SOAP Server and Client.

Adding an HTTP Connection

Add an HTTP Connection resource to your project. In the **Host** and **Port** fields, specify the host and port to which HTTP clients send requests.

In the **Host** field, enter the name or IP address of the machine running TIBCO BusinessEvents. This is the HTTP server.

In the **Port** field, enter any available port on the host machine. This is the port on which the server listens for HTTP requests.

To configure an HTTPS (Secure) Connection

Select the **SSL** checkbox, click the **Configure SSL** button, and complete the pop-up dialog settings. The server must authenticate to the client. In the **Identity** field, provide the location of the Server Identity File. (See Adding an Identity Resource — For SSL Only).

Note: In one-way SSL, the server authenticates itself to the client using a Server Identity File, but the client does not have to authenticate to the server.

In two-way SSL, the server authenticates itself to the client using a Server Identity File and the client also authenticates to the server using the Client Identity file.

Select the **Requires Client Authentication** checkbox. This enables the **Trusted Certificates Folder** field, where you provide the Client certificates.

Adding an Identity Resource — For SSL Only

TIBCO BusinessEvents supports use of an identity file for SSL. Before you configure the secure HTTP connection, add an Identity resource to your project and configure it.

In the **URL** field, specify the project path of the KeyStore file, which must be within the project folders.

In the **File Type** field, specify the KeyStore file type, and in the **Password** field, provide the password for the KeyStore file.

Adding an HTTP Channel

Add a channel to your project and configure it as follows:

1. At the New Channel Wizard, provide a name and description, and in the **Driver** field select HTTP. Click **Finish**.
2. In the Channel editor **Channel** tab, update the description as desired. The **Driver** field is set to HTTP (as set in the wizard). The Method of Configuration is preset to Resource and cannot be changed.
3. In the **Resource** field, browse to the HTTP connection resource you configured in Adding an HTTP Connection.
4. Click **Advanced**, and configure run-time configuration properties. These properties provide information such as the location of the document root folder. Some properties are for SSL configuration of HTTP Component servers.

Adding a Destination

Add a destination to the channel in the usual way.

The fields for destination differs based on different approaches.

To follow the action rule function-based approach

Select the **Type** value as PageFlow and specify the appropriate **Context Path** and **Action Rule function** for the destination. Specify the context path in the same format as the server would receive the `requestURI` in the HTTP request.

To follow the event-based approach

In the **Serializer** field select the appropriate serializer:

```
com.tibco.cep.driver.http.serializer.RESTMessageSerializer
```

Specify the default event in the usual way as needed by your project requirements.

> **Note:** HTTP clients of the TIBCO BusinessEvents server would use the complete destination URI, after the host and port for example, `http://www.acme.com:5560/Transport/Channel/MyDestination`.

WebSocket Protocol

If required, you can also use WebSocket protocol for the HTTP channel. Select the WebSocket serializer (com.tibco.cep.driver.http.serializer.WebSocketMessageSerializer) for the destination and value of the **Type** field as WebSocket.

Creating Events as Needed and Setting Default Destinations

Skip this step for action rule function based approach.

In the event-based approach

For receiving HTTP requests and sending responses, configure events in the usual way, and select an HTTP-based destination as the default destination.

Configuring Rules and Rule Functions

Configure rules and rule functions according to your needs and HTTP request processing approach.

In the action rule function-based approach

For example, in response to a POST request you might do the following actions:

- Create a servlet request object for the HTTP request using the catalog functions to extract data from the HTTP request.
- Identify the POST request method from the HTTP request and process the parameters and data extracted
- Create a response message to be used in preprocessor or rule or to be sent directly to the HTTP client.

In the event-based approach

For example, in response to a POST request you might do the following actions:

- Create a concept instance, using XPath functions to extract data from the POST data in the request event payload
- Create a response event and use `Event.replyEvent()` to send back an empty response using the request event's default destination.

As another example, in response to a `GET` request you might do the following actions:

- Identify a concept instance using a property in the request event (created on arrival of the `GET` request message).
- As needed, identify or generate data to return and create a response event to hold that data.
- Return the data using `Event.replyEvent`.

In the CDD Configure the Processing Unit

In the Cluster Deployment Descriptor (CDD) configure the processing unit for deployment as needed.

Working with HTTP Requests

An HTTP request is mapped to an event.

HTTP requests are parsed and run using either of the following approaches:

Event-based approach

HTTP request is mapped to an event using a deserializer.

The supported methods are as follows:

- GET
- POST

If the `Type` parameter is set to WebSocket, the destination uses the WebSocket protocol.

Action Rule Function based approach

HTTP request parameters and data are retrieved using HTTP catalog functions and processed using a rule function.

The `Type` parameter of the destination identifies the approach followed by the destination for processing HTTP requests. If the `Type` parameter is set to PageFlow, the HTTP request is processed using the Action Rule Function based approach, otherwise the Event-based approach is used.

The supported methods are as follows:

- GET
- PUT
- POST
- HEAD
- OPTIONS
- TRACE
- PATCH

HTTP Requests and Events Mapping with the REST Serializer

`RESTMessageSerializer`, which is set while configuring the channel, maps HTTP requests to TIBCO BusinessEvents.

HTTP headers and HTTP parameters in the GET method are mapped to similarly named event properties. When both parameters and headers are specified, parameters take precedence.

Each HTTP Header consists of a name and field value, which are mapped into an event property name and value. The POST data in the request must match with the payload (XML or JSON) as defined in the corresponding event. If no payload is defined for the event, the POST data is translated into a `ByteArray` payload, and is not accessible through the mapper. XML is set as the default payload for all events; however, you can also configure the HTTP channel for JSON payload. See JSON Payload for more details.

The transfer encoding (`charset`) in the `Content-Type` header indicates what type of transformation is applied to the POST data (message body) to safely transfer content between sender and recipient. If the `Content-Type` header is missing, UTF-8 is used as the default transfer encoding.

When you want the REST serializer to deserialize a GET request into an event with a payload, include the `_payload_` request parameter. The string value of the `_payload_` parameter is always used as payload in the event.

Note: Avoid sending long requests using GET. For large payload data requests, it is recommended to use the POST method.

For HTTP responses, all event properties are translated to similarly named HTTP headers and the payload is sent as HTTP content.

Sending Non-ASCII Content to Event Properties

The HTTP 1.1 specification states that only ASCII characters can be sent in HTTP headers.

To send non-ASCII event properties in GET methods, use HTTP parameters. HTTP parameters are passed as the `QueryString` of the request URI, that is, the part of the URI that contains data to be passed to web applications.

To decode the `QueryString`, use either the URI Encoding or the body encoding. The body encoding is specified in the `contentType` HTTP header. Select either the URI Encoding or Use Body Encoding for URI setting in the HTTP channel Advanced tab.

Mapping of HTTP Request URI to Destination

An HTTP request URI must map to a valid TIBCO BusinessEvents destination. If not, an error is returned, and the message is discarded.

In the event-based approach, once the destination is established, the HTTP message is converted either into an event-based upon `_ns_/_nm_`, or into the default event associated with this destination.

In this case TIBCO BusinessEvents server looks for the destination having the same URI as the `requestURI`.

For example, the `requestURI` for the request `https://localhost:7000/Transport/Channel/StudentDestination` is `/Transport/Channel/StudentDestination`. TIBCO BusinessEvents engine maps the request with a destination having `URI /Transport/Channel/StudentDestination` if it exists.

JSON Payload

TIBCO BusinessEvents can deserialize a JSON payload to an event and can serialize an event to JSON.

For processing the JSON payload, you must perform these tasks. Otherwise, the payload is treated as XML:

- Select the **Is JSON Payload** while creating a destination. See Adding a Destination to a Channel and Destination Resource Reference.
- The event mapped to the destination should have the destination URL in its default destination field.

Note:

- Regular payload validation rules apply as they would for default XML-based events, that is, if the namespace check fails, the payload is not set. Namespace for an XML element is part of its root element, while for JSON it is part of the "attributes" node under "type".
- After a JSON payload is deserialized to an event, it is used the way an XML-based event is used.

The "type" information is no longer needed when parsing the JSON payload. Previously this was required. In the output, the "type" information for the top-level elements is only printed. The attributes of the nested elements are printed if it contains an Id or extId.

When processing the JSON payload, by default all properties are converted to their respective data type. For example, if an int property is passed as a string in the payload, it is converted to an int. To disable this default behavior and enable the behaviour where all

JSON properties are treated as strings, set the system property `com.tibco.datamodel.untyped` to `true`. If you set this property to true, creating an event payload with the mapper does not convert single values to a JSON array if the payload element corresponds to an array property.

XML and JSON Payload Examples

XML Request

```
<root xmlns="www.tibco.com/be/ontology/Events/Event2">
    <param1>testProp</param1>
    <param2>123</param2>
    <param3>true</param3>
</root>
```

JSON Request

```
{"root" :
  { "attributes" :
    { "type" : "www.tibco.com/be/ontology/Events/Event2" }
    ,
    "param1" : "testProp",
    "param2" : "123",
    "param3" : "true"
  }
}
```

XML Response

```
<Address>
    <street>243 Buena Vista Avenue</street>
    <city>Sunnyvale</city>
    <zip>94086</zip>
</Address>
```

JSON Response

```
{"Address" :
{ "street" : "243 Buena Vista Avenue",
  "city" : "Sunnyvale",
  "zip" : "94086" }
}
```

Payload Without Wrapping Elements

You can set the payload schema to send the JSON document with no wrapping elements. To exclude the wrapper and all the internal attribute elements, use the following properties:

- `be.http.json.rootElement.ignore`
- `be.http.json.childAttributes.ignore`

To ignore the parent element and its attributes, set the `be.http.json.rootElement.ignore` property to `true` and to ignore the child concept attributes, set the `be.http.json.childAttributes.ignore` to `true`. The default value of these properties is `false`.

Note: When the property `be.http.json.rootElement.ignore` is set to true, you cannot transform event payload data by using XPath and XSLT mapper. Use other JSON catalog functions to parse and access JSON nodes manually, or write a custom function to do the same.

During JSON serialization of both concepts and events, you can use the following properties:

- `json.serialization.attributes.exclude`
- `json.serialization.childAttributes.exclude`

To exclude all the attributes, including the top-root level attributes, set the `json.serialization.attributes.exclude` property to `true`. Whereas to ignore the child attributes of any nested elements, set the `json.serialization.childAttributes.exclude` property to `true`. However, here the attributes at the root JSON node are printed. The default value of these properties is `false`.

Action Rule Function Based Approach

HTTP request can be processed in TIBCO BusinessEvents Studio without mapping to an event.

An HTTP request can be processed in two ways:

HTTP Request Processing

The TIBCO BusinessEvents server parses the HTTP request received from client. The server extracts the request header and identifies the request URI of the request. The request URI should map to the context specified in the destination defined for the channel. In this case TIBCO BusinessEvents server looks for the destination having the same context path as the `requestURI`.

For example, the `requestURI` for the request `https://localhost:7000/Transport/Channel/StudentDestination` is `/Transport/Channel/StudentDestination`. TIBCO BusinessEvents engine maps the request with a destination having context path `/Transport/Channel/StudentDestination` if it exists.

See Sample Code for Action Rule Function.

Deploying Multiple Web Applications

You can deploy multiple web applications on single channel. Specify the *.WAR file or a valid J2EE web application folder under the Web Applications section in the advanced tab. Specify the context URI and resource path for the web application. Resource path identifies the actual path of web application resources. In case of a *.WAR file resource path is the location of the war file, and in case of the J2EE web application folder the resource path is the location of the base folder of the web application.

Sample Code for Action Rule Function

Action rule function defined in the destination uses catalog functions to get required data and parameters from the HTTP request. The rule function processes these parameters and creates a response, similar to a preprocessor. TIBCO BusinessEvents supports all HTTP methods stated in the HTTP 1.1 specification for Action Rule Function based approach.

```
void rulefunction RuleFunctions.Callback {
   attribute {
      validity = ACTION;
   }
   scope {
      Object asyncContextObject;
   }
   body {
      //getting servlet request and response objects
      Object servletRequest = HTTP.Servlet.getServletRequest
(asyncContextObject);
      Object servletResponse = HTTP.Servlet.getServletResponse
(asyncContextObject);
      System.debugOut("##Servlet request method : " +
HTTP.Servlet.Request.getMethod(servletRequest));
      System.debugOut("##Servlet request content : " +
HTTP.Servlet.Request.getRequestContent(servletRequest));
      System.debugOut("##Servlet request Requester Address : " +
HTTP.Servlet.Request.getRequestorAddress(servletRequest));
      System.debugOut("##Servlet request Request URI : " +
HTTP.Servlet.Request.getRequestURI(servletRequest));
      //getting parameters
      String[] params = HTTP.Servlet.Request.getRequestParameters
(servletRequest);
      for(int i=0;i<params@length;i++)
      {
         System.debugOut("## Servlet request Parameters  :" +
HTTP.Servlet.Request.getRequestParameter(servletRequest,params[i]));
      }
      //getting headers
      System.debugOut("## Servlet request Header Accept  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept"));
      System.debugOut("## Servlet request Header Accept-Encoding  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept-
Encoding"));
      System.debugOut("## Servlet request Header Accept-Language  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept-
Language"));
      System.debugOut("## Servlet request Header Accept-Charset  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept-Charset"));
      System.debugOut("## Servlet request Header Connection  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Connection"));
      System.debugOut("## Servlet request Header User-Agent: " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"User-Agent"));
      System.debugOut("## Servlet request Header Content-Length  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Content-Length"));
      System.debugOut("## Servlet request Header Content-Type  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Content-Type"));
      System.debugOut("## Servlet request Header Host  : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Host"));
      HTTP.Servlet.Response.setResponseHeader(servletResponse, "Content-
type", "text/plain");
      HTTP.Servlet.Response.setResponseContent(asyncContextObject,
"response", true);
   }
}
```

Setting Up Fault Tolerance for the HTTP Channel

Use the Apache HTTP server ("httpd") and mod_jk to setup fault tolerance to the TIBCO BusinessEvents HTTP channel and a third-party fronting server. These properties are used in addition to sharing the cache server between the nodes in the cluster.

Procedure

1. Create two new property files with suitable names. For example, `Ajp_1.properties` and `Ajp_2.properties`.
2. Add the AJP connector port property in the two newly created property files.

   In `Ajp_1.properties`:

   ```
   <channel path>.ajp.connector.port=8011
   ```

   In `Ajp_2.properties`:

   ```
   <channel path>.ajp.connector.port=8012
   ```

3. Specify the new property files as part of the TRA file of individual engines. Ensure that there is one properties file per engine.

   ```
   java.property.be.channel.external.config.file=<location of
   properties file>
   ```

   For example,

   ```
   java.property.be.channel.external.file=/tibco/app/conf/Ajp_
   1.properties
   ```

   ```
   java.property.be.channel.external.file=/tibco/app/conf/Ajp_
   2.properties
   ```

4. Configure the Apache HTTP server (“httpd”) as a fronting web server.

    a. Add the `worker.properties` file in Apache httpd under the `config` folder.

    b. Set the Apache http for load balancing in the `worker.properties` file.

    c. Configure two workers to listen to the above specified AJP ports.

    d. Disable one worker (for example, worker 2) by default, so that no requests are sent to it.

    e. Configure another worker (worker1), so that in case of a failure, it redirects all incoming requests to the disabled worker (worker2).

    The `worker.propeties` file should contain the following properties:

    ```
    worker.list=balancer
    worker.balancer.type=lb
    worker.balancer.sticky_session=0
    worker.balancer.balance_workers=worker1,worker2
    worker.worker1.type=ajp13
    worker.worker1.host=localhost
    worker.worker1.port=8011
    worker.worker1.redirect=worker2
    worker.worker2.type=ajp13
    worker.worker2.host=localhost
    worker.worker2.port=8012
    worker.worker2.activation=disabled
    ```

    Worker1, in case of a failure, now becomes the active worker.

5. Add Tomcat connectors `mod_jk.so` under the `modules` folder.

    You can download `mod_jk` from the following location:

    http://tomcat.apache.org/download-connectors.cgi

6. Configure Apache httpd to load this module.
    a. Redirect all requests to the load balancer defined earlier.
    b. Add the following snippet to `httpd.conf`, after completion of all the basic module load statements:

```
LoadModulejk_modulemodules/mod_jk.so
<IfModule jk_module>
JkWorkersFile conf/worker.properties
JkLogFile logs/mod_jk.log
JkLogStampFormat "[%b %d %Y - %H:%M:%S] "
JkRequestLogFormat "%w %V %T"
JkLogLevel info
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
<Location//WEB-INF/>
deny from all
</Location>
JkMount /* balancer
</IfModule>
```

7. Start Apache server; Start both the instances of be-engines running the HTTP-based project.
8. Open the browser and client server.
    a. Enter the URL `http://localhost/`*`<context-path>`*, without any port.

    The request is now routed to the active worker (worker1) only, since the other worker (worker2) is disabled. You can verify through logs that logs are being created for one be-engine only.
9. Shut down the active be-engine associated with the active worker (worker1).

    All subsequent requests from the client now route to other worker (worker2). To verify, check the logs in the corresponding be-engine instance.

HTTP Channel Advanced Configuration Settings

The following settings configure the internal Tomcat HTTP server used by the channel. They are set in the **Advanced** tab of the HTTP channel resource editor.

Important: You can use global variables in all the advanced properties. For more details, see Working with Global Variables.

HTTP Channel Advanced Configuration Settings

Property	Note
Server Type	The only server type in this release is TOMCAT.
Connector Type	Type of the TOMCAT connector. The default value is NIO.
No. of Connector Instances	Number of the connector instances. The default value is 1.
Debug Request Info	Select the checkbox to specify whether the information is displayed for request debugging. If **Debug Request Info** is selected, the following two debug fields are also activated for update: • **Debug Log Folder** • **Debug Log Pattern**
Debug Log Folder	Specifies the location of generating the log file. This field is activated only if the **Debug Request Info** checkbox is selected. The default location is a new log folder in the currently running folder.
Debug Log Pattern	Specifies the configurable pattern for generating the log content. The syntax of the pattern is the same as used in the Apache HTTP server. Please refer to the Tomcat documentation for more details about pattern syntax.
Debug Log Rotatable	Select this option to activate rotation (regular cycling) of log files by removing the old log files and creating ones. Use the following fields to set up log file rotation cycle and retention: • **Debug Log File Pattern** • **Debug Log Rename on Rotatable**

Property	Note
	• **Debug Log Max Days** The default status is *selected*.
Debug Log File Pattern	The pattern to specify the rotation cycle. The default pattern is `.yyyy-MM-dd`, which rotates the file every day. You can also specify the `.yyyy-MM-dd.HH` pattern to rotate every hour and `.yyyy-MM-dd.mm` to rotate log files every minute.
Debug Log Rename on Rotatable	Select this option to rename the log files automatically on rotation. The default status is *selected*.
Debug Log Max Days	Set a value to define the number of days after which older files are deleted. The default value is `-1` (forever).
Connection Timeout (msec)	The number of milliseconds the HTTP server waits after accepting a connection, for the request URI line (that is, the first part of the request message) to be presented. The value `0` means no timeout. The default is 60000.
Accept Count	The maximum queue length for incoming connection requests when all request processing threads are in use. Any requests received when the queue is full are refused. The value `-1` means that a default of 10 is set. The default is -1.
Socket Output Buffer Size	The size in bytes of the buffer to be provided for socket output buffering. The value `-1` means that the use of a buffer is disabled. The default is 9000.
Max Processors	The maximum number of simultaneous requests that can be handled by

Property	Note
	the HTTP server. Set to one of these: • A value of 10 or greater. Values of less than 10 are treated as 10. • The value `-1` which means that the value of 200 is used. The default is `-1`.
Connection Linger	The number of milliseconds during which the sockets used by the HTTP server linger (that is, not complete immediately) when they are closed. Use of socket linger allows time for a graceful shutdown sequence to complete. To disable use of socket linger, set to -1. The default is `-1`.
Enable DNS Lookups	Set to `true` if you want the calls to `request.getRemoteHost()` to perform DNS lookups and return the actual host name of the remote client. Set to `false` to skip the DNS lookup and return the IP address as a string instead (thereby improving performance). By default, DNS lookups are disabled.
TCP No Delay	If the checkbox is selected, the `TCP_NO_DELAY` option is set on the server socket. If the checkbox is cleared, the `TCP_NO_DELAY` option is not set on the server socket. Using `TCP_NO_DELAY` improves performance under most circumstances. The default is selected.
Compression	Compression can be used to save server bandwidth. Uses `HTTP/1.1 GZIP` compression. The compression option set here is used together with the MIME types shown in the Compressible Mime Types setting. The valid values are as follows: **off**

Property	Note
	Disables compression. Values in the Compressible Mime Types setting are ignored. **on** Enables compression **force** Forces compression. **An integer** Specifies a threshold amount of data above which output is compressed. The unit is bytes. For example, if set to `2048` then any file above 2MB is compressed. If content length is unknown, the output is always compressed. Compression applies to MIME types shown in the Compressible Mime Types setting that are over the threshold (or unknown). The default is off.
Use Body Encoding for URI	If selected, the encoding specified in the `contentType` HTTP header is used to decode the request URI. If not selected, the value in the URI Encoding field (or its default value) is used. If the URI Encoding setting is specified, then the Use Body Encoding for URI checkbox is ignored. For an example showing `contentType`, see Working with Outgoing SOAP Messages (Event Payloads) . By default, this option is not selected.
URI Encoding	Specifies the character encoding used to decode the request URI. If not specified, UTF-8 is used. If specified, the **Use Body Encoding for URI** setting is ignored. The default is UTF-8.

Property	Note
Max KeepAlive Requests	The maximum number of HTTP requests that can be pipelined until the connection is closed by the server. A value of 1 disables `HTTP/1.0 keep-alive`, as well as `HTTP/1.1 keep-alive` and pipelining. A value of -1 allows an unlimited amount of pipelined or keep-alive HTTP requests. The default value is -1.
Max HTTP Header Size	The maximum size of the request and response HTTP header, specified in bytes. The default is 4096, that is 4 KB.
Max HTTP Post Size	The maximum size of the POST data, specified in bytes, which is handled by the container FORM URL parameter parsing. You can disable the limit by setting this to less than or equal to 0. The default is 2097152, that is, 2 MB.
Max HTTP Save Post Size	The maximum size of the POST data, specified in bytes, which is saved or buffered by the container during FORM or CLIENT-CERT authentication. For both types of authentication, the POST data is saved or buffered before the user is authenticated. For CLIENT-CERT authentication, the POST data is buffered for the duration of the SSL handshake and the buffer emptied when the request is processed. For FORM authentication the POST is saved when the user is redirected to the login form and is retained until the user successfully authenticates or the session associated with the authentication request expires. A value of -1 means no limit. A value of 0 disables the saving of POST data during authentication. The default is 4096, that is 4 KB.
Scheme	Specifies the HTTP scheme to use. The valid values are as follows:

Property	Note
	• `http` - HTTP protocol • `https` - For secure communications (SSL) • `memory` - Memory protocol • `ajp` - Apache JServ Protocol, a binary protocol The default value is `http`.
Session Timeout Interval (secs)	Specifies the timeout interval for inactive HTTP sessions. The value is specified in seconds. The default value is 1800.
Max Spare Threads	The maximum number of unused request processing threads that are allowed to exist until the thread pool starts stopping the unnecessary threads. The default is 50.
Min Spare Threads	The number of request processing threads that are created when this `Connector` is first started. The connector also makes sure that it has the specified number of idle processing threads available. Set this to a value smaller than that set for Max Spare Threads. The default is 4.
Compressible Mime Types	This is a comma-separated list of MIME types for which HTTP compression may be used. This setting works with the Compression setting See notes for that setting for more details. The default value is `text/html,text/xml,text/plain`.
Restricted User Agents	Used to limit support to specific browsers and versions of specific browsers. Comma-separated list containing one or more browser names, such as Mozilla, Internet Explorer. Prepend version with a slash, for example, `Mozilla/4.0`.

Property	Note
Document Root	The absolute path where static HTML files are stored. The HTTP server retrieves pages from this location. This setting is used by internal TIBCO BusinessEvents applications and may have limited application for other purposes. **Tip:** To enable clients of a TIBCO BusinessEvents HTTP server to view the imported concrete WSDL, provide the URL to the document root folder. No default value. You can use this setting for running the `readme.html` file for your application on your machine.
Document Page	The name of the default static HTML file stored in the document root. This setting is used by internal TIBCO BusinessEvents applications and may have limited application for other purposes.
SSL Server: Key Manager Algorithm	The key manager algorithm for the SSL Service provider. The default is SunX509.
SSL Server: Trust Manager Algorithm	The trust manager algorithm for the SSL Service provider. Ensure that a provider is available for any algorithm other than PKIX. The default is PKIX.
SSL Server: Protocols	The SSL protocols that can be enabled on the server. Add each protocol using a comma-separated list. The default protocols are the ones supported by the SSL Provider. (SSLv3 and TLSv1 are the most widely supported.) Leave blank to use the default protocols for your SSL provider.
SSL Server: Ciphers	The cipher suite (SSL protocols) used by the server. The following cipher suites are supported: • `TLS_RSA_WITH_AES_128_GCM_SHA256`

Property	Note
	• `TLS_RSA_WITH_AES_256_GCM_SHA384` • `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256` • `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384` • `TLS_DHE_DSS_WITH_AES_128_GCM_SHA256` • `TLS_DHE_DSS_WITH_AES_256_GCM_SHA384` • `TLS_AES_128_GCM_SHA256` • `TLS_AES_256_GCM_SHA384` • `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256` • `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384` • `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` • `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` • `TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256` • `TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384` • `TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256` • `TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384` • `TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256` • `TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384` • `TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256` • `TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384` • `TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256` • `TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384` • `TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256` • `TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384` • `TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256` • `TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384` • `TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256`

Property	Note
	• `TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384` • `TLS_RSA_WITH_AES_128_CCM` • `TLS_RSA_WITH_AES_256_CCM` • `TLS_DHE_RSA_WITH_AES_128_CCM` • `TLS_DHE_RSA_WITH_AES_256_CCM` • `TLS_RSA_WITH_AES_128_CCM_8` • `TLS_RSA_WITH_AES_256_CCM_8` • `TLS_DHE_RSA_WITH_AES_128_CCM_8` • `TLS_DHE_RSA_WITH_AES_256_CCM_8` • `TLS_ECDHE_ECDSA_WITH_AES_128_CCM` • `TLS_ECDHE_ECDSA_WITH_AES_256_CCM` • `TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8` • `TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8` • `TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256` • `TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256` • `TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256` Leave blank to use the default cipher suites for your SSL provider.
SSL Server: Use Cipher Order	Use the server cipher order to choose from the common list of ciphers between the client and the server. Select the **SSL Server: Use Cipher Order** checkbox to enable this property.
Server Header	Specifies the text that overrides the server details, displayed in the response header. The default value is "Apache-Coyote/1.1". The **Server Header** field is useful when you do not want to display server details in the response HTTP header for enhanced security.

You can configure additional HTTP channel connection properties provided by TIBCO BusinessEvents on the **Cluster** tab in the project CDD file or in the system TRA file to configure your HTTP channel application. For the list of TIBCO BusinessEvents HTTP channel connection properties, see *TIBCO BusinessEvents Developer Guide*

Defining Event Properties for Standard HTTP Header Properties

You need to define corresponding event properties for HTTP header names so that the header names are mapped to the event properties at run time.

Standard HTTP header properties can have a dash (-) in their names. While defining the corresponding event properties for such header properties, use an underscore character (_) instead of a dash (-). A dash is *not* allowed in the event property names.

Following is the list of the HTTP header properties. In this a dash is replaced by an underscore while converting from HTTP headers to Event Properties, and vice versa.

```
accept-charset
accept-encoding
accept-language
accept-ranges
cache-control
content-type
if-match
if-modified-since
if-none-match
if-unmodified-since
max-forwards
proxy-authorization
user-agent
content-encoding
content-disposition
content-language
content-location
content-md5
content-range
last-modified
proxy-authenticate
retry-after
set-cookie
transfer-encoding
proxy-authenticate
retry-after
set-cookie
```

```
transfer-encoding
www-authenticate
```

Defining the `HttpStatusCode` property to the event properties explicitly sets the HTTP status code. `HttpStatusCode` can be an integer or a string, and is not case-sensitive.

HTTP Functions for HTTP Request Messages Configuration

This section explains how HTTP functions are used to send secure and non-secure HTTP requests to other servers, and work with responses received.

HTTP functions are in the HTTP section of the Standard function catalog.

Note: For configuring SSL over HTTP, when the Identity Resource is of the type Certificate/KeyURL, you need to set up Native Tomcat libraries. For more information, see the *TIBCO BusinessEvents Installation*.

Generating a Self-Signed SSL Certificate (KeyStore)

HTTPS requires use of an SSL certificate. It creates a KeyStore file and password for the specified type of SSL certificate (keystore).

Note: Avoid using self-signed certificates for production.

If you use trusted certificates, see Loading Trusted Certificates.

Signature

```
Object createKeystore(String ksFilePath, String ksType, String ksPassword)
```

Description

Creates and returns a KeyStore object, using the given parameters.

Parameters

Name	Type	Description
`ksFilePath`	`String`	The absolute path of the KeyStore file.
`ksType`	`String`	The type of KeyStore. JKS and PKCS12 are supported.
`ksPassword`	`String`	Obfuscated password for the KeyStore.

Returns

Type	Description
`Object`	The KeyStore object

Getting POST Data

Signature

```
Object getPostData(SimpleEvent event)
```

Description

Returns the POST data sent in an HTTP POST request (event).

Parameters

Name	Type	Description
`event`	`SimpleEvent`	Request event sent on the HTTP destination. Must not be null.

Returns

Type	Description
`Object`	The POST data

Loading Trusted Certificates

As explained in Adding an HTTP Connection, ensure that certificates from trusted certificate authorities are stored in the project, using an Identity resource in the TIBCO Shared Resources folder. Alternatively, use the `BE_GLOBAL_TRUSTED_CA_STORE` global variable to store a location of the certificates outside of the project.

Signature

```
Object loadTrustedCertificates(String trustedCertsFolder, String
passwordToSet)
```

Description

HTTPS requires the use of an SSL certificate. This function loads a trusted certificate (that is, creates and returns a keystore object) from the trusted certificates folder.

Parameters

Name	Type	Description
`trustedCertsFolder`	`String`	The project path to the folder containing the certificates. If trusted certificates are stored outside the project, use the following constructs in the function: `System.getGlobalVariableAsString ("BE_GLOBAL_TRUSTED_CA_STORE")`
`passwordToSet`	`String`	Obfuscated password for the KeyStore.

Returns

Type	Description
`Object`	The certificate's KeyStore object

Sending an Event

To send an event as a request encapsulating request headers as properties.

Signature

```
Event sendRequest(String url, SimpleEvent requestEvent, String
responseEventURI, long timeoutMillis, Object httpConnectionInfo)
```

Description

This function sends an event as a response to the request.

You can disable the cookies using the `disableCookies()` function. To ensure that the entire message along with the response header and the request body is sent at once, use `disableExpectContinueHeader()`.

Parameters

Name	Type	Description
`url`	`String`	The URL for the endpoint that will receive this request.
`requestEvent`	`SimpleEvent`	The event to serialize and send to the server.
`responseEventURI`	`String`	The fully qualified path of an event. This event is created when the response is received.
`timeoutMillis`	`Long`	The timeout interval for the operation. If the value is `-1`, the server waits forever.
`httpConnectionInfo`	`Object`	HTTP Connection Info object.

Returns

Type	Description
`Event`	An event as a response to the request

Sending an Asynchronous Request (Not Secure)

To send requests to an HTTP server asynchronously, use `sendAsynchronousRequest` function.

Signature

```
String sendAsynchronousRequest(String url, SimpleEvent requestEvent,
String correlationId, String successCallbackRuleFunctionURL, String
errorCallbackRuleFunctionURL, Object httpConnectionInfo)
```

Description

Sends a request to the server specified by the `url` parameter. When TIBCO BusinessEvents receives a response, the callback function is called.

Returns a correlation ID, which is either passed as input, or is generated from the server if the parameter is null. This ID enables you to correlate a request with its response.

You can disable the cookies using the `disableCookies()` function. To ensure that the entire message along with the response header and the request body is sent at once, use `disableExpectContinueHeader()`.

For handling timeout case, you can set the java property `com.tibco.be.http.client.socketTimeout` with the required timeout period.

Set the property `be.engine.http.client.async.responseEventOnError.enabled` to true to pass an additional parameter (output event) to the errorCallback rulefunction specified in the `HTTP.sendAsynchronousRequest()` function. The default value of the property is false.

Parameters

Name	Type	Description
`url`	`String`	The URL for the server that will receive this request.

Name	Type	Description
`requestEvent`	`SimpleEvent`	The event to serialize and send to the server.
`correlationID`	`String`	An optional ID to correlate the request and the response. If not specified, the ID is generated by the server.
`successCallbackRuleFunctionURL`	`String`	The fully qualified path of a rule function to be invoked for success case. This rule function is called when a successful response is received. The response event would contain the correlation ID. The rule function must have correlation ID, RequestEvent, and ResponseEvent as parameters.
`errorCallbackRuleFunctionURL`	`String`	The fully qualified path of a rule function to be called in case of an error. This rule function is called when the response received

Name	Type	Description
		indicates an error. The response event would contain the correlation ID. The parameter can be null. The rule function must have correlation ID, RequestEvent, and ResponseEvent as parameters.
`httpConnectionInfo`	`Object`	HTTP Connection Info object.

Returns

Type	Description
`String`	A correlation ID

Sending a Secure Asynchronous Request

To send asynchronous requests using SSL, use this function.

Description

This function is the same as the `sendAsynchronousRequest()` function.

For secure communication, use `sendAsynchronousRequest()` function along with the `setSecureInfo()` catalog function with the addition of the SSL-related parameters shown below:

```
Object connectionInfo= HTTP.ConnectionInfo.createHTTPConnectionInfo
(true);
HTTP.ConnectionInfo.setSecureInfo(Object connectionInfo, String
sslProtocol, Object clientIdKeyStore, String clientIdPassword, Object
trustedCertsKeystore, String trustedCertsPassword, boolean
verifyHostName)
```

For handling timeout case, you can set the java property `com.tibco.be.http.client.socketTimeout` with the required timeout period.

Parameters

All of the parameters for `sendAsynchronousRequest()` plus the following:

Name	Type	Description
`clientIdKeystore`	`Object`	The keystore object for client identity.
`clientIdPassword`	`String`	Password for the client ID keystore.
`trustedCertsKeystore`	`Object`	Keystore Object for trusted certificates.
`trustedCertsPassword`	`String`	Password for the trusted certificates keystore.
`verifyHostName`	`Boolean`	Flag for checking if a host name matches the names stored in the server's certificates.
`httpConnectionInfo`	`Object`	HTTP Connection Info object.

Returns

Type	Description
`String`	A correlation ID

Sending a Secure Synchronous Request

To send synchronous requests using SSL, use this function.

Description

This function is the same as the `sendRequest()` function.

For secure communication, use `sendRequest()` function along with the `setSecureInfo()` catalog function for SSL-related parameters:

```
Object connectionInfo= HTTP.ConnectionInfo.createHTTPConnectionInfo
(true);
HTTP.ConnectionInfo.setSecureInfo(Object connectionInfo, String
sslProtocol, Object clientIdKeyStore, String clientIdPassword, Object
trustedCertsKeystore, String trustedCertsPassword, boolean
verifyHostName)
```

Parameters

Name	Type	Description
`url`	`String`	The URL for the server that will receive this request.
`requestEvent`	`SimpleEvent`	The event to serialize and send to the server.

Name	Type	Description
`responseEventURI`	`String`	The fully qualified path of an event. This event is created when the response is received.
`clientIdKeystore`	`Object`	The keystore object for client identity.
`clientIdPassword`	`String`	Password for the client ID keystore.
`trustedCertsKeystore`	`Object`	Keystore Object for trusted certificates.
`trustedCertsPassword`	`String`	Password for the trusted certificates keystore.
`verifyHostName`	`Boolean`	Flag for checking if a host name matches the names stored in the server's certificates.
`timeoutMillis`	`Long`	The timeout interval for the operation. If the value is `-1`, the server waits forever.
`httpConnectionInfo`	`Object`	HTTP Connection Info object.

Returns

Type	Description
`Event`	An event as a response to the request

Sending a Request via Proxy Server

To send requests via a proxy server, use this function.

Signature

```
Object setProxy(Object connectionInfo, String proxyHost, int proxyPort)
```

Description

This function updates the `HttpConnectionInfo` object with the proxy server details.

For secure communication, use the function as follows for both Synchronous and Asynchronous requests:

```
Object connectionInfo= HTTP.ConnectionInfo.createHTTPConnectionInfo
(true);
HTTP.setSecureInfo(Object connectionInfo, String sslProtocol, Object
clientIdKeyStore, String clientIdPassword, Object trustedCertsKeystore,
String trustedCertsPassword, boolean verifyHostName)
HTTP.setProxy(Object connectionInfo, String proxyHost, int proxyPort)
```

Parameters

Name	Type	Description
`connectionInfo`	`Object`	HTTP Connection Info object.
`proxyHost`	`String`	Proxy host.
`proxyPort`	`Integer`	Proxy port.

Returns

Type	Description
`Object`	Connection info object.

Disabling HTTP Methods at Channel and Destination Level

If required, you can disable specific HTTP methods on the native HTTP channel.
The HTTP methods can be disabled at the channel level and the destination level. You can also disable them at both levels and use the combination according to your requirement. For example, you can disable the GET method for a channel (TestChannel) and for the same channel (TestChannel) disable the POST method for a destination (getData). Then for that destination (getData) the GET and POST methods do not work, while for other destinations only the GET method is disabled.

```
Channels.TestChannel.disableHttpMethods=GET
Channels.TestChannel.getData.disableHttpMethods=POST
```

Procedure

1. Add the `disableHttpMethods` system property at the channel level, in the system TRA file or the project's CDD file, to disable the specified HTTP methods for all destinations of the channel.

   <*Channel Folder*>.<*Channel Name*>.`disableHttpMethods`

   Note: If you are using TIBCO BusinessEvents Studio, prefix the channel folder name with `java.property`.

   For example, using the CLI,

   ```
   Channels.TestChannel.disableHttpMethods=GET
   ```

   For example, using TIBCO BusinessEvents Studio,

   ```
   java.property.Channels.TestChannel.disableHttpMethods=GET
   ```

2. Add the `disableHttpMethods` system property at the destination level , in the system TRA file or the project's CDD file, to disable the specified HTTP methods for that destination of the channel.

    - Regular destination - *<Channel Folder>*.*<Channel Name>*.*<Destination Name>*.`disableHttpMethods`
    - Pageflow destination - *<Channel Folder>*.*<Channel Name>*. *<Context Path>*.`disableHttpMethods`

    Note: If you are using TIBCO BusinessEvents Studio, prefix the channel folder name with `java.property`.

    For example, using the CLI

    ```
    Channels.TestChannel.getData.disableHttpMethods=POST
    ```

    For example, using TIBCO BusinessEvents Studio,

    ```
    java.property.Channels.TestChannel.getData.disableHttpMethods=POST
    ```

    Note: Security constraints are added to the `web.xml` file of the WebStudio. If you use OPTIONS, TRACE, and TRACK methods, a 403 forbidden error occurs.

Configuring TIBCO BusinessEvents as a SOAP Server and Client

Using an HTTP channel and a destination configured to use a SOAP Serializer, TIBCO BusinessEvents can act as a web services platform, sending and receiving SOAP requests.

You can configure the project manually or you can use a WSDL import utility, which simplifies configuration.

This release supports SOAP version 1.1. Some understanding of SOAP protocol is required to work with this feature. See http://www.w3.org/TR/soap/ for details.

Tip: When TIBCO BusinessEvents acts as a web service server, clients of the service can view the exported concrete WSDL for the web service using the URL to the document root folder. You must place the WSDL file in that location.

Overview of SOAP Related Resources

Project resources must be configured so that TIBCO BusinessEvents can send and receive SOAP messages.

The resources to configure are as follows:

SOAP Destinations

When a correctly configured HTTP destination receives a SOAP request, the SOAP serializer deserializes the SOAP message to its corresponding event. That event has to be inherited from a SOAPEvent. The event payload contains the SOAP envelope.

SOAP Events

To create a SOAP event, create a SimpleEvent that inherits a SOAPEvent. It makes event configuration easier. Its payload has a `message` root element having an `Envelope` child element. The root element contains `Header` and `Body` elements, and the Body element has a `Fault` element. You can further configure these elements using the payload editor.

SOAP messages (and events) can have attachments.

Note: For outbound SOAP events, ensure that the default destination is specified. If it is not specified, an incorrect SOAP message, with “Message” as its root instead of “Envelope”, is sent out.

Also, ensure that the default destination for the SOAP event uses `com.tibco.cep.driver.http.serializer.SOAPMessageSerializer`.

SOAP Encoding

Only the SOAP document/literal form of encoding is supported.

Rules and Rule Functions

A rule in the project accesses the SOAP event payload as needed using the mapper or SOAP catalog functions. The rule puts the SOAP response into the payload of a

SOAPEvent that is sent to the client. The SOAP serializer sends the SOAP response to the client.

SOAP Catalog Functions

SOAP catalog functions enable you to access and process the contents of the following elements in the event payload of an incoming SOAPEvent, and add elements to the outbound SOAPEvent:

- Header
- Body
- Fault
- Attachment

Mapping of SOAP Request URI to Destination

For SOAP requests, the header property `SOAPAction` represents the destination name, and the `requestURI` represents the channel URI. TIBCO BusinessEvents combines the `SOAPAction` and `requestURI` to create a destination URI. It maps the request with a destination having same URI as the newly created URI.

For example, if the `requestURI` of a request is `/QueryBooks` and the value of `SOAPAction` is `QueryBooksByAuthor`, then TIBCO BusinessEvents engine maps the request with a destination having `/QueryBooks/QueryBooksByAuthor` URI if it exists.

Manually Creating Resources to Work with SOAP Services

This section explains how you can manually create and configure project resources to work with SOAP services.

The section Creating Resources Using the WSDL Import Utility explains how to use the WSDL import feature to create the resources.

Procedure

1. **Configure the HTTP Channel and SOAP Destination**

   Configure an HTTP channel, an HTTP connection resource for the channel. Connection configuration is the same for SOAP and HTTP channels, except for the serializer. For SOAP destinations, use the following serializer:

   com.tibco.cep.driver.http.serializer.SOAPMessageSerializer

   After you complete Add a SOAPEvent (and Other Ontology as Needed), set that `SOAPEvent` as the default event for the destination.

2. **Add a SOAPEvent (and Other Ontology as Needed)**

   Configure the SOAPEvents that receives the SOAP requests and send out SOAP responses. Also configure any other ontology as needed. SOAPEvent is an event type provided with TIBCO BusinessEvents. However it is created in a two-step manner:

   a. Add a simple event in the usual way.

   b. In the Inherits From field, select **SOAPEvent**.

   c. Set the default destination to the destination you created in Configure the HTTP Channel and SOAP Destination.

      The payload of a SOAPEvent is automatically configured with the structure of a SOAP message. It has Header and Body elements, and within the Body element, a Fault element. You can further configure the Header and Body elements using the payload editor.

3. **Configure Rules and Rule Functions using SOAP Functions**

   In general the procedure of serving requests and sending requests to other servers is the same for SOAP services as for other HTTP interactions. One rule function is used as the event preprocessor.

   In addition, you must configure rules and rule functions to access information from the SOAP messages in the inbound events, and to populate outbound events with SOAP message details. Set two SOAPevents you configured in Add a SOAPEvent (and Other Ontology as Needed) as input and return arguments of the preprocessor. See Parsing and Building SOAP Messages for details.

4. **Configure an Event Preprocessor**

   Open the project CDD for editing, and configure an event preprocessor for the destination you configured in Configure the HTTP Channel and SOAP Destination. Use the rule function you configured in Configure Rules and Rule Functions using SOAP Functions.

Creating Resources Using the WSDL Import Utility

WSDL (Web Services Description Language) is an XML format for describing web services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

Note:
- Only document style WSDLs with literal encoding are currently supported.
- Only In-Only and In-Out Message Exchange Patterns (MEPs) are currently supported.

TIBCO BusinessEvents can parse an imported WSDL file and create the required project artifacts based on its contents: the channels and destinations (concrete WSDLS only), events, rules, and rule functions.

Both abstract and concrete WSDL files can be used. When you import an abstract WSDL, channel and transport information is not available and you must create those resources manually. Concrete WSDLS contain information that is used to create these resources automatically.

You then implement the newly created rules and rule functions to provide the expected behavior for the web service described in the WSDL file.

Importing a WSDL File

First, import the WSDL file into the project. Then you can use the WSDL import utility.

Procedure

1. Import the WSDL file into the project, as follows.
    a. In Studio Explorer, right-click the project folder (or the project root) where you want to store the WSDL file and select **Import**. You see the Import wizard.
    b. At the Select dialog, select **General > From File System**. Click **Next**.
    c. At the File system dialog, click **Browse**, select the directory that contains the WSDL file you want to use, and click **OK**.
    d. The File system dialog displays the directory in the left panel and its files in the right panel. Select the checkbox to select the file you want to use. Click **Finish**.

       The WSDL file is imported into the project.
2. Create project artifacts from the WSDL file as follows.
    a. In Studio Explorer, right-click anywhere in the project and select **Import**. You see the Import wizard.
    b. At the Select dialog, select **TIBCO BusinessEvents > WSDL**. Click **Next**.
    c. At the WSDL Import dialog, click **Browse** and select the WSDL file you imported in *Step 1*, then click **OK**.
    d. The WSDL Import dialog displays again, showing the selected WSDL. Click **Finish**.

       If the WSDL file you select is an abstract WSDL, you see a warning letting you know that a channel will not be created. See the section introduction for more details about abstract WSDL files.

       The project artifacts are created within a folder named using part of the WSDL file name.
3. Save the project.

Reimporting a WSDL File

When you import a WSDL file, TIBCO BusinessEvents creates the project artifacts. If any changes are made to the WSDL file after the import, you must perform the following steps to use the updated WSDL:

Note: Reimporting a WSDL file fails if the project artifacts are already present.

Procedure

1. Navigate to the project directory and rename the WSDL file.
2. Copy the updated WSDL file to the same directory. Ensure that the name of the WSDL file is the same.
3. In Studio Explorer, select the WSDL file and then select **Edit > Delete** from the top menu to delete the WSDL file.
4. Refresh the project (keyboard shortcut **F5**) and then select **Project > Clean** from the top menu.
5. If you have configured mappings in the project, check for errors caused by broken links in the mapper.
6. Open the mapper and click the **Mapper Check and Repair** button and check only the mappings in red.
7. Click **Ok** to fix the mapping.
8. If there are new or changed elements, you may have to map the new or changed elements manually.

Exporting Resources as a WSDL File

The WSDL export utility allows you to export rules and rule functions that have a SOAPEvent as a parameter, as a WSDL file.

Note:

- TIBCO BusinessEvents supports WSDL 1.1 specification. See the following page for more details:

  http://www.w3.org/TR/wsdl

- WSDL Filenames must conform to the `NCName` datatype. See the following page for more details:

  http://www.w3.org/TR/REC-xml-names/#NT-NCName

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

  http://www.w3.org/Submission/japanese-xml/

The following table shows how related project resources are exported as a WSDL file.

Exporting Project Artifacts as WSDL

Project Folders and Resources	WSDL Artifact after Exporting
RuleFunction Use the rule function specified as the input event's default destination as the SOAPEvent's preprocessor. That rule function must have the "SOAPEvent for input" as the input and the "SOAPEvent for output" as the return type.	Forms the `<wsdl:operation>` operation.
SOAPEvent • `SOAPEvent` for the WSDL input argument OR • `SOAPEvent` for the WSDL output argument Use the event you configured in Add a SOAPEvent (and Other Ontology as Needed) .	Creates a message. The contents of **Envelope > Header > Body** become message parts.
HTTP Channel Channel with HTTP as the **Driver**, Resource as the **Method of Configuration**, with a pointer to the HTTP Connection.	The URI of the channel forms the `<soap:address>` URI.
HTTP Connection for HTTP Channel Host and port are used.	The name of the connection forms the `<wsdl:port>`. Host and port of the connection form the `<soap:address>` host and port.
Destination for HTTP Channel With `SOAPMessageSerializer` as the default serializer. The destination used is the default destination of the SOAPevent.	Forms the `soapAction` attribute of the operation. The destination name becomes `soapAction` specified for that particular operation.

Project Folders and Resources	WSDL Artifact after Exporting
CDD In the **Agent Classes** tab, Input Destination Collections list, the **Preprocessor** specified for the input SOAPEvent's default destination. Place the input destination directly under the agent class input destination collections, and not under **Collections > Input destination**.	Associates `soapAction` to the operation in the HTTP binding.

Exporting a Rule or Rule Function as a WSDL

Procedure

1. Right-click the project, and click **Export.**
2. In the Export dialog, under TIBCO BusinessEvents, select **WSDL** and click **Next**.
3. Select the WSDL location by specifying the path, and type a valid `NCName` for the WSDL file.
4. Select the CDD file of the project.
5. Click **Finish**.
6. Save the project.

Parsing and Building SOAP Messages

This section explains how to use the SOAP functions to parse information from incoming requests, and to construct outgoing messages (responses and requests) in your rules and rule functions.

This manual does not explain how to work with the SOAP protocol. For example, you should understand how the `Actor` and `MustUnderstand` attributes in SOAP headers are used to process the message as it passes from its originator, through intermediary applications, to its ultimate destination.

Working with Incoming SOAP Messages (Event Payloads)

An incoming SOAP message can be a request, or a response, depending on whether TIBCO BusinessEvents is acting as the server or the client. TIBCO BusinessEvents can also act as an intermediate node along the path of a SOAP message to its ultimate destination. This section explains how to parse information out of the incoming event payload, which contains the SOAP message.

To Parse the SOAP Envelope

```
String getEnvelope(SimpleEvent inSoapEvent)
```

Given a request SOAPEvent, this function returns the SOAP envelope in the request event payload, for example:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sch="http://www.tibco.com/schemas/SoapOverHttp/Schema/Schema.xsd"
>
  <soapenv:Header/>
    <soapenv:Body>
      <sch:root>
        <sch:First>1</sch:First>
        <sch:Second>2</sch:Second>
      </sch:root>
    </soapenv:Body>
  </soapenv:Envelope>
</message>
```

Parsing (and Optionally Removing) Headers and Header Attributes

```
String[] getHeaders(SimpleEvent inSoapEvent, String actor, Boolean
removeHeaders)
```

If the `actor` parameter has a null value then all the immediate children of the Header element are retrieved:

```
getHeaders(inSoapEvent,null,false)
```

Otherwise, the header specified by the `actor` attribute is retrieved. For example, given this Headers element in a SOAP event payload:

```
    <soapenv:Header>
      <t:user xmlns:t="http://schemas/xml.com"
soapenv:mustUnderstand="true"
            soapenv:actor="http://localhost:9090/Service">jon</t:user>
      <t:user_surname xmlns:t="http://schemas/xml.com"
                    soapenv:mustUnderstand="true"
                    soapenv:actor="http://localhost:9090">smith</t:user_
surname>
    </soapenv:Header>
```

If you specify the following:

```
getHeaders(inSoapEvent,"http://localhost:9090/Service",false)
```

To remove the specified header part or parts, set the final parameter to `true`. (The SOAP specification states that if a header is processed it should be removed. You would remove a header if TIBCO BusinessEvents is acting as an intermediary node and the request created using the SOAP functions are sent on to another server.)

Then the first Headers element is returned:

```
      <t:user xmlns:t="http://schemas/xml.com"
soapenv:mustUnderstand="true"
            soapenv:actor="http://localhost:9090/Service">jon</t:user>
```

You can also retrieve the attributes of a SOAP Header element:

`String[] getSOAPHeaderAttribute(SimpleEvent` *`inSoapEvent`*`, int index, String attribute)`

You can also remove all or selected headers using one of these functions:

```
removeHeaderPart()
removeHeaderParts()
```

Parsing the SOAP Body (SOAPBodyParts)

Two functions are available for getting SOAP body parts.

```
String[] getAllSOAPBodyParts(SimpleEvent inSoapEvent)
String[] getSOAPBodyParts(SimpleEvent inSoapEvent, String name, String
namespace)
```

The `getAllSOAPBodyParts()` function simply returns all SOAP body parts.

Using the `getSOAPBodyParts() function` you can specify which parts are of interest. Given a body part name and a namespace for a specified SOAPEvent, it returns a String array of matching SOAP body parts in serialized form. Name and namespace parameters cannot be null.

Example

Given this function:

```
String[] body_part= getSOAPBodyParts
(soapeventin,"root",http://www.tibco.com/schemas/SoapOverHttp/Schema/Sch
ema.xsd)
```

And this `soapeventin` event payload:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
  <soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sch="http://www.tibco.com/schemas/SoapOverHttp/Schema/Schema.xsd"
xmlns:temp="http://temp/">
  <soapenv:Header/>
    <soapenv:Body>
      <sch:root>
        <sch:First>1</sch:First>
        <sch:Second>2</sch:Second>
      </sch:root>
      <sch:parent2>
        <sch:child1>3</sch:child1>
        <sch:child2>4</sch:child2>
      </sch:parent2>
    </soapenv:Body> </soapenv:Envelope>
```

You would get this as the SOAP body part:

```
body_part[0]=
<?xml version="1.0" encoding="UTF-8"?>
<ns0:root
xmlns:ns0="http://www.tibco.com/schemas/SoapOverHttp/Schema/Schema.xsd">
  <ns0:First>1</ns0:First>
  <ns0:Second>2</ns0:Second>
</ns0:root>
```

Parsing Attachments

The following functions enable you to work with SOAP attachments in the request message:

```
getNumberOfAttachments(SimpleEvent inSOAPEvent)
getAttachmentContentID(SimpleEvent inSOAPEvent, int Index)
getAttachmentContentType(SimpleEvent inSOAPEvent, int Index)
getAttachmentContent(SimpleEvent inSOAPEvent, int Index)
getAttachmentContentByContentID(SimpleEvent inSOAPEvent, string contentID)
```

The content ID is the attachment identifier. You can select which attachment to work with using its index position. First get the count of the attachments using `getNumberOfAttachments()`. Then using the index, you can get the content ID and content type, as well as the attachment content itself.

The content is returned in byte form, so after you get the content, you must then use other functions to make the content human-readable.

For an example, see http://www.w3.org/TR/SOAP-attachments#SOAPReferenceToAttachements.

Parsing SOAP Fault XML Nodes

The following functions enable you to work with the standard SOAP Fault XML nodes from the payload of a SOAP event:

```
getFault (SimpleEvent, soapEvent)
getFaultActor (SimpleEvent, soapEvent)
getFaultCode (SimpleEvent, soapEvent)
getFaultString (SimpleEvent, soapEvent)
```

Working with Outgoing SOAP Messages (Event Payloads)

You can add header parts, body parts, fault parts and attachments to the outgoing SOAP message (whether it is a response or a request).

The signatures of the relevant functions are as follows:

```
addHeaderPart(SimpleEvent outSOAPEvent, String headerPartXml)
addSOAPBodyPart(SimpleEvent outSOAPEvent, String bodyXML)
addSOAPHeaderAttribute(SimpleEvent outSOAPEvent, int index, String
attribute, String value)
addFaultPart(SimpleEvent outSOAPEvent, String faultCode, String faultMessage,
String faultActor, String faultDetailString)
addAttachment(SimpleEvent outSOAPEvent, String contentID, String content, String
contentType, String contentEncoding)
```

TIBCO BusinessEvents adds each type of fragment to the appropriate part of the event payload: header, body, or fault. The fragments must be well-formed XML. You can also add attachments.

For example, to add a body part containing information for a response you would include all the required details including any namespace information:

```
SOAP.addSOAPBodyPart(outSOAPEvent,"<ns0:BookStore
xmlns:ns0=\"http://www.abc.com/xsd/books\"><ns0:Book><ns0:Author>J.K.Row
ling</ns0:Author></ns0:Book></ns0:BookStore>");
```

The specified body part is added to the correct place in the outline structure of the SOAP message, which is provided by the SOAPEvent. The resulting payload would look similar to the following:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns0:BookStore xmlns:ns0="HTTP://www.abc.com/xsd/books">
      <ns0:Book>
        <ns0:Author>J.K.Rowling</ns0:Author></ns0:BookStore>"
      </ns0:Book>
    </ns0:BookStore>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Understanding the WSDL to Project Resource Mapping

A WSDL file describes a web service. The WSDL Import utility imports a WSDL file and generates TIBCO BusinessEvents project artifacts using elements in the WSDL. TIBCO BusinessEvents can import abstract and concrete WSDL files. The source of the WSDL could be, for example, an ActiveMatrix BusinessWorks `SOAPRequestReply` activity.

Note: For WSDL import to be successful, all names in the WSDL must conform to TIBCO BusinessEvents folder and entity naming requirements.

Example WSDL

The table following this example shows which WSDL elements and attributes are used to create TIBCO BusinessEvents project artifacts.

Elements and attributes used in the import are highlighted in bold text. Differences between import from abstract and concrete WSDL files are also highlighted. See Example Project Folder Structure for more details.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Created by TIBCO WSDL-->
<wsdl:definitions omitted to keep the example short>
   <wsdl:types>
      <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.books.org" elementFormDefault="qualified"
attributeFormDefault="unqualified"
targetNamespace="http://www.books.org">
. . . . . . Elements omitted
      </xsd:schema>
   </wsdl:types>
. . . . . . Elements omitted

Note:In an abstract WSDL the following elements are used in the import. However in a concrete WSDL, the
<wsdl:binding> elements are used instead.  <wsdl:portType name="GetBookPortType">
      <wsdl:operation name="GetBook">
         <wsdl:input message="tns:GetBookRequestMessage"/>
         <wsdl:output message="tns:GetBookResponseMessage"/>
      </wsdl:operation>
   </wsdl:portType>
```

```
Note: This is a concrete WSDL example, so the <wsdl:binding> elements are used in the import. (In an abstract
WSDL, the <wsdl:portType> element contents are used instead.)<wsdl:binding
name="getBookBinding" type="tns:GetBookPortType">
      <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="GetBook">
         <wsdl:documentation>The operation has no documentation
         </wsdl:documentation>
         <soap:operation style="document"
soapAction="/Service/getBook"/>
         <wsdl:input>
            <soap:body use="literal" parts="part1"/>
            <soap:header use="literal"
                     message="tns:TransactionRecordMessage"
part="user"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal" parts="part1"/>
            <soap:header use="literal"
                     message="tns:TransactionRecordMessage"
part="transactionID"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="getBook">
      <wsdl:port name="getBookHttpPort" binding="tns:getBookBinding">
         <soap:address location="http://ACME:9090/Service/getBook"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Example Project Folder Structure

Suppose you import the example WSDL(Web Service Description Language) above into a Studio project called Library. The imported and generated project artifact names would appear as shown in the Project Folders and Resources column in the table below. Folders that are added by TIBCO BusinessEvents are shown in bold. The example is a concrete WSDL. In the WSDL Source column, the source of folder and resource names is given for abstract as well as concrete WSDL sources.

Imported WSDL Project Artifacts

Project Folders and Resources	Project Resource Type	WSDL Source
`Library`	Project root folder	N/A
`getBook/`	Folder	For concrete WSDLS: `<wsdl:service>` For abstract WSDLS: `<wsdl:service>` is not present, so the folder structure starts from the folder created from `<wsdl:portType>`.
`GetBookPortType/`	Folder	For abstract and concrete WSDLs: `<wsdl:portType>`
`getBook/Events/`	TIBCO BusinessEvents folder	
`GetBookRequestMessage`	Event	`<wsdl:input>` For abstract WSDLs, in `<wsdl:portType>` section. For concrete WSDLs, in `<wsdl:binding>` section.
`GetBookResponseMessage`	Event	`<wsdl:output>` For abstract WSDLs, in `<wsdl:portType>` section. For concrete WSDLs, in `<wsdl:binding>` section.
`getBook/RuleFunctions/`	TIBCO BusinessEvents folder	

Project Folders and Resources	Project Resource Type	WSDL Source
`GetBook`	Rule function	`<wsdl:operation>` For abstract WSDLs, in `<wsdl:portType>` section. For concrete WSDLs, in `<wsdl:binding>` section.
`getBook/Rules/`	TIBCO BusinessEvents folder	
`GetBookPortType`	Folder	Abstract WSDL: `<wsdl:portType name>` Concrete WSDL: `<wsdl:binding type>`
`GetBook`	Rule	`<wsdl:operation>` For abstract WSDLs, in `<wsdl:portType>` section. For concrete WSDLs, in `<wsdl:binding>` section.
Import from Concrete WSDLs Only If the import is from a concrete WSDL, the HTTP Connection resource, Channel resource, and Destination resource are added using the details in the `<wsdl:service>` section of the WSDL. If the import is from an abstract WSDL, you must create these resources manually.		
`getbook/Transports/`	TIBCO BusinessEvents folder	
`getBookHTTPPort`	HTTP Connection	`<wsdl:port>` The host and port come from the `<soap:address location>`

Project Folders and Resources	Project Resource Type	WSDL Source
`Service/`	TIBCO BusinessEvents folder	
`getBook`	Channel See Channel Folders.	`<soap:address location>` (from the last part of the location URL)
`Service_getBook`	Destination See Destination Names	`<soap:operation soapAction>`

How Project Artifacts are Named

Channel Folders

Given a concrete WSDL, folders are created for all the elements after the `port`, up to the last forward slash of the location URL. The text after the last forward slash is the channel name. For example, given the following location URL:

```
http://ACME:9090/Service/Trial/getBook
```

The folder structure would be `/Service/Trial` and the channel name would be `getBook`.

Destination Names

In a concrete WSDL, the `SOAPAction` attribute of a `<soap:operation>` element specifies the URL of a destination. It also becomes the destination name. Forward slashes (`/`), colons (`:`), and periods (`.`) are converted to underscore characters (`_`) to form the name. For example:

```
http://www.acme.com/TNT/webservices/getByteField
```

Becomes:

```
http___www_acme_com_TNT_webservices_getByteField
```

Rules and Rule Functions

For each operation, the import utility creates a rule and a rule function. The rule has no body. The rule functions have `SoapEventOut` as the return type. Null value is returned by default.

For example, the `GetBook` operation becomes a `GetBook` rule in the `GetBookPortType` folder which is in the `Rules` folder, and also a `GetBook` rule function in the `Rulefunctions` folder.

You implement the rules and rule functions in your project according to the web service you want to implement.

Events

The `<wsdl:input>` element becomes a request event and the `<wsdl:output>` element becomes a response event. Each event type inherits from the `SoapEvent` event type.

Event names come from the `message` attributes. In the example, the request event is `GetBookRequestMessage` and the response event is `GetBookResponseMessage`.

Faults

Faults specified in a WSDL are used in the outbound SOAP event, as the Fault element.

JMS Channels

This chapter provides additional information about working with JMS channels.

Selecting a JMS Serializer

When working with the JMS channels, choose the serializer that handles the JMS message types that will be sent to the destination you are configuring.

Which Serializers to Use for JMS Message Types

Message Type	BytesMessage Serializer/ UtfBytesMessage Serializer	TextMessage Serializer	MessageWithNoBody (1)
`Message`	Supported	Supported	Supported
`BytesMessage`	Supported	(N/A)	Supported
`MapMessage`	Supported	Supported	Supported
`ObjectMessage`	Not Supported	Not Supported	Not Supported
`StreamMessage`	Not Supported	Not Supported	Not Supported
`TextMessage`	(N/A)	Supported	Supported

(1) MessageWithNoBody corresponds to the BasicMessageSerializer option in TIBCO BusinessEvents Studio.

For `MapMessage` messages, you create properties whose names match the message keys.

Payload Handling

All serializers support reading and writing application header properties and JMS header properties.

The difference between the serializers is in how they handle payloads.

BytesMessageSerializer

The `BytesMessageSerializer` decodes the body as a sequence of bytes and parses them to create an XML structure according to the payload definition in the event.

The `BytesMessageSerializer` is used to receive JMS messages coming in as a stream of uninterrupted bytes in the message body.

Typically, `BytesMessageSerializer` is a low-level serializer and can have performance implications. Choose the serializer for a defined message type (MapMessage, TextMessage, and so on) that most closely matches the expected usage.

For incoming messages of type JMS BytesMessage, the serializer converts the message bodies to event payloads. The payloads are XML String type, but are not human-readable. However, you can access them using XPath functions. For outgoing events, the serializer converts XML payloads to JMS BytesMessage message bodies.

The `BytesMessageSerializer` class is the default serializer.

UtfBytesMessageSerializer

The `UtfBytesMessageSerializer` is similar to the `BytesMessageSerializer` except that it serializes the payload using `writeUTF()` instead of `writeBytes()`, and deserializes the payload using `readUTF()` instead of `readBytes()`.

For outgoing events, the serializer converts XML payloads to JMS BytesMessage message bodies.

TextMessageSerializer

The `TextMessageSerializer` decodes the text from the message as an XML string.

For incoming messages, the `TextMessageSerializer` serializer converts JMS `TextMessage` using XPath functions. For outgoing events, the serializer converts XML payloads to JMS TextMessage messages.

MessageWithNoBody

The `MessageWithNoBody` serializer does not serialize or deserialize the payload.

This serializer corresponds to the BasicMessageSerializer option in TIBCO BusinessEvents Studio.

For outgoing events, the serializer converts the payload to messages of type Message.

See JMS Header Properties in Incoming and Outgoing Messages in *TIBCO BusinessEvents Developer Guide*.

Creating Unique JMS DurableSubscriber Name Properties

For destinations that are JMS topics, if you provide a `DurableSubscriber Name` when you configure the destination resource, the destination becomes a JMS durable topic subscriber with the specified name. This section explains how you can ensure that the `DurableSubscriber Name` value is unique.

Note:

- When using topic destination with a durable name in applications using In Memory OM and fault tolerance, do not provide a value for the Client ID setting and do not check the Auto-generate Client ID checkbox in the JMS shared resource.
- Do not use durable topic destinations for multi-agent applications, even when only one agent instance is active at a time (that is, even when **Agent Classes > AgentClassName > Max Active** is set to 1). Instead, use queue destinations.

The value of the `DurableSubscriber Name` property can be any unique string and can include any global variables. TIBCO BusinessEvents provides a set of case-variables that produce a unique `DurableSubscriberName` string:

```
%%Deployment%%:%%EngineName%%:%%SessionName%%:%%ChannelURI%%:%%Destinati
onName%%
```

The first variable *%%Deployment%%* is a standard TIBCO global variable. The other three are only for use with the `DurableSubscriberName` property within TIBCO BusinessEvents. For details see Variables to Use with DurableSubscriberName.

Variables to Use with DurableSubscriberName

Do not attempt to use `%%EngineName%%`, `%%SessionName%%`, `%%ChannelURI%%`, or `%%DestinationURI%%` in any area of TIBCO BusinessEvents software except the `DurableSubscriberName` property.

DurableSubscriberName Variables

Variable	Description
`%%EngineName%%`	The name of the TIBCO BusinessEvents engine. The name used is established using a series of checks. See Determining the Engine Name in *TIBCO BusinessEvents Administration* for details.
`%%SessionName%%`	The name of the agent class that is associated with the durable subscriber. Agent classes are defined in the CDD resource. See Collections, Agent Classes, and Processing Units in *TIBCO BusinessEvents Configuration Guide* for details.
`%%ChannelURI%%`	The path to the channel within the TIBCO BusinessEvents project: `/`*folder*`/ . . . /`*channel_name*
`%%DestinationName%%`	The name of the TIBCO BusinessEvents destination, within the channel specified in `%%ChannelURI%%`.

JMS Message Acknowledgement Modes

JMS channels support connection to TIBCO Enterprise Message Service destinations. The default acknowledgement mode is `EXPLICIT_CLIENT_ACKNOWLEDGE`.

JMS Message Acknowledgement Modes

Mode	Description
`AUTO_ACKNOWLEDGE`	Specifies that the session is to automatically acknowledge consumer receipt of messages when message processing has finished.
`CLIENT_ACKNOWLEDGE`	Specifies that the consumer is to acknowledge all messages that have been delivered so far by the session. When using this mode, it is possible for a consumer to fall behind in its message processing and build up a large number of unacknowledged messages. See Using CLIENT_ACKNOWLEDGE Mode with WebSphere MQ and Cache-Aside for required configuration for WebSphere MQ when cache-aside database write strategy is used.
`DUPS_OK_ACKNOWLEDGE`	Specifies that the session is to lazily acknowledge the delivery of messages to the consumer. “Lazy” means that the consumer can delay acknowledgement of messages to the server until a convenient time; meanwhile the server might redeliver messages. This mode reduces session overhead. However, should JMS fail, the consumer may receive duplicate messages.
`EXPLICIT_CLIENT_ACKNOWLEDGE` (TIBCO Proprietary)	TIBCO Enterprise Message Service extension to JMS acknowledge modes. This is the default acknowledgement mode. `EXPLICIT_CLIENT_ACKNOWLEDGE` is like `CLIENT_ACKNOWLEDGE` except it acknowledges only the individual message, rather than all messages received so far on the session. One example of when `EXPLICIT_CLIENT_ACKNOWLEDGE` would be used is when receiving messages and putting the information in a database. If the database insert operation is slow, you might want to use multiple application threads, all doing simultaneous inserts. As each thread finishes its insert, it can use `EXPLICIT_CLIENT_ACKNOWLEDGE` to acknowledge only the message that it is currently working on.

Mode	Description
`EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE` (TIBCO Proprietary)	TIBCO Enterprise Message Service extension to JMS acknowledge modes. `EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE` mode is like `TIBEMS-DUPS-OK-ACKNOWLEDGE` except it "lazily" acknowledges only the individual message, rather than all messages received so far on the session.
`NO_ACKNOWLEDGE` (TIBCO Proprietary)	TIBCO Enterprise Message Service extension to JMS acknowledge modes. Suppresses the acknowledgement of received messages. After the server sends a message to the client, all information regarding that message for that consumer is eliminated from the server. Therefore, there is no need for the client application to send an acknowledgement to the server about the received message. Not sending acknowledgements decreases the message traffic and saves time for the receiver, therefore allowing better utilization of system resources. **Note:** • Sessions created in `NO_ACKNOWLEDGE` receipt mode cannot be used to create durable subscribers. • Also, queue receivers on a queue that is routed from another server are not permitted to specify `NO_ACKNOWLEDGE` mode.

Using CLIENT_ACKNOWLEDGE Mode with WebSphere MQ and Cache-Aside

The cache-aside database write strategy is multi-threaded. However, when WebSphere MQ messages are sent using `CLIENT_ACKNOWLEDGE_MODE`, each message must be handled from start to finish using a single thread. To address this issue, follow these steps:

Procedure

1. Define the destination using Caller’s Thread in the Threading Model setting (in the **CDD Collections** tab or **Agent Classes** tab).

2. To ensure sequential operations set the following property in the CDD file at the appropriate level:

   ```
   Agent.agentClassName.enableParallelOps=false
   ```

   Setting this property to false means that all post-RTC operations are done on a single thread.

When JMS Messages are Acknowledged

When TIBCO BusinessEvents acknowledges JMS messages depends on the JMS acknowledgment mode, time to live (TTL) setting, and object management (OM) type, as shown in the table below.

When JMS Messages are Acknowledged

JMS Acknowledgement Mode	OM Type	Acknowledged
`AUTO_ACKNOWLEDGE DUPS_OK_ACKNOWLEDGE`		On receipt
`NO_ACKNOWLEDGE`		Never
`CLIENT_ACKNOWLEDGE EXPLICIT_CLIENT_ACKNOWLEDGE EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE`	Cache	Post RTC
	In Memory	When retracted (when the event is deleted (using the `Event.consumeEvent` function) or when the event reaches the end of its time to live (TTL) period.

For more details about message acknowledgment with Cache OM, refer to the Post-RTC and Epilog Handling and Tuning Options section in *TIBCO BusinessEvents Architect Guide*.

Using JMS Header Properties in Incoming and Outgoing Messages

Information in this section assumes you are familiar with JMS and its header properties. Consult your JMS provider documentation for information. This section explains only how TIBCO BusinessEvents supports use of these properties.

Setting Certain Header Properties in Destinations

In the JMS destination Configuration section, you can configure the following three header properties:

- DeliveryMode (`JMSDeliveryMode`)
- Priority (`JMSPriority`)
- TTL (`JMSExpiration`)

> **Note:** JMS header properties defined in events take precedence over properties defined in destinations.

Setting Header Properties Using Header Properties from Incoming JMS Messages

You can configure events created from incoming JMS messages to have properties that match the JMS header properties. You can then use those event properties to set JMS header properties in outgoing messages.

These event properties must match the JMS header fields. You must use the names as shown in JMS Header Field Names. You only have to configure event properties for those fields that you want to use. Incoming JMS message header properties will then populate the corresponding TIBCO BusinessEvents event properties.

Setting JMS Header Properties in Outgoing JMS Messages Using Event Properties

Similarly outgoing JMS message header properties will be populated by the corresponding TIBCO BusinessEvents event properties.

Note that the `JMSMessageID` and `JMSTimeStamp` properties are generated when the message is sent. You cannot set these properties manually.

See How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages for special handling of the `JMSReplyTo` header.

JMS header properties defined in events take precedence over properties defined in destinations.

Tip: You can add the JMS properties to the Base event in your project so that the properties are inherited by all other events.

See JMS Header Field Names for details about all properties.

How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages

Note: TIBCO BusinessEvents cannot act as a client in a JMS request-response scenario because TIBCO BusinessEvents currently cannot dynamically create a destination to listen for JMS messages.

If an event has a string type property named `JMSReplyTo` (case sensitive), TIBCO BusinessEvents reads this event property value as a JMS queue or topic name, according to the event's default destination type. TIBCO BusinessEvents looks up `javax.jms.Destination` on the connected JMS server using this queue or topic name. If TIBCO BusinessEvents cannot find one, it creates a `javax.jms.Destination` using the given queue or topic name. TIBCO BusinessEvents then sets the `JMSReplyTo` header property of the outgoing JMS message using this destination.

Note: If you use the catalog function `Event.replyEvent(`*`requestEvent`*`,` *`replyEvent`*`)` during the RTC in which *`requestevent`* is received, then *`replyevent`* is sent to the destination in the `JMSReplyTo` header property of the JMS message associated with the *`requestevent`*.

JMS Header Field Names

The table below shows the names that you must use to define event properties corresponding to JMS header field names, as well as some details about the purpose of each property. The property names are not case-sensitive.

Important: You can use global variables in all the advanced properties. See Working with Global Variables for more details.

JMS Header Field Names

Field name	Type	Description
`JMSDestination`	Object	The destination (queue or topic) to which the message is sent.
`JMSDeliveryMode`	Integer	The delivery mode is specified by the sender. This property instructs the server about the persistent storage for the message. Value can be: 2-interpreted as PERSISTENT (default). 1-interpreted as NON-PERSISTENT. 22-interpreted as RELIABLE. This value is an extension to the standard used in TIBCO Enterprise Message Service. The integer values are interpreted as the text names of the delivery modes. You can also set a delivery mode for a destination. See Setting Certain Header Properties in

Field name	Type	Description
		Destinations .
`JMSExpiration`	Long	The length of time (in milliseconds) that the message lives before expiration. If set to 0, the message does not expire. You can also set an expiration (TTL) for a destination. See Setting Certain Header Properties in Destinations .
`JMSPriority`	Integer	The message priority, a numerical value between 0 and 9. Larger numbers represent a higher priority. You can also set a priority for a destination. See Setting Certain Header Properties in Destinations .
`JMSMessageID`	String	An ID that uniquely identifies each message sent by a provider. A generated value overrides any value set in the corresponding event property.
`JMSTimestamp`	Long	The time when the message was handed off to a provider to be sent. The message may be sent later than this timestamp value. A generated value overrides any value set in the corresponding event property.
`JMSCorrelationID`	String	A correlation ID that can be used to link messages. For example, you can link a response message to a request message. Optional.
`JMSReplyTo`	String	The name of the JMS destination (queue or topic) to send the message reply to. If null, TIBCO BusinessEvents does not set the outgoing message's property. Optional.

Field name	Type	Description
		Note: Do not set the value to an empty string (""). If you do, TIBCO BusinessEvents sets the queue or topic name to an empty string that creates an exception, and the message is not sent. See How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages .
`JMSType`	String	The message type identifier, if used by the provider.
`JMSRedelivered`	Boolean	If this field is set, it is possible that the message was delivered to the client earlier but not acknowledged at that time.

JMS Sender Session Pooling

When working in the non-transacted mode, you can create a pool of JMS sender sessions for each JMS channel in the project.

Each thread, that runs any of the JMS sending functions (such as `sendEvent`, `replyEvent`, `routeTo`, and so on), takes a session from the pool. After utilizing the session and on completion of the send task, the thread returns the session to the pool. If the max pool size is reached, threads wait for a session to become available and until then, do not create any session.

By default, the pool is disabled and the sending functions share a single session per channel. To activate the pool set the value of the `be.channel.jms.sender.session.pool.maxsize` property, which identifies the maximum pool size, with the minimum allowed value as 1.

SOAP over JMS Support in WSDL Import and Export

SOAP and JMS are supported in WSDL Import and Export for events, channels, rules, rulefunctions, and catalog functions.

WSDL Import

The WSDL Import wizard generates resources for handling SOAP and JMS bindings.

Events

One Event type named `/Events/SoapJms`. This event inherits from SOAPEvent and provides a set of properties that may be useful for SOAP and JMS. Events are operation-specific events and they are imported similarly as for the SOAP and HTTP WSDL Import.

Serializers

Two types of serializers are available for conversion between JMS messages and `SOAPEvent` events. To send SOAP messages in byte format from the TIBCO BusinessEvents engine, use the `com.tibco.cep.driver.jms.serializer.SoapMessageSerializer` serializer. To send out messages in text format, you can use the `com.tibco.cep.driver.jms.serializer.SoapTextMessageSerializer` serializer.

Channels

A JMS channel by default is at `/Channels/SoapJms` for each different set of parameters, with one shared JMS connection resource.

By default the resource is in the Transport folder of the imported service folder. One destination for each different set of destination parameters is designated.

By default the destination is named `destinationName`, which uses the serializer SoapMessageSerializer and the default event type `/Event/SoapJms`. See the `RegisterSoapEventUris` function below, which allows the deserializer to know how to deserialize a message into operation-specific events instead of using the destination default.

Rules

There is one rule for each operation, which is imported similarly as for the SOAP and HTTP WSDL Import.

By default, the rule replies with a SOAP fault stating that the operation is not implemented.

RuleFunctions

There is one RuleFunction for each operation, which is imported similarly as for the SOAP and HTTP WSDL Import.

If desired, the RuleFunction can be used as a preprocessor function.

The one RuleFunction is named, `RegisterSoapEventUris`.

By default, it is available in the Transport folder of the imported service folder. This registers in the JMS SoapMessageSerializer, and event type for each operation that is reachable by a SOAP and JMS binding. It should be invoked as a startup function.

Catalog Functions

A serializer and deserializer for SOAP and JMS has been added to the JMS destinations. This serializer handles the conversion between the JMS messages and SOAPEvent events. Two help catalog functions are provided as well:

Catalog Function	Description
`SOAP.registerEventUri()`	Voids registerEventUri(String event URI, String destinationUri, String serviceName, String soapAction, String Preprocessor) Associates an event type to a given input destination, target service, and SOAP action. This is used by the SOAP and JMS deserializer to generate events of specific types (eventUri) instead of using the default event type resolution mechanism. • **eventUri**: The string path of an event type in the project. • **destinationUri**: The string path of the TIBCO BusinessEvents destination receiving the message. If this is empty, it matches all the destinations. • **serviceName**: The string name of the target service declared in the message received. If this is empty, it matches all the service names. • **soapAction**: The string value of the SOAPAction parameter in the message received. • **Preprocessor**: The string path of the preprocessor in the project.

Catalog Function	Description
`SOAP.newCorrelationId()`	String newCorrelationId(String responseEventUri) Creates a correlation ID for use when sending a message in a request and response case. This is used by the SOAP and JMS deserializer to generate a response event of a specific type, instead of using the default event type resolution mechanism. • **responseEventUri**: The string path of an event type in the project. • **returns**: The string correlation ID.

In the context of TIBCO BusinessEvents, the two tasks of a message serializer are deserialization from the transport-level message to an event and serialization from an event to the transport-level message.

WSDL Export

The WSDL Export was implemented to support SOAP and JMS, but it has some limitations. The following include its limitations:

1. It uses the TIBCO format and not W3C.
2. It relies on the presence of an event registration startup function with a body that looks like something that is generated during WSDL Import.
3. It relies on a folder structure that is similar to what is generated during import.
4. It does not export all the possible information, for example, the reply-to names.

Note: The import of a WSDL, then an export to another WSDL should result in similar WSDLs, though not identical.

Kafka Channel

To send and receive messages from a Kafka broker, use the TIBCO BusinessEvents Kafka channel. The Kafka channel converts the incoming Kafka messages to BusinessEvents events and transforms BusinessEvents events as outgoing Kafka messages.

Before You Begin

Before setting up the Kafka channel, go through the Kafka documentation for information (concepts, architecture, demos, APIs, and so on) about them. The following table lists the URLs that you can see for information about Kafka.

Channel Type	Documentation URL
Kafka	Kafka Documentation

Kafka Channel Serializers

The Kafka channel provides the following serializers to handle payloads:

KafkaMapSerializer

The KafkaMapSerializer serializer (`com.tibco.cep.driver.kafka.serializer.KafkaMapSerializer`) serializes and deserializes a BusinessEvents event along with its payload into (or from) HashMap. The KafkaMapSerializer is used to send and receive events between BusinessEvents instances. For incoming messages, the serializer converts the byte sequence to the event and its payload. For outgoing events, the serializer converts the event and its payloads into HashMap as bytes.

KafkaJsonSerializer

The KafkaJsonSerializer serializer (`com.tibco.cep.driver.kafka.serializer.KafkaJsonSerializer`) serializes and deserializes a BusinessEvents event along with its payload into (or from) JSON. For incoming messages, the KafkaXmlSerializer decodes the text from the message as a JSON string and deserializes it to an event. For outgoing events, the serializer converts the event and its payloads into a JSON string. The KafkaXmlSerializer serializer is useful for processing or sending messages between BusinessEvents and external systems.

KafkaXmlSerializer

The KafkaXmlSerializer serializer (`com.tibco.cep.driver.kafka.serializer.KafkaXmlSerializer`) serializes and deserializes a BusinessEvents event along with its payload into (or from) a JSON string that does not have any qualifiers. For incoming messages, the KafkaXmlSerializer decodes the text from the message as an XML string and deserializes it to an event. For outgoing events, the serializer converts the event and its payloads into an XML string. The KafkaXmlSerializer serializer is useful for processing or sending messages between BusinessEvents and external systems.

KafkaTextPayloadSerializer

The KafkaTextPayloadSerializer serializer (`com.tibco.cep.driver.kafka.serializer.KafkaTextPayloadSerializer`) serializes and deserializes a BusinessEvents event along with its payload into (or from) text ignoring event properties. This serializer ignores the event properties even if they are defined. For incoming messages, the KafkaTextPayloadSerializer reads the text from the message and deserializes it to an event. For outgoing events, the serializer converts the event payload into a Kafka message value.

Note: If you are migrating a project from an earlier version that contains `KafkaStringPayloadSerializer` then the project still works in this version. However, the `KafkaStringPayloadSerializer` serializer is not available for a new Kafka channel.

KafkaAvroSerializer

The KafkaAvroSerializer serializer (`com.tibco.cep.driver.kafka.serializer.KafkaAvroSerializer`) serializes and deserializes a BusinessEvents event along with its payload into (or from) Avro data format. For incoming messages, the KafkaAvroSerializer reads the data from the Avro message and deserializes it to an event. For outgoing events, the serializer converts the event payload into an Avro message value.

KafkaProtobufSerializer

The KafkaProtobufSerializer serializer (`com.tibco.cep.driver.kafka.serializer.KafkaProtobufSerializer`) serializes and deserializes a BusinessEvents event along with its payload into (or from) Protobuf data format. For incoming messages, the KafkaProtobufSerializer reads the data from the message in Protobuf format and deserializes it to an event. For outgoing events, the serializer converts the event payload into a Protobuf message value.

Note: The Confluent schema registry supports the Protobuf data format. However, the TIBCO Schema Registry does not support this. For details about the Kafka Schema Registry, see Apache Kafka Schema Registry.

Adding a New Channel

See the following topics for more information about the new Kafka channel:

- Adding a Kafka Channel in TIBCO BusinessEvents Application

Sample TIBCO BusinessEvents Applications

The following sample TIBCO BusinessEvents application is provided with TIBCO BusinessEvents for Kafka channel to help you understand the channel better:

Channel Type	Sample TIBCO BusinessEvents Application
Kafka	*BE_HOME*`/examples/standard/KafkaChannel`

Catalog Functions

You can send and receive events with payload in the Kafka channel using Event.sendEvent (), and Event.routeTo(). Currently, Kafka does not support synchronous request-reply of messages. Thus, the functions Event.requestEvent() and Event.replyEvent() are not supported.

Kafka Channel Advanced Properties

You can specify Kafka channel advanced properties in the CDD file by adding appropriate prefixes to the channel. By using the same prefix, you can also override an existing Kafka channel property, which has been exposed through the TIBCO BusinessEvents Studio channel editor. For example, for specifying the `default.replication.factor` Kafka property, use `be.channel.kafka.default.replication.factor`.

Channel Type	Prefix	Description
Kafka	`be.channel.kafka`	Used for Kafka properties that are applicable for the Kafka channel and all its destinations.
	`be.channel.kafka.`*`<destination_name>`*	Used for Kafka properties that are applicable for only the specified destination of the Kafka channel.
	`be.channel.kafka.`*`<Channel URI>`*`.`*`<Kafka Property>`*`=`*`<value>`*	• Used to configure a multiple Kafka channels property. For example, • To configure multiple Kafka Client JAAS configurations in a single be-engine, use the below syntax: `be.channel.kafka/Channels/KafkaConsumer.sasl.jaas.config=<value1>` `be.channel.kafka/Channels/KafkaProducer.sasl.jaas.config=<value2>` • To set a property `default.replication.factor`, only for a particular destination, you can include a Destination URI instead of a Channel URI. `be.channel.kafka/Channels/KafkaChannel_1/dest.default.replication.factor=value1`

Support for Kafka Message Parameters

The Kafka message parameters, such as key and offset, can be accessed through the TIBCO BusinessEvents event properties. You can define event properties for each Kafka parameter and use them in a rule or rulefunction. The deserializer fills the values in these event property fields when creating a TIBCO BusinessEvents event instance for a received Kafka message. For details, see Event Property and Rules and Functions.

The following properties can be added to events for Kafka message parameters.

Event Property	Description
`kafka_message_key`	Placeholder for a key of the Kafka message.
`kafka_message_offset`	Placeholder for an offset of the Kafka message.
`kafka_message_partition`	Placeholder for a partition number of the Kafka message.
`kafka_message_timestamp`	Placeholder for a time stamp of the Kafka message.
`kafka_message_timestamptype`	Placeholder for a type of the time stamp field of the Kafka message.

Adding a Kafka Channel in TIBCO BusinessEvents Application

Configure channel properties and destination properties for the Kafka channel to add it successfully to the BusinessEvents application.

Procedure

1. In TIBCO BusinessEvents Studio, add a channel with **Driver Type** as Kafka.

   See Adding a Channel.

2. In the channel editor, **Method of Configuration** is preselected as Properties. Enter the channel configuration properties values under the **Properties** section.

   See Kafka Channel Configuration Properties.

3. Now, add a destination for the Kafka channel and configure its properties.

   See Adding a Destination to a Channel and Kafka Destination Configuration Properties.

4. Save the new Kafka channel.

Setting Up Authentication for Kafka

You can set up authentication and SSL to improve security between BusinessEvents and a Kafka broker.
You must configure Kafka broker and Kafka client (BusinessEvents) for authentication.

Procedure

Kafka Broker Configuration

1. Configure the Kafka broker for a security protocol that you require for authentication, for example `SASL_PLAIN`.

   For more details about the steps involved for configuring a Kafka broker for a security protocol (JAAS configuration, JVM parameters, and server properties), refer to the Kafka documentation at https://kafka.apache.org/documentation/#security.

2. Start ZooKeeper and Kafka broker.

Kafka Client (BusinessEvents) Configuration

3. In BusinessEvents studio, configure the Kafka channel fields for security (**Security Protocol** and **Security Mechanism**).

   See Kafka Channel Configuration Properties. for more details.

   For example, `SASL_PLAINTEXT` is selected as the value of **Security Protocol** and `PLAIN` is selected as the value of **Security Mechanism** fields respectively.

4. Open the BusinessEvents default JAAS configuration file (*BE_HOME*`\mm\config\jaas-config.config`) for editing.

   Note: If you want to use your own JAAS configuration file (for example, `kafka_client_jaas.config`) specify its location in the `java.security.auth.login.config` CDD property. For example:

   ```
   <property name="java.security.auth.login.config"
   value="D:/kafka_client_jaas.config"/>
   ```

5. Configure the `KafkaClient` section in the JAAS configuration file. Specify the login module based on the **Security Mechanism** selected in the Kafka channel properties (see Kafka Channel Configuration Properties).

   You can configure `KafkaClient` using the following login modules for authentication:

   - `org.apache.kafka.common.security.plain.PlainLoginModule` - Specify `PlainLoginModule` for `PLAIN` SASL security mechanism when you want to send plain text a (non-encrypted) password for authentication. Specify your username and non-encrypted password in the section.

   - `com.tibco.cep.driver.kafka.security.BEPlainLoginModule` - Specify `BEPlainLoginModule` for `PLAIN` SASL security mechanism when you want to send your encrypted password for authentication. Specify your username and encrypted password in the section. You can use `studio-tools` utility to encrypt the password, see Generating Encrypted Passwords.
     `com.sun.security.auth.module.Krb5LoginModule` - Specify `Krb5LoginModule` for `GSSAPI` (Kerberos) SASL security mechanism.

   - `org.apache.kafka.common.security.scram.ScramLoginModule` - Specify `ScramLoginModule` for `SCRAM-SHA-256` and `SCRAM-SHA-512` SASL security mechanisms.

   For more information on setting configuring the JAAS file for Kafka clients, refer to the Kafka documentation at https://kafka.apache.org/documentation/#security.

   For example, see the following sample configuration of the `KafkaClient` section for the `PLAIN` SASL security mechanism and to send an encrypted password:

```
KafkaClient {
  com.tibco.cep.driver.kafka.security.BEPlainLoginModule required
//BEPlainLoginModule is a wrapper over Kafka's PlainLoginModule
with added support of TIBCO encrypted passwords.
  username=admin
  password="#!8McplDveXbBUsDBnPWzGvAfwlNhVIYS/";
};
```

6. Save the JAAS configuration file and start the BusinessEvents engine (producer and consumer).

Apache Kafka Schema Registry

You can use the Schema Registry to validate Kafka messages against a registered schema. Schema Registry ensures data consistency and increases data quality. Schema Registry supports schema versioning so that the different versions of the schema can be used simultaneously without causing compatibility issues. TIBCO BusinessEvents supports Confluent Schema Registry and TIBCO Schema Registry.

Before you begin

- For information about the installation and concepts of the Schema Registry, see the following documentation:
  - Confluent Schema Registry
  - TIBCO Schema Registry

Configuring the Schema Registry

See Kafka Destination Configuration Properties for information about configuring the Schema Registry.

Subject Naming Strategies

Confluent Schema Registry

When the schema is registered, the name of the subject is decided by the subject name strategy. Subject name strategy can be set using Kafka destination properties. By default the subject name strategy is TopicNameStrategy. The subject name strategy can be configured by using the following property:

```
be.channel.kafka/Channels/<channelname>/<destinationname>.value.subject.
name.strategy=<Subject Name Strategy Class>
```

You can set one of the following subject-named strategy classes:

- io.confluent.kafka.serializers.subject.TopicNameStrategy
- io.confluent.kafka.serializers.subject.RecordNameStrategy
- io.confluent.kafka.serializers.subject.TopicRecordNameStrategy

TIBCO Schema Registry

For the TIBCO Schema Registry, you can set the subject name by using the property `value.subject.name` with a few "%" template variables. The default value for this property is `%t-value`.

The application automatically replaces the special characters %t, %r, and %% with topic, name of schema, and literal % character respectively.

You can set the value for the `value.subject.name` as TIBCO BusinessEvents property for each destination.

Example:

```
be.channel.kafka/Channels/KafkaChannel/orderDest.value.subject.name=%r-%
t-value
```

The preceding property configures `value.subject.name` for "orderDest" destination. The subject name is "RecordName-TopicName-Value".

See Apache Avro Client Library Configuration Options for more information.

Schema Compatibility

When schemas evolve and change, you must add (register) different schema versions. Before adding a version of a schema, the Schema Registry checks whether the new schema is compatible with the previously registered schema under the subject. This compatibility is managed through the Schema Registry compatibility parameter. See Confluent documentation for more information about setting schema compatibility.

Schema Registration Scenarios

Refer to the following table for different scenarios about how schema is registered in the Schema Registry based on whether Subject and Schema fields are specified or not.

Note: You must set the appropriate schema compatibility type before registering a new schema.

Subject field specified?	Schema field specified?	How Schema is registered?
Yes	Yes	If the schema subject is valid, TIBCO BusinessEvents fetches the latest schema from the Schema Registry associated with the subject. If the subject name is not as per the subject name strategy, then TIBCO BusinessEvents registers the latest schema under subject that is as per the subject name strategy. If the schema subject is not valid, TIBCO BusinessEvents registers the schema configured in the Kafka destination properties with the subject name as per the subject name strategy. If the schema is already registered, then TIBCO BusinessEvents updates the schema in the Schema Registry.
No	Yes	Registers the schema configured in the Kafka destination properties. If the schema is already registered, then TIBCO BusinessEvents updates the schema in the Schema Registry.
Yes	No	If the schema subject is valid, TIBCO BusinessEvents fetches the latest schema from the Schema Registry associated with the subject. If an associated subject name is not as per the subject name strategy, then TIBCO BusinessEvents registers the latest schema under subject that is as per the subject name strategy. If schema subject is not valid, TIBCO BusinessEvents creates schema using event properties and payload associated with the destination and registers it as per the subject name strategy
No	No	If schema subject is not valid, TIBCO BusinessEvents creates schema by using event properties and payload associated with the destination and registers it as per the subject name strategy.

Note:

- Confluent Schema Registry always sets the subject name as per the subject name strategy. TIBCO Schema Registry sets subjects as per % template variables provided in `value.subject.name`.
- If the **Update Schema in Schema Registry** checkbox is selected, then Schema and Subject field scenarios are not considered. In this case, TIBCO BusinessEvents creates the schema by using event properties and the payload associated with the destination and registers it as per the subject name strategy.

Importing Schema Registry Specification in TIBCO BusinessEvents Studio

This feature imports a schema from the Kafka Schema Registry for a subject to an Event payload in TIBCO BusinessEvents.

Procedure

1. Select **File** > **Import** > **TIBCO BusinessEvents** > **SchemaRegistry Specification**, then click **Next**.
2. Specify the fields required. See Kafka Destination Configuration Properties for more information about the fields.
3. Click **Finish**.

Advanced Configuration Properties for Kafka Channel

You can configure the following configuration properties in the CDD file.

Property	Description
`be.channel.kafka.skip.attributes`	Set this property to true to remove `attributes` field from the event JSON payload. **Note:** The property `be.channel.kafka.skip.attributes=true` works for only KafkaJsonSerializer.

Apache Pulsar Channel

To send and receive messages from a Pulsar broker, use the TIBCO BusinessEvents Pulsar channel. The Pulsar channel converts the incoming Pulsar messages to BusinessEvents events and transforms BusinessEvents events as outgoing Pulsar messages.

Before setting up the Pulsar channel, go through Pulsar documentation for information about concepts, architecture, demos, APIs, and so on.

Pulsar Channel Serializer

The Pulsar channel provides the following serializers to handle payloads:

PulsarJsonSerializer

The PulsarJsonSerializer serializes and deserializes a SimpleEvent along with its payload into (or from) JSON. For incoming messages, the PulsarJsonSerializer decodes the text from the message as a JSON string and deserializes it into an event. The serializer converts the events and payload into a JSON string for outgoing events.

PulsarAvroSerializer

The PulsarAvroSerializer, `(com.tibco.cep.driver.pulsar.serializer.PulsarAvroSerializer)` serializes and deserializes an event along with its payload into (or from) Avro data format. The PulsarAvroSerializer, process the schema into AVRO format while sending the data and deserilize the AVRO format while receiving the data.

See Adding a Pulsar Channel for more information about adding a Pulsar channel.

Advanced Configuration Properties for Apache Pulsar Channel

See the following topic for more information about Advanced Configuration Properties for an Apache Pulsar Channel:

- Advance Configuration Properties for the Pulsar Channel

Sample TIBCO BusinessEvents Applications

To help you understand this channel better, the following sample TIBCO BusinessEvents application is provided with TIBCO BusinessEvents:

`BE_HOME/examples/standard/PulsarChannel`.

Adding a Pulsar Channel

Configure channel properties and destination properties for adding the Pulsar channel to TIBCO BusinessEvents application.

Procedure

1. In TIBCO BusinessEvents Studio, add a channel with **Driver Type** as Pulsar.

   See Adding a Channel.

2. In the channel editor, **Method of Configuration** is preselected as **Properties**. Enter the properties values for channel configuration under the **Properties** section.

   See Pulsar Channel Configuration Properties.

3. Add a destination for the Pulsar channel and configure its properties.

   See Adding a Destination to a Channel and Pulsar Destination Configuration Properties.

4. Save the new Pulsar channel.

Setting Up Authentication for Pulsar

You can set up authentication and SSL to improve security between TIBCO BusinessEvents and Pulsar broker. You must configure a Pulsar broker and Pulsar client (TIBCO BusinessEvents) for authentication.

Procedure

1. Configure the Pulsar broker for a security protocol that you require for authentication.

   For more details about the steps for configuring a Pulsar broker for a security protocol, refer to the Pulsar documentation.

2. In BusinessEvents studio, configure the Pulsar channel fields for security:

Security Protocol	Procedure
mTLS	1. Start Pulsar broker with mTLS Authentication mode. See Pulsar documentation to create the server and client certificate and to start the Pulsar broker. 2. Select mTLS from the **Security Protocol** dropdown. 3. Click **Configure**. 4. Specify the configuration fields. See Pulsar Channel Configuration Properties for more information. **Note:** When client certificates are not encrypted, provide any dummy password in the identity file.

JWT	1. Generate the JWT authentication token by using the following commands:
	`> bin/pulsar tokens create-secret-key --output my-secret.key`
	`> bin/pulsar tokens create --secret-key file:///path/to/my-secret.key --subject test-user`
	2. Start Pulsar broker with JWT Authentication type. See Pulsar documentation for more information.
	3. In BusinessEvents Studio, select JWT from the **Security Protocol** dropdown.
	4. Click **Configure**.
	5. Specify the generated token in the **Authentication Token** field.

Security Protocol	Procedure
OAuth 2	1. Start the Pulsar broker using OAuth 2 Authentication mode. See Pulsar documentation for more information. 2. In BusinessEvents Studio, from the **Security Protocol** dropdown, select OAuth 2. 3. Click **Configure**. 4. Specify the JSON configuration file path that includes `type`, `client_id`, `client_secret`, `client_email`, `issuer_url` parameter details. 5. Specify Issuer and Audience URL.
HTTP Basic	1. Start the Pulsar broker with HTTP_BASIC Authentication. See Pulsar documentation. 2. In BusinessEvents Studio, select HTTP_BASIC from the **Security Protocol** dropdown. 3. Click **Configure**. 4. Create an Identity Resource for username and password fields. Specify the Identity Resource path when the HTTP_BASIC configuration dialog opens.

Apache Pulsar Schema Registry

Pulsar messages are stored as unstructured byte arrays. Both the producer and consumer need to have the same data structure for the messages. Apache Pulsar schema is the metadata that defines how to translate the raw message bytes into a formal structure type. It serializes data into raw bytes before they are published to a topic and deserializes the raw bytes before they are delivered to consumers.

Pulsar uses a built-in `Pulsar Schema Registry` as a central repository to store the registered schema information. The producers and consumers use the schema registry to coordinate the schema of a topic's messages through brokers. The Pulsar schema is applied at the `Topic` level. As producers and consumers can upload schema to brokers, Pulsar schema work on both the producer and consumer side.

Schema registry ensures data consistency and increases data quality. It supports schema versioning so that the different versions of the schema can be used simultaneously without causing compatibility issues.

Pulsar Schema is defined in a data structure called SchemaInfo. The following table describes the field of a SchemaInfo defined for Pulsar Schema.

Field	Description
Name	Name of the Schema in string format.
Type	Specify the Schema Type for serializing and deserializing the schema data. See Pulsar documentation for more information about Schema Types.
Schema	Specify the schema to be registered in the Schema Registry. Only the Avro schema type is supported.
Properties	Specify a user-defined property as a string or string map, which can be used by applications to carry any application-specific logic.

Configuring the Schema Registry

See Pulsar Destination Configuration Properties for information about configuring the Schema Registry.

Schema Compatibility

When schemas evolve and change, you must add (register) different schema versions. Before adding a version of a schema, the Schema Registry checks whether the new schema is compatible with the previously registered schema for a given topic. See Pulsar documentation for more information about setting schema compatibility.

Schema Registration Scenarios

The schema compatibility check ensures that the existing consumers can process the introduced messages. The following table describes the different scenarios of schema registration in the Schema Registry based on the schema exists or not.

Note: You must set the appropriate schema compatibility type before registering a new schema.

Schema Compatibility Scenarios	Result
If no schema exists for the topic	• The producer is created with the schema. • The schema is transmitted to the broker and stored as there is no existing schema. • The consumer created using the same topic can consume messages using the same schema.
If a schema exists and the producer or consumer connects using the same schema that is already stored	• The schema is transmitted to the broker. • The broker determines the compatibility of the schema. • The broker attempts to store the schema in BookKeeper. But if it is already stored, then the same schema is used to produce and consume the messages.
If a schema exists and the producer or consumer connects using a new schema that is compatible	• The schema is transmitted to the broker. • The broker determines the compatibility of the schema and stores the new schema as the current version (with a new version number).

Importing Pulsar Schema Registry Specification in TIBCO BusinessEvents Studio

This feature imports a schema from the Apache Pulsar Schema Registry for a given topic to an event payload in TIBCO BusinessEvents.

Procedure

1. Select **File > Import > TIBCO BusinessEvents > SchemaRegistry Specification**, then click **Next**.
2. Specify the fields required. See Pulsar Destination Configuration Properties for more information about the fields.
3. Click **Finish**.

Advance Configuration Properties for the Pulsar Channel

You can configure the following advanced properties for the Pulsar channel in the CDD or `be-engine.tra` file.

Property	Description
`be.channel.pulsar.producer.name`	Use this property to set a custom Pulsar producer name.
`be.channel.pulsar.consumer.name`	Use this property to set a custom Pulsar consumer name.
`be.channel.pulsar.destination.subscription.name`	Specifies the Pulsar consumer subscription name.

Events

TIBCO BusinessEvents supports three sorts of events: Simple events (usually referred to just as events); time events, which are timers; and advisory events. This chapter explains how to use simple events.

A simple event defines an object that represents an occurrence of something. When the term "event" is used without the qualifier advisory or time, it refers to a simple event.

- *TIBCO BusinessEvents Getting Started* provides a practical introduction to events, including use of default events and default destinations.
- See *TIBCO BusinessEvents Architect Guide* for more detailed information on Events.

Overview of Simple Events

A simple event defines an object that represents an occurrence of something.

When the term "event' is used without the qualifier advisory or time, event refers to a simple event.

For more information, see the following:

- *TIBCO BusinessEvents Getting Started* guide provides a practical introduction to events, including the use of default events and default destinations.
- *TIBCO BusinessEvents Architect Guide* provides a detailed overview and information about Message Acknowledgment, Default Destinations and Default Events, Simple Events -- Time to Live and Expiry Actions.

Using Inheritance

Use of inheritance can simplify event configuration.

A child event inherits:

- All the parent event’s properties.

- The parent event’s expiry action (if set). However, an expiry action set in the child event overrides the parent event expiry action.

> **Note:** A parent event cannot have a payload. Also, all child events of an event with TTL=0 must also use TTL setting TTL=0 .

Events that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore the following restrictions apply:

- If two events are related by inheritance, you cannot create a new property in one with a name that is already used in the other.
- If two unrelated events have properties that share a name, you cannot create an inheritance relationship between the two events.

Working with Events in Rules

At runtime, event instances that are created using rules are not automatically asserted.

You must explicitly assert such events, for example using the `Event.assertEvent()` function.

Events that are created from incoming messages, on the contrary, are automatically asserted.

STIBCO BusinessEvents includes two functions that allow you to send simple events out to another application: `Event.sendEvent()` and `Event.routeTo()`.

- `Event.sendEvent()` automatically sends the event to its default destination.

> **Note:** The Event.sendEvent() function does not recognize whether the event is Put or Take. Use Legacy ActiveSpaces catalog function for Put or Take operations.

- `Event.routeTo()` takes a destination as an argument, ignoring the event's default destination.

  With `routeTo`, you can direct an event to a destination on a different channel from the event's default destination. You can also override the properties of the destination, for example, the subject.

  You cannot, however, override the properties of the channel itself.

You can use two methods to schedule simple events:

- Rule-based time events schedule the assertion of simple events.
- Scheduler functions schedule the sending of simple events to their default channels.

Adding a Simple Event

This section provides summary steps for adding a simple event. See Simple Event Reference for details about how to complete the values.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the folder where you want to store the event and select **New > Simple Event**. You see the New Simple Event Wizard.
2. In the **Simple Event Name** field, type a name for the event. In the **Description** field, type a description as desired.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in TIBCO BusinessEvents Studio Explorer and select **Refactor > Rename**.

3. Click **Finish**. You see the Simple Event Editor.
4. Complete the **Standard** tab and **Advanced** tab sections as explained in Simple Event Reference.
5. Save the resource.

Defining Payloads Using XML Schema Files

You can define the XML schema file within TIBCO BusinessEvents Studio, or import an existing one. Then you can reference the schema when defining the payload.

To access an XML payload in the rules, you can use the @payload attribute, as well as the Event, Mapper, and XPath functions. For simple XML payloads, you can use string functions.

Defining an XML Schema File in TIBCO BusinessEvents Studio

An Eclipse resource is available for this task.

Procedure

1. Right-click the folder where you want to add the resource and select **New > Other > XML > XML Schema** and click **Next**.
2. Enter a name and click **Finish**.
3. In the XML schema editor use the **Design** or **Source** tabs as preferred to define the schema. Refer to *Eclipse help* for details.

Importing an XML Schema File

An Eclipse resource is available for this task.

Procedure

1. Go to **File > Import > File System**.
2. At the Import from File System wizard, browse to a folder and select the desired schema file within that folder.
3. In the **Into Folder** field, specify the project folder where you want to put the file.
4. Set the overwrite and folder options as desired, then click **Finish**.

Using an XML Element in a Schema File in an Event Payload

To use an XML element in a schema file as the root element of the event payload, you will have to perform the following actions.

Procedure

1. In the event's **Advanced** tab, click the **Add** (+) button. You see a root node in the left panel.
2. In the **Content** field, select **XML Element Reference**.
3. In the **Schema** field, click the **Browse** (binocular) button.
4. You see the Select a Resource dialog. Depending on the XML schema files available in the project, multiple entries may be grouped according to location (on the By Location tab) and namespace (on the **By Namespace** tab). Select one XML schema resource from either of these tabs.
5. In the Element panel, select the element you want to use as the root element for the payload. (You can also change this in the Payload section of the Event's **Advanced** tab.)
6. To return to the **Advanced** tab, click **OK**. The word `root` in the left panel is now replaced with this element name.
7. Save the resource.

Simple Event Reference

Simple Event resources are used to define an object that represents an occurrence, such as sending an invoice, debiting an account, and so on.

You can modify and enrich events before they are asserted into the Rete network. Rule evaluation depends on event values at time of assertion, so they can be changed only before assertion.

Note: If you are working with a project imported from a release earlier than 5.0.0, you might be able to see metadata properties. However, do not use them. Instead, use the settings and properties in the Domain Objects section of the CDD file as needed.

Wizard and Configuration (Standard Tab)

The Wizard and the Configuration section have the following fields.

Field	Global Var?	Description
Name	No	Not shown as a field because it cannot be changed. The name appears in the Wizard, and in the title of the event. The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements.
Description	No	Short description of the resource.
Inherits From	No	This event inherits from the event you select here. Leave blank if you do not want to use inheritance. For convenience, you can open the selected event resource by clicking the underlined label.
Time to Live	Yes	Specify a numerical value and a time unit. Time units can be specified from the drop-down list available to the right of the **Time to Live** field. Choose one of: milliseconds, seconds, minutes, hours, and days. The default time unit is seconds. The numerical value is interpreted as follows: • One or higher (>0): the event expires after the specified number of units elapse.

Field	Global Var?	Description
		• Zero (0): the event expires after the completion of the first RTC cycle. • A negative integer (<0): the event does not expire, and must be explicitly consumed. The value `-1` is generally used to indicate an event that does not expire. See Declaration and Expiry Action (Advanced Tab) .
Default Destination	No	When the destination is not otherwise specified (for example in rules or rule functions), events of this type are sent to the destination you select here. You can send an event to the default destination of its event type using the `Event.sendEvent()` function. For convenience, you can open the selected destination resource by clicking the underlined label.
Retry On Exception	No	When a destination's event preprocessor fails due to an exception, the behavior of an event instance of this type is determined by this checkbox setting: • When this checkbox is selected, TIBCO BusinessEvents attempts to reprocess the event instance that failed and retries indefinitely with a delay of five seconds between retries. • When the checkbox is not selected TIBCO BusinessEvents does not attempt to reprocess the event instance that failed.

Properties (Standard Tab)

The Properties section has the following fields. Event properties generally map to incoming or outgoing message properties.

Field	Global Var?	Description
Name	No	The name to appear as the label for the property. Names follow Java variable naming restrictions. Do not use any reserved words. See Identifier Naming Requirements. **Note:** In addition to standard naming restrictions, do not begin an event property name with `_ns_` or `_nm_`. These have a special use. Also note that the property name cannot begin with the character `_`. **For events used in JMS channels** Names beginning with `_jms` or `jms` (case insensitive) are used only for JMS header properties. You can, however, use properties beginning `jms_` (case insensitive) for event properties. JMS Header Field Names shows the list of JMS header properties. Consult the JMS specification for more details.
Type	No	One of: `String`, `Integer`, `Long`, `Double`, `Boolean`, `DateTime` **Note:** For properties of type Double, all NaN (Not a Number) values are converted to `0.00`.
Domain	No	Click the search button and select the domain model you want to use for this property. See Domain Models for details about adding domain models.

Declaration and Expiry Action (Advanced Tab)

If the Time to Live field is zero or higher, define the action or actions to take when an event expires in the Expiry Action section.

If an event is explicitly consumed in a rule, TIBCO BusinessEvents does not execute the expiry action.

The editor in the Expiry Action section is the same as the Rule function editor. See Rule Function Resource Reference for details.

See *TIBCO BusinessEvents Architect Guide* for more details.

Payload (Advanced Tab)

An event can have a payload. The payload often corresponds to a message body. Payloads can be defined using an XML schema. In the left panel, you add groups (elements) and parameters (attributes). You can add groups as children of a selected group, or at the same level, to define a hierarchy as desired. In the right panel, you define the type of each element or parameter. The table below describes the payload parameters available for each content type. The content types appear in the dropdown list for the **Content** field in the Payload section:

Simple Event Payload Element Parameters (Sheet of)

Content/Parameter	Description
Complex Element	An element that contains other elements. This is like a structure in a programming language. The complex element can contain zero or more elements of other types, including other complex elements.
Name	The name of the element.
Cardinality	Values for Cardinality: • Required: The payload must include an instance of this element. • Optional (?); The element is not required. • Repeating (*); The element is a list that has zero or more instances. • At least one (+): The element is a list that has one or more instances.
Element of Type	An element with a specified data type. You can specify a scalar data type (string, integer, and so on), you can reference an XML type, or you can specify the `TIBCO ActiveEnterprise Any` data

Content/Parameter	Description
	type.
Name	The name of the element.
Cardinality	See Cardinality under Complex Element.
Type	The generic data type. For example, decimal or date/time.
Type	The specific data type. For example, float or month. Refer to the *TIBCO ActiveMatrix BusinessWorks Palette Reference* for a complete list.
XML Element Reference	A reference to an element in a stored XML schema. See TIBCO Designer documentation for more information about XML schemas.
Cardinality	See Cardinality under Complex Element.
Schema	Stored XML schema that contains the element or type you want to reference.
Element	The element within the stored XML schema that you want to reference.
Attribute of Type	An attribute with a specified data type. You can specify a scalar data type (string, integer, and so on), you can reference an XML type, or specify the `TIBCO ActiveEnterprise Any` data type.
Name	The name of the element.
Cardinality	See Cardinality under Complex Element.
Type	The generic data type. For example, decimal or date/time.
Type	The specific data type. For example, float or month. Refer to the *TIBCO ActiveMatrix BusinessWorks Palette Reference* for a complete list.

Content/Parameter		Description
Sequence		A sequence of elements. Each item in the sequence is a structure of the sub-elements of this element.
	Cardinality	See Cardinality under Complex Element.
Choice		A choice of elements. The data type of this element can be one of the sub-elements defined.
	Cardinality	See Cardinality under Complex Element.
All		The data type of this element can be all the data types of the sub-elements defined.
	Cardinality	See Cardinality under Complex Element.
XML Group Reference		A reference to an XML group in a stored XML schema. See TIBCO Designer documentation for more information about the XML schema.
	Cardinality	See Cardinality under Complex Element.
	Schema	Stored XML schema that contains the element or type you want to reference.
	Model Group	Select the appropriate model group from the pull-down list.
Any Element		A reference to any XML Element. You can use the Coercions button to supply a reference to the XML Element for this item when it appears in the input or process data.
	Cardinality	See Cardinality under Complex Element.
	Validation	Select the level of validation to be performed on the XML Element: • Strict: Must validate by locating a declaration for the element.

Content/Parameter	Description
	• Skip: Do not validate. • Lax: Validate if a declaration for the element is available.

Note: The CDD property `be.serialization.inline.namespaces` defines the location of namespace references in the payload.
`be.serialization.inline.namespaces = false` (default) declares the namespace at the top of the document.
`be.serialization.inline.namespaces = true` (same as in BE version 5.6.x) declares the namespace at the level where it is first referenced.

Simple Event Attributes Reference

Note: When using an event in a rule's form editor, type *`eventname@`* to see the list of its attributes.

You can use the following attributes in rules to return information about a simple event instance.

Attribute	Type	Returns
`@id`	object	The event's unique internal ID.
`@extId`	string	The event's unique external ID. Optional. The value of the `extId` is set at creation time, for example, using the event's ontology function, or the `Event.creatEvent()` function. The value cannot be changed after that. **Note:** The `extId` value (if set) must be unique across all objects in the cluster.

Attribute	Type	Returns
		Tip: You can use the property `Agent.`*`AgentClassName`*`.checkDuplicates` to check for duplicate `extIds` across the cluster. You can use this property only when the legacy ID lookup strategy is enabled.
`@ttl`	long	The event's time to live, where the assertion of the event defines the start of the time to live period. You can specify the value in the SimpleEvent resource TTL field. See Simple Event Reference .
`@payload`	string	The payload as a string value. See Simple Event Reference for more on specifying the payload in a Simple Event resource.

Working With Time Events

This section explains how to create and configure a time event.

Scheduled Time Events

TIBCO BusinessEvents offers two kinds of time events, repeating and rule-based. In addition you can schedule events using functions.

Note: Time events configured to repeat at intervals are not supported in multiple-agent (multi-engine) configurations. Rule-based time events, however, are supported.

You can configure a time event to repeat at a configurable time interval. For example, if you configure a time event to repeat every thirty seconds, then every thirty seconds TIBCO BusinessEvents creates a new time event of that type.

You can configure a repeating time event to create a specified number of events at each interval. The time interval begins during engine startup. See Engine Startup and Shutdown Sequence in *TIBCO BusinessEvents Administration* for specific details.

Rule-Based Time Events

A rule-based TimeEvent resource has only a name and description.

You can then use it in a rule to schedule a simple event to be asserted, using its ontology function, `Schedule`*`TimeEventName`*`()` in a rule. It allows you to schedule the event to be asserted after a period, and pass the information to the event and specify its time to live. Also, you can call the `Schedule`*`TimeEventName`*`()` function in different places with different time delays.

You can use rule-based time events in various ways. For example, you might write rules that check for delays to fulfillment:

1. A new Order event is asserted, and Rule A (which has Order in its scope) creates a time event T and configures it to be asserted in 60 minutes, and passes the order ID as the closure parameter value. (Rule A also sends the order details to another system.)
2. Sixty minutes after Rule A runs, timer event T is asserted.
3. The assertion of time event T triggers Rule B, which has T in its scope. Rule B checks the order status. If the order is delayed, it sends out an alert.

Adding a Time Event

See TimeEvent Resource Reference for information on how to complete the values.

Procedure

1. In BusinessEvents Studio Explorer, right-click the folder where you want to store the time event and select **New > Time Event**. You see the New Time Event Wizard.
2. In the Time Event Name field, type a name for the time event. In the Description field, type a description.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename**.

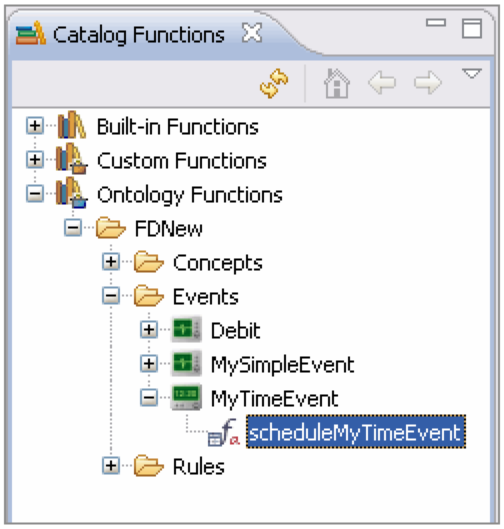
3. Click **Finish**. You see the Time Event editor.

4. In the **Type** field, select the type of time event you want to create and configure accordingly:
   - ruleBased: Select Rule Based. Use the event in a rule as explained in Configuring a Rule-Based Time Event in a Rule or Rule Function.
   - repeat: The event occurs at specified intervals. Select Repeat and configure the time event following guidelines in TimeEvent Resource Reference.
5. Click **Apply** and save the resource.

Configuring a Rule-Based Time Event in a Rule or Rule Function

Procedure

1. Open the rule editor for the rule or rule function where you want to use the rule-based time event.
2. Display the Catalog Functions view if it is not available: Select **Window > Show View > Other > TIBCO BusinessEvents > Catalog Functions**.
3. In the Catalog Functions view, expand Ontology Functions and drill down to the rule-based time event. Expand the rule-based time event and select its ontology function (schedule*TimeEvent*).



4. Drag the time event's ontology function into the Actions area of the rule (or Body area of the rule function) and configure the parameters, following guidelines in Rule-Based TimeEvent Function Reference.

5. Save the resource.

TimeEvent Resource Reference

TimeEvent resources are used as timers. You can configure rule-based timers, which are scheduled in a rule, and repeating timers, which are scheduled in the TimeEvent resource. Use time events to trigger rules.

> **Note:** If you are working with a project imported from a release earlier than 5.0.0, you may see metadata properties. However, do not use them. Instead use the settings and properties in the Domain Objects section of the CDD file as needed.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. **For events used in JMS channels** Names beginning with `_jms` or `jms` (case insensitive) are used only for JMS header properties. You can, however, use properties beginning `jms_` (case insensitive) for event

Field	Global Var?	Description
		properties.
Description	No	Short description of the resource.
Type	No	• ruleBased: The time event is scheduled by a rule. • repeat: Time events are created periodically. Complete the rest of the settings to define the period and number of events per interval.
Interval	No	Used with repeating time events only. Specify the interval between time events using a numerical value and a time unit. TIBCO BusinessEvents creates the first time event immediately after the engine starts and then creates the next time event based on the interval. A repeating time event expires after the completion of the first RTC cycle (that is, the time to live code is set internally to zero). Time units available are milliseconds, seconds, minutes, hours, and days.
Events per Interval	No	Used with repeating time events only. Enter the number of events to create in each Repeat Every interval. Default is 1.

TimeEvent Attributes Reference

Note: When using an event in a rule's form editor, type `eventname@` to see the list of its attributes.

You can use the following attributes in rules to return information about an event instance.

Attribute	Type	Returns
`@id`	object	The event's unique internal ID.
`@closure`	String	• For repeating time events: Not used (Null value). • For rule-based time events: A string that was specified when the event was scheduled.
`@interval`	long	• For repeating time events: The number of milliseconds between creation of successive time events. • For rule-based time events: Not used (0 value).
`@scheduledTime`	DateTime	The time scheduled for asserting an instance of this event into the Rete network. • For repeating time events: calculated based on the time of the last assertion of an instance of this event, and the interval. • For rule-based time events: specified using the `Schedule`*`TimeEventName`*`()` function's `delay` parameter. See Rule-Based TimeEvent Function Reference .
`@ttl`	long	• For repeating time events: always 0 (zero). • For rule-based time events: specified using the ontology function's `ttl` parameter. See Rule-Based TimeEvent Function Reference .

Rule-Based TimeEvent Function Reference

Signature

```
Event ScheduleTimeEventName(long delay, String closure, long ttl)
```

Domain

action

Description

For each rule-based time event you create, an ontology function is also created to enable you to schedule the assertion of an instance of the event in a rule action. The function name follows this format: Schedule*TimeEventName*.

Parameters

Name	Type	Description
`delay`	`long`	The event is created (and asserted) the specified number of milliseconds after the rule action runs.
`closure`	`String`	The TimeEvent created stores the information passed in the closure parameter. You can put the closure string value in the rule conditions to specify the time events that trigger the rule.
`ttl`	`long`	The event's time to live. It follows the same rules as the time to live setting in a simple event. See Simple Event Reference.

Returns

Type	Description
`Event`	An instance of the event type specified in the function name (Schedule*TimeEventName)*.

Using Scheduler Functions (Requires Cache OM)

You schedule simple events using scheduler functions. Events scheduled using these functions are sent to their default destination.

You might use the event scheduler functions instead of a time event in the following cases:

- If you need to send any event on a schedule, you must use the scheduler. If you schedule a memory-only simple event the schedule is persisted. If you schedule a memory only time event (using the time event's function, not the scheduler functions) then the schedule is not persisted.
- If you need to create many of schedules at one time, performance is better using the event scheduler functions.
- Because schedule management and event sending can be handled by a cache agent, fewer resources in the inference agents are required compared to time events. Additionally, with backing store enabled the schedules are only loaded into memory a batch at a time, reducing the total memory requirement compared to time events.

Creating a Scheduler

Procedure

1. Create a scheduler using the function `Cluster.createScheduler()`.

   You can do this, for example, in a startup rule function, or other rule function or rule:

   ```
   void createScheduler(String schedulerName, long pollInterval, long
   refreshAhead)
   ```

   `schedulerName` - A unique ID for the scheduler you are creating.

   `pollInterval` - TIBCO BusinessEvents checks the scheduler cache every *`pollInterval`* milliseconds, for scheduler work items whose *`scheduledTime`* falls in the current interval.

   `refreshAhead` - Time in milliseconds (into the future) used to pre-load the scheduled events from the backing store.

   Note: In this release, the larger value of *`pollInterval`* and *`refreshAhead`* is used for both these settings.

   For example:

   ```
   Cluster.createScheduler("myScheduler", 1000, 5000);
   ```

2. Schedule the Event to be Sent

Result

You use the function `Cluster.scheduleEvent()` to schedule an event to be sent at a certain time, using a scheduler you created earlier. You can do this in a rule or rule function as needed:

```
void scheduleEvent(StringschedulerName,StringworkKey, SimpleEventevt,
longscheduledTime)
```

Where:

- *`schedulerName`* is the unique ID for the scheduler you created at an earlier time.
- *`workKey`* is a unique key that identifies the work item (that is, the scheduled task). This key can be used to identify the work item later, for example to cancel it.
- *`evt`* is the simple event to be scheduled.

- `scheduledTime` is used to specify the time the event is sent to its default destination. The value is interpreted as an absolute time. To schedule a time relative to the present, use the `System.currentTimeMillis()` function, as shown below.

For example:

```
Cluster.scheduleEvent("MyScheduler", myworkKey, Event.createEvent("xslt://
                      details omitted "),
                      System.currentTimeMillis() + 5000)
```

Working with Advisory Events

You never have to add or configure an event of type Advisory Event. Advisory events are asserted into the Rete network automatically when certain conditions, for example, exceptions, occur. Add the Advisory event type to rule declarations to be notified of such conditions. Then use the event attributes in the rule as needed (Advisory Event Attributes Reference).

For more information on Events, refer to the *TIBCO BusinessEvents Architect Guide*.

Procedure

1. To add an advisory event to a rule, open the rule editor for the rule where you want to use the advisory event.
2. Perform one of the following actions:
   - In the form-based editor, click the plus icon in the Declarations panel and select Advisory Event from the list of resources.
   - In the source editor, add a line like the following in the `declare` block:

     ```
     AdvisoryEvent a;
     ```

     In both cases you can change the alias as desired (from `a` to something else).
3. Save the resource.

Result

The Advisory event type has no properties. You can use its event attributes in rules to return information about an advisory event. See Advisory Event Attributes Reference.

Advisory Event Attributes Reference

Note: When using an event in a rule's form editor, type *`eventname@`* to see the list of its attributes.

The AdvisoryEvent event type has no properties. You can use the following attributes in the rules to return information about an advisory event. Add-on products may use additional advisory event types.

Name	Type	Description
`@id`	`object`	The event's unique internal ID.
`@extId`	`String`	Null.
`@ruleUri`	`String`	The URI of the current rule when the AdvisoryEvent was asserted.
`@ruleScope`	`Object[]`	The scope object of the current rule when the AdvisoryEvent was asserted.
`@category`	`String`	Broad categories of advisory: `Exception`—Used for exceptions not otherwise caught. `Engine`—Used for engine events. Also used for TIBCO BusinessEvents-ActiveMatrix BusinessWorks integration projects. Deployment-Used for hot deployment
`@type`	`String`	Type of advisory within the category. See Attributes Used for Each Type of Advisory Event for details.
`@message`	`String`	Message content depends on the type of advisory event. See Attributes Used for Each Type of Advisory Event for details.

Attributes Used for Each Type of Advisory Event

Description	@category	@type	@message
Exceptions arising from user code	`Exception`	The Java class name of the exception	The stack trace and a message (if the exception includes a message)
TIBCO ActiveMatrix BusinessWorks Activity Failure	`Engine`	`INVOKE BW PROCESS` Indicates a failure or cancellation or timeout of the ActiveMatrix BusinessWorks process.	The error message from the failed ActiveMatrix BusinessWorks process, or the timeout message.
Engine is activated	`Engine`	`engine.primary.activated` Indicates that the engine has been activated.	`Engine` *EngineName* `activated.`
Hot deployment	`Deployment`	`deployment.hotdeploy.success`	Hot deployed project *`ProjectPath`*.
		`deployment.hotdeploy.fail`	Failed to hot deploy project *`ProjectPath`*.
		`deployment.externalclasses.success`	Hot deployed project *`ProjectPath`* using the WebStudio.

Description	@category	@type	@message
		`deployment.externalclasses.failure`	Failed to hot deploy project *`ProjectPath`* using the WebStudio.
Log level updates via MBean or a catalog function	`Engine`	`engine.log.level.updated`	Log level for logger `loggerName` updated to level `logLevel`

Event Property

Event properties can be used in the rule language grammar by following certain rules.

This is the syntax for accessing an event property:

```
eventName.propertyName
```

For example:

```
String x = eventA.customer;
```

where *eventName* is the identifier of the concept instance and *propertyName* is the name of the event property that you want to access.

Concepts

This section explains how to work with concepts, and how to set up relationships between concepts.

Adding a Concept

A concept type is a definition of a set of properties that represent the data fields of an entity.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the folder where you want to store the concept and select **New > Concept**.

   The New Concept wizard opens up.

2. Type the values in the **Concept Name** and **Description** fields.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in TIBCO BusinessEvents Studio explorer and select **Refactor > Rename**.

3. Click **Finish**.

4. In the TIBCO Cloud Events Studio concepts editor:

   - Configure the remaining fields in Configuration section, as explained in Concept Resource Reference.
   - As needed, add and configure properties as explained in Concept Resource Reference.

5. Save the resource.

> **Note:** The ability to add metadata groups and properties is provided for customization purposes and is not documented. Metadata properties are also used for database concepts, a feature available in TIBCO BusinessEvents Data Modeling. See *TIBCO BusinessEvents Data Modeling Developer Guide* for details.

Adding Concept Relationships

See *TIBCO BusinessEvents Architect Guide* for information about concept relationships.

To Create an Inheritance Relationship

In the child concept resource’s Configuration section, identify the parent concept in the **Inherits From** field.

Note that inheritance is set at the concept level. Other relationships are set at the concept property level.

Creating a Containment Relationship

A concept can be contained by only one concept at a time.

Procedure

1. In the container concept resource’s Property section, add and configure a property of type **ContainedConcept**.
2. Configure the property as follows:

   Name: Provide a helpful name. For example, "wheels" could indicate that this concept is contained by a "car" concept (in an appropriate project).

   Multiple: Select the checkbox if this property is an array.

   Policy: Changes Only or All Values.

   History: The number of historical values to keep.

Creating a Reference Relationship

A concept can refer to itself, and can be referred to by more than one concept.

Procedure

1. In the **Property** tab of one of the concept resources, create a property of type `ConceptReference`.
2. Configure the property as follows:

   Name: Provide a helpful name such as `is a lineitem of` to express the relationship.

   Multiple: Select the checkbox if this property is an array.

   Policy: Changes Only or All Values.

   History: The number of historical values to keep.

   See *TIBCO BusinessEvents Architect Guide* for more information on this topic.

Enabling the Concept Reference Serialization in Concept to JSON

Reference concepts are not serialized by default. Set a property in the parent concept metadata to expand them during serialization.
Add the metadata property `tibco.be.schema.ref.expand` for all the concept with the ConceptReference property to true. This enables the serializer to convert the referenced concept to JSON. This property works for In-memory as well as Grid storage type.

If there are multiple level concept reference, then add this property to all the concept with the ConceptReference property.

Procedure

1. In TIBCO BusinessEvents Studio, open the concept for editing.
2. In the concept editor, click **Enable Metadata Configuration** () on top-right corner to enable the metadata properties for editing.

   The **Metadata** section is now enabled.

3. Expand the **Metadata** section and click **Add** to add a new property.

   A new property with the system-generated-name is added.

4. Edit the property name to `tibco.be.schema.ref.expand` and set the value to true.

   In runtime, the serializer read this option and convert the referenced concept to JSON.

5. Save the project and rebuild the EAR.

Editing or Deleting Concept Relationships

You cannot edit or delete the relationship using the concept view graphical user interface. Instead, work with the concept properties directly in the Concept Editor.

Contained Concepts Configuration Properties

You can configure the following property in the CDD file.

Property	**Description**
`be.engine.containedconcept.reader.poolsize`	You can load contained concepts by using a dedicated thread pool instead of the same thread that is running the RTC. Specify this property with the size for the thread pool. If the pool size is not set or set to 1, or less than 1, this thread pool is not created. Default: 0

Concept Resource Reference

Concept resources are descriptive entities similar to the object-oriented concept of a class. They describe a set of properties. For example, one concept might be Department, and it could include department name, manager, and employee properties.

Note: If you are working with a project imported from a release earlier than 5.0.0, you might be able to see metadata properties. However, do not use them. Instead use the settings and properties in the Domain Objects section of the CDD file as needed.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. This field appears on the New Concept Wizard. The name then appears in the title of the concept editor.
Description	No	Short description of the resource. This field appears on both the New Concept Wizard and in the concept editor.
Inherits From	No	If you want this concept to inherit all the properties of another concept, browse to and select that concept. Concepts that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore the following restrictions apply: • If two concepts are related by inheritance, you cannot create a new property in one with a name that is already used in the other.

Field	Global Var?	Description
		• If two unrelated concepts have properties that share a name, an inheritance relationship cannot exist between them.
State Models	No	State models that are owned by this concept. Models can be added and removed as needed. Requires TIBCO BusinessEvents Data Modeling.
Auto Start State Model	No	If selected, when a concept is asserted at runtime, its main state machine (if any) is started. If not selected, then state machines are started using this function in rules: `Instance.startStateMachine()` The function has a parameter specifying whether child concepts' state machines are also started. For contained concepts, the value of the Auto Start State Machine is not relevant and is not selected. Requires TIBCO BusinessEvents Data Modeling.

Properties

The Properties tab has the following fields:

Field	Global Var?	Description
Name	No	The property name. **Note:** The property name cannot begin with the character _.

Field	Global Var?	Description
Type	No	Any of the following types: `String`, `Integer`, `Long`, `Double`, `Boolean`, `DateTime`, `ContainedConcept`, `Concept`Reference When you create a property of type `ContainedConcept`, you are creating a containment relationship. The concept that you are currently configuring is the container; the concept you specify as a property is the contained concept. **Note:** A concept cannot contain itself. For instance, for a concept A we cannot add a contained property of type A. When you create a property of type `ConceptReference` you are creating a property that references another concept. See *TIBCO BusinessEvents Architect Guide* for more details. **Note:** For properties of type Double, when a backing store is used, all NaN (Not a Number) values are converted to `0.00`.
Multiple	No	Select the Multiple checkbox if this property is an array. Consider, for example, an Order concept: In most cases, an Order concept would allow only one value for the customer property but multiple values for the `line_item` property. Selecting the Multiple checkbox creates a property array.
Policy	No	TIBCO BusinessEvents can record historical values using either of these policies: **Changes Only** TIBCO BusinessEvents records the value of the property every time it changes to a new value. **All Values** TIBCO BusinessEvents records the value of the property every

Field	Global Var?	Description
		time an action sets the value even if the new value is the same as the old value.
History	No	Determines if historical values are stored, and if so how many. The default maximum value is 32767. Zero (0): TIBCO BusinessEvents does not store historical values for the concept. It stores the value without a time and date stamp One or more (>0): TIBCO BusinessEvents stores the property value when the property changes, along with a date and timestamp, up to the number specified. When the maximum history size is reached, the oldest values are discarded as new values are recorded. See *TIBCO BusinessEvents Architect Guide* for more details. **Note:** • History enabled columns cannot be used as a primary key. • Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.
Domain	No	Domain associated with the property.

Metadata

The Metadata section is used in a special way with database concepts. Requires TIBCO BusinessEvents Data Modeling. See *TIBCO BusinessEvents Data Modeling Developer Guide* for details. It is not used otherwise (except for customization, which is not documented).

Concept Attributes Reference

Note: When using a concept in a rule's form editor, type `*conceptname*.@` to see the list of its attributes.

You can use the following attributes in the rules to return information about a concept instance.

Entity	Attributes	Type	Returns
Concept	@id	object	The concept instance has unique internal ID.
	@extId	string	The concept instance has unique external ID. Optional. The value of the `extId` is set at creation time (for example, using the `Instance.createInstance()` function) and cannot be changed after that. **Note:** The `extId` value (if set) must be unique across all objects in the cluster. **Tip:** You can use the property `Agent.`*`AgentClassName`*`.checkDuplicates` to check duplicate `extIds` across the cluster. You can use this property only when the legacy ID lookup strategy is enabled.
ContainedConcept	@id	object	The contained concept instance's unique internal ID.
	@extId	string	The contained concept instance's unique external ID.

Entity	Attributes	Type	Returns
	@parent	concept	The parent concept instance. (This is treated as a concept reference in the language.)
ConceptReference	@id	object	The contained concept instance's unique internal ID.
	@extId	string	The contained concept instance's unique external ID.
	@isSet	boolean	Available for any concept property (not for referenced concepts). For example, you can use: `acc.Balance@isSet`. It indicates if a property is assigned or not. If a property is assigned, then `@isSet` returns `true`.

Note: With the new ID (Key-based Lookup), when a new contained or referenced Concept with the same extId is reassigned to the parent concept, the new contained concept fails to get assigned. You must set a contained or referenced concept property to null before deleting a contained or referenced concept:

Example:

```
Instance.PropertyAtom.setContainedConcept(testcon.TestCon_property_
6,null,1L);
          Instance.deleteInstance(cont);
          // cont1 = new contained concept with same id
         Instance.PropertyAtom.setContainedConcept(testcon.TestCon_
property_6,cont1,1L);
```

Note: When using the new ID (Key-based Lookup), reinstantiating a concept with the same extID without deleting the first concept does not work.

Adding a Scorecard

Configuring a scorecard is similar to configuring a concept, except that scorecards do not have relationships. Scorecards, therefore, have none of the properties used for setting up relationships. Scorecard properties can be of any primitive type.

See Concept Resource Reference for details about scorecard properties

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the folder where you want to store the scorecard and select **New > Scorecard**. You see the New Scorecard Wizard.
2. Type in the values for the Scorecard **Name** and **Description** fields.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in TIBCO BusinessEvents Studio Explorer and select **Refactor > Rename**.

3. Click **Finish**.
4. In the Scorecard editor, add and configure properties as explained in Concept Resource Reference.
5. Save the resource.

   Note: The ability to add metadata groups and properties is provided for customization purposes and is not documented.

What to do next

After configuring a scorecard resource, use rules to gather the information you need in the scorecard. To access the scorecard in a rule, use the following syntax:

```
folder.folder.scorecard.property
```

For Example:

```
int i = SalesFolder.StatsScorecard.numOrdersProperty;
```

Using a Scorecard in Rules

After configuring a scorecard resource, use rules to gather the information you need in the scorecard. To access the scorecard in a rule, use this syntax:

```
folder.folder.scorecard.property
```

For Example:

```
int i = SalesFolder.StatsScorecard.numOrdersProperty;
```

Scorecard Resource Reference

Scorecard configuration is the same as concept configuration, except that scorecards have no relationships with each other. Scorecards, therefore, have none of the properties used for setting up relationships. Scorecard properties can be of any primitive type.

See Concept Resource Reference for details about scorecard properties.

Accessing a Concept PropertyAtom

Access to a concept *PropertyAtom* is defined with the syntax.

Syntax for accessing a concept property:

```
instanceName.propertyName
```

where *instanceName* is the identifier of the concept instance, and *propertyName* is the name of the concept property that you want to access.

For example, to get the current value of the cost *PropertyAtom*:

```
int x = instanceA.cost;
```

For example, to set a value with the current system timestamp:

```
instanceA.cost = value;
```

Note: If the history size is `0`, TIBCO BusinessEvents does not record a timestamp.

Note: Rule dependency and rule condition re-evaluation are performed only on concept properties.

This re-evaluation is not performed on an attribute such as `@parent`.

Get and Set PropertyAtom Value With User-Specified Time

PropertyAtom can be set for the rule language grammar.

You can get and set PropertyAtom values as follows:

- Specify a time and get the PropertyAtom value stored in the history at that time using one of the standard functions:

  ```
   type Instance.PropertyAtom.gettype(PropertyAtom propertyAtomName, \
  long time)
  ```

  where *type* is the type of the PropertyAtom, *propertyAtomName* is the name of the PropertyAtom, and *time* is the time from which you want to retrieve the value.

- Set a value in the PropertyAtom History using one of the standard functions:

  ```
  Instance.PropertyAtom.settype(PropertyAtom propertyAtomName, \
                                type value, long time)
  ```

  where *type* is the type of the PropertyAtom and the type of the new value, *propertyAtomName* is the name of the PropertyAtom, *value* is the value to store in the ring buffer, and *time* is the timestamp for the new entry.

  TIBCO BusinessEvents manages these requests as follows:

  - If the ring buffer has vacancies, TIBCO BusinessEvents inserts the new entry into the correct place based on its timestamp, shifts the older values out one place, and returns True.
  - If the ring buffer is full, and the new value has a more recent timestamp than the oldest value, TIBCO BusinessEvents inserts the new value into the correct place, shifts older values if necessary, drops the oldest value, and returns True.
  - If the ring buffer is full, and the new value has a timestamp that is older than the oldest value in the ring buffer, TIBCO BusinessEvents does not insert the new value into the ring buffer, and it returns False.

Concept Property Arrays

To work with concept property arrays, you need to follow certain rules.

Accessing a Concept Property Array

This is the syntax for accessing a concept property array:

```
instanceName.propertyName
```

where *instanceName* is the identifier of the concept instance, and *propertyName* is the name of the concept property that you want to access.

Accessing a Value in a Concept Property Array

To access a value in a property array, identify the position in the array of the value as shown:

```
instanceName.propertyName[indexPosition]
```

For example:

```
String x = instanceA.lineItem[0];
```

This gets the current value of the first property atom in the array `lineItem`, and assigns it to the local variable, x.

Note: Array index difference: In the TIBCO BusinessEvents language, array indexes start from zero (0). However, in XSLT and XPath languages, they start from one (1). It is important to remember this difference when using the rule language in the rule editor, and when working in the XSLT mapper and the XPath builder.

Setting the Value for an Existing Concept Property Array Position

You can set the value of an existing position in an array. For example:

```
int[] ii = {1,2,3};
ii[2] = 1;
```

Adding a Value to a Concept Property Array

You can append a value to the end of a property array. You cannot, however, add a value to any other position in an array. This is the syntax:

```
instanceName.propertyName[indexPosition] = value
```

To use the syntax shown above, you must know the index position of the end of the array. You can append a value to the end of an array without knowing the index position of the end of the array using the @length attribute as shown:

```
instanceName.propertyName[instanceName.propertyName@length] = value
```

Deleting Values in a Property Array

This is the syntax for deleting an array property:

```
Instance.PropertyArray.delete(instanceName.propertyName,indexPosition);
```

Array identifier numbers are positional. When history is not tracked and you delete multiple items in an array (using a for loop), delete higher position numbers before lower position numbers to ensure the correct entries are deleted. For example, if you want to delete items in positions 1, 2, and 4, delete them in this order: 4, 2, and then 1.

The array entry that held a deleted concept is removed, reducing the array size by one, and reducing by one the index of every entry in the array at a higher index than the deleted one. If you delete entries with lower position numbers first, the remaining lower numbers are then occupied by different items.

For example, if you delete the item in position 1, then the item that was in position `2` is now in position `1`.

Checking if the Multiple Property of a Concept is Empty

If you have a concept (for example, CustomerConcept) which contains multiple properties (for example, address) and you did not pass any value to the address property when creating the concept then the following check always returns false:

```
customerConcept.address == null
```

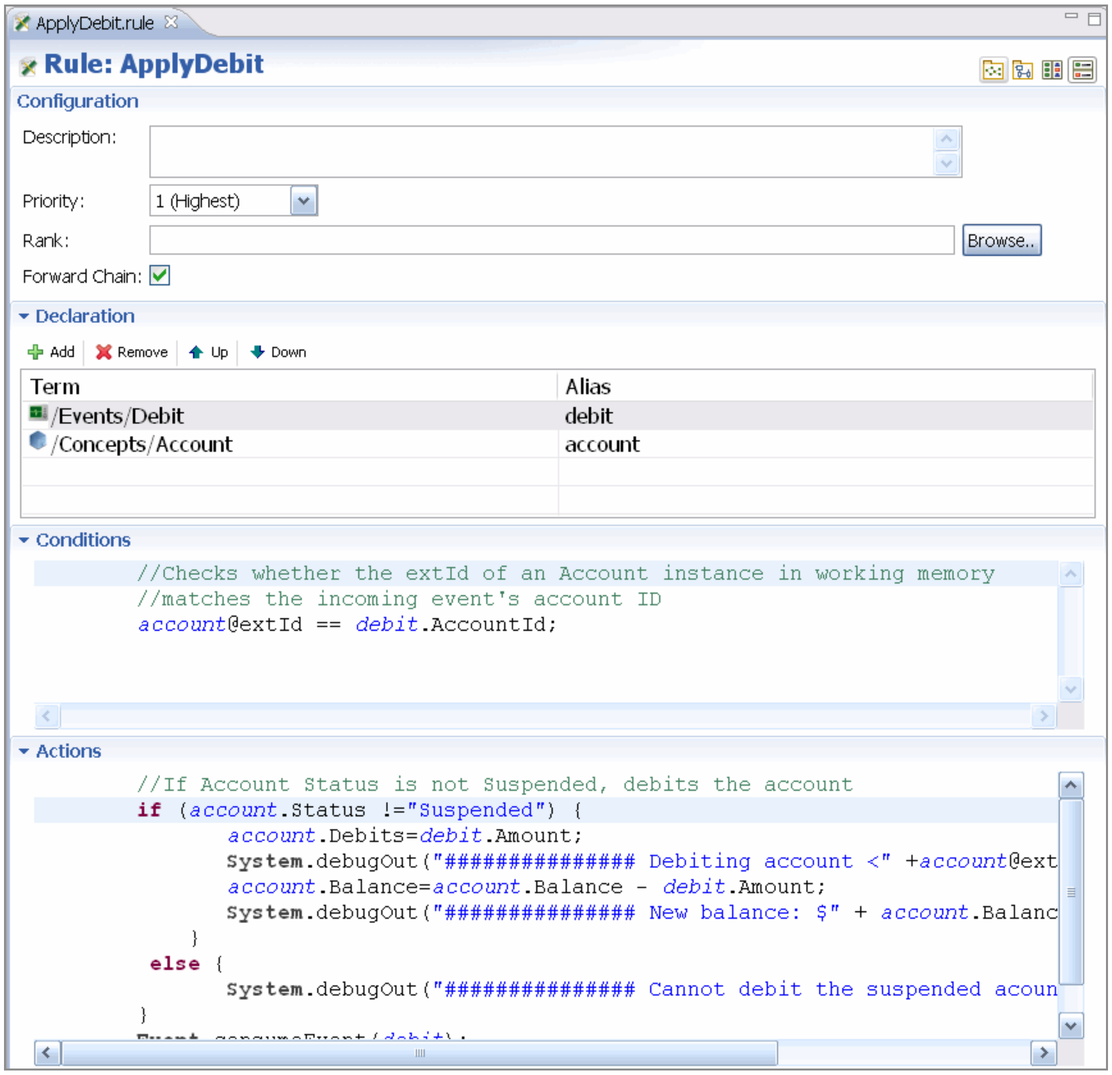
The reason for this is that the multiple check on a concept property (simple property, ContainedConcept, or ConceptReference) makes the property an array. Even if you do not add any property, ContainedConcept, or ConceptReference to the multiple properties, the property is an array with the length zero but not null.

Thus, to resolve this, for a multiple property (simple property, ContainedConcept, or ConceptReference), check the length of the property (array). The length zero indicates that there is no simple property, ContainedConcept, or ConceptReference.

Rules and Functions

If you are using the source editor, adapt these instructions that focus on the form-based editor and mention the equivalent settings in the source editor.

Figure 7: Rule Form Editor



ApplyDebit.rule

Rule: ApplyDebit

Configuration

Description:	
Priority:	1 (Highest)
Rank:	
Forward Chain:	✓

Declaration

Add Remove Up Down

Term	Alias
/Events/Debit	debit
/Concepts/Account	account

Conditions

```
        //Checks whether the extId of an Account instance in working memory
        //matches the incoming event's account ID
        account@extId == debit.AccountId;
```

Actions

```
        //If Account Status is not Suspended, debits the account
        if (account.Status !="Suspended") {
                account.Debits=debit.Amount;
                System.debugOut("############### Debiting account <" +account@ext
                account.Balance=account.Balance - debit.Amount;
                System.debugOut("############### New balance: $" + account.Balanc
            }
         else {
                System.debugOut("############### Cannot debit the suspended acoun
        }
```

Rule Source Editor

```
  * @description
  * @author
  */
rule Rules.ProcessDebits.ApplyDebit {
  attribute {
    priority = 1;
    forwardChain = true;
```
Optionally add entry for Rank as needed (`Rank=`*RuleFunction), or enter in Form view.*
```
  }
  declare {
    Events.Debit  debit;
    Concepts.Account  account;
  }
  when {
    //Checks whether the extId of an Account instance in working memory
    //matches the incoming event's account ID
    account@extId == debit.AccountId;
  }
  then {
    //If Account Status is not Suspended, debits the account
    if (account.Status !="Suspended") {
    account.Debits=debit.Amount;
    System.debugOut("############### Debiting account <" +account@extId+
"> by $" +debit.Amount);
    account.Balance=account.Balance - debit.Amount;
    System.debugOut("############### New balance: $" + account.Balance);
    }
    else {
    System.debugOut("############### Cannot debit the suspended acount
<" +account@extId +">");
    }
    Event.consumeEvent(debit);
  }
}
```

Adding a Rule

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the folder where you want to store a rule and select **New > Rule**. You see the New Rule Wizard.
2. In the **Rule Name** field, type a name for the rule. In the **Description** field, type a description as desired. (In the source editor the description appears in the `@description` line of the comments at the top of the editor.)

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in TIBCO BusinessEvents Studio Explorer and select **Refactor > Rename**.

3. Click **Finish**. By default, you see the source rule editor on opening the editor. It shows the outline for a rule's source code. Click the **Form** tab at the bottom of the editor to use the form editor.

   At any time, you can click the **Form** and **Source** tabs at the bottom of the editor to switch between the form-based editor and the source editor as you work, depending on your preference.

   Tip: Rule Editor Preference

   To set which mode the editor uses on the first opening, go to **Window > Preferences > TIBCO BusinessEvents > Rules** and select or deselect the **Initially show 'Form'** tab checkbox as desired.

4. In the Form editor Configuration section, add or edit a description as desired. (In the source editor the description appears in the `* @description` line of the comments at the top of the editor.)

5. To control the order of rule execution:

   a. To control the order in which the rules run, set the **Priority** field accordingly. The highest priority is 1. (In the source editor set `priority = `*`n`* where n is the priority number.)

      Also, to control the order in which rules with the same *priority* run, set the Rank field accordingly. Browse to and select the rule function you created for this purpose. (To use the source editor, add `Rank=`*`RuleFunction`* in the list of rule attributes.) See Using Priority and Rank to Control Order of Rule Execution for details.

6. To disable forward rule chaining (the default behavior), uncheck the Forward Chain checkbox. (In the source editor set `forwardChain = false`.)

7. In the Declarations section (equivalent to the `declare` statements in the source editor), drag an ontology entity into the section, or perform the following actions.
    a. Click **Add** to add the resources that you are using in your rule. You see the Select Rule Declaration Arguments dialog.
    b. In the upper half of the Select Rule Declaration Arguments dialog, select the kind of entity you want to use.
    c. In the lower half of the dialog, select a resource from the filtered ontology tree, and click **OK**. Your selection appears in the Declarations list. TIBCO BusinessEvents assigns an alias to the resource. You can edit the alias.
    d. To reorder the declarations, highlight a declaration and click the up or down arrow to move it. This is relevant only in rule functions, to order the arguments.

       Repeat to add more entity types as desired.
8. In the Conditions section (equivalent to the `when` statements in the source editor), write the condition statements (in the TIBCO BusinessEvents rule language).

   Each line is a complete statement. Each condition must evaluate to a Boolean value. Each line is joined to the others with an implicit AND operator. All of a rule's conditions must be evaluated true for the conditions to be satisfied. Refer to the "Order of Evaluation of Rule Conditions" section in the *TIBCO BusinessEvents Architect Guide* for more information.

   See Using Variables and Functions in the Rule Editor for more information on working in the rule editor.
9. In the Actions section (equivalent to the `then` statements in the source editor), write action statements (in the TIBCO BusinessEvents rules language).
10. Save the resource.

Rule Editor Reference

The rule editor and rule function editor are similar. This section focuses on the form-based rule editor. You can adapt the information in the Area and Property section to apply to the different blocks of code in the source editor.

Property	Description
Configuration Section	
Name (Wizard only)	The name to appear as the label for the resource. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements.
Description (Editor and Wizard)	Short description of the resource.
Priority (Editor and Wizard)	Specify a value between 1 and 10, where 1 is the highest priority and 10 is the lowest priority. Rules with a higher priority execute before rules with a lower priority. See also Rank. Only set priority or rank where there is a reason to control the order of rules. As a general practice, it is more efficient to let the engine determine the order of rule firing.
Rank	Specifies a rule function that controls the order of execution of rules with the same priority. Validity must include Condition (that is, do not select Action Only as the validity.) See Using Priority and Rank to Control Order of Rule Execution for details. Default is 0.0.
Forward Chain	Determines if the rule is used in forward chaining. If the checkbox is not selected, changes made by this rule won't trigger other rules. By default the checkbox is selected
Declaration Section	
Term	A concept or event type in the project that you will use in your rule. Types you add to the declaration define the scope of the rule. For example, a concept definition such as `/Concepts/Accounts/CheckAccount`.

Property	Description
	Note: Do not use the generic "Event" as event type, instead use "SimpleEvent", "TimeEvent", or any other event type defined in the project. It is not necessary to add scorecards in the declaration in order to use them in the rule.
Alias	A name used to refer to the scope element in the body of the rule. You can change the alias. Like resource names, aliases must be valid identifiers See Identifier Naming Requirements.
Conditions Section	
	Each line in the Conditions area is a single expression that evaluates to `true` or `false`. Each line is joined to the others with an implicit `and` operator. For the OR operator, use a double pipe (||) on the same line. TIBCO BusinessEvents evaluates single conditions from left to right. TIBCO BusinessEvents optimizes the evaluation of multiple conditions (refer to the "Order of Evaluation of Rule Conditions" section in *TIBCO BusinessEvents Architect Guide*).
Actions Section	
	List of statements that will be executed when the rule is fired.

Tips for Working in the Rule Editor

Here are some tips for working with the Rule Editor.

Switching between Form and Source Editors

You can freely switch between form and source editors for rules and rule functions. In each case the editors stay synchronized with the latest changes.

You cannot switch from the source editor to the form editor if there are any syntax or resolution errors in your code.

The Priority setting

It is used by the runtime engine when determining the order in which rules are fired. Rules with a number closer to one fire first. When there is no reason to force rules to run in a particular order, leave the Priority set to the default and let the runtime engine determine the rule order.

Declaring multiple terms of the same type

It allows the rule to consider multiple instances of the corresponding entity. Specify different aliases to keep the terms distinct.

Scorecards

Scorecards are like concepts except that there is only one instance of a scorecard (or more accurately, one instance per agent when multi-engine features are used). It is therefore not necessary to put scorecards in the declaration of a rule because a scorecard never requires an alias. You can use scorecard properties in conditions (just as you would concept properties). However, because a scorecard does not have an alias, refer to it like a function, for example, `Folder.Folder.Scorecard.prop1`

Standard Eclipse Features

In addition to some TIBCO BusinessEvents-specific features, the source and form rule editors support standard Eclipse functionality such as the following: Undo and redo; copy and paste; breakpoint features; standard text annotations (which can be changed using Preferences); text folding (source editor only); Java outline view (source editor only).

✓ **Tip:** When you are working in the source editor, press **Ctrl+Shift+L** to see a list of keyboard shortcuts available in that context. (This is a general Eclipse feature.)

Information Highlighted

Keywords, variables, and functions are highlighted in the text. Also, when you hover the mouse over a resource such as a concept, event or function, information about it displays in a tooltip.

Syntax and resolution errors are automatically flagged by visual cues. They are underlined and also display in the vertical and overview rulers.

Miscellaneous Tips

Some other tips are highlighted below.

Tips for Working in the Rule Editor

To do this...	Do this...
Switch Between Source and Form Editors	Click the bottom tab to switch between the source and form editors. The code remains synchronized. However, you cannot switch if there are errors in the code. First resolve the errors, then switch.
To Use Content Assist to Complete Values	The content assist feature helps you complete values using information that is available in resources. For example, if you type the name of a concept type (or its alias) and then a period, a list of the concept type properties appears for you to select from. The selection list also appears when the cursor is in an appropriate location and you press **Ctrl+Space**, or if you right-click and select **Edit > Content Assist** from the context (right-click) menu.
To Comment (and Uncomment) a Line	To comment out a line, or uncomment a line, press **Ctrl+/** or select **Edit > Toggle Comment** from the context menu.
To Search for References	When the cursor is placed in an appropriate item in the code such as an entity or function name, you can find all references to that item references in the rule code. Press **Ctrl+Shift+G**, or select **Search > Search for References** from the context (right-click) menu. The item references are highlighted in the text, and an arrow appears in the vertical ruler.
To Jump to the Definition of an Item	To jump to the location where an item is defined, you can use two methods. • Click the item name and press **F3**, or right-click and select Open Declaration.

To do this...	Do this...
	• Press and hold the Control key while you move (hover) the mouse over the text. When you hover over an item that displays an underline, **Ctrl+click** the item to jump to the place where it is defined. For example, you would jump from an alias to the declaration, and from an entity or entity property to the entity's editor.

Adding a Rule Function

Regular rule functions have arguments and a body containing the code for the function. Virtual rule functions have arguments but no body. The implementation for virtual rule functions is provided by one or more decision tables. See *TIBCO BusinessEvents® Decision Manager User Guide* for more details.

Adding a Rule Function

See Rule Function Resource Reference for details about completing values.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the folder where you want to store the rule function and select **New > Rule Function**. You see the New Rule Function Wizard.

   a. In the **Rule Function Name** field, type a name for the rule function.

   b. In the **Description** field, type a description. (In the source editor the description appears in the `* @description` line of the comments at the top of the editor).

   c. Set the return type for the rule function. Default is `void`. Browse to select a return type as needed. For return types that require additional configuration, such as `ContainedConcept`, complete the configuration in the Rule Function editor.

   d. If you want this rule function to be a virtual rule function (to be implemented by a decision table), select the **Virtual** checkbox.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in TIBCO BusinessEvents Studio Explorer and select **Refactor > Rename**. See Element Refactoring Operations for more details.

2. Click **Finish**. If you see the source editor, click the **Form** tab at the bottom of the editor to use the form editor as desired.

   At any time you can click the **Form** and **Source** tabs at the bottom of the editor to switch between the form-based editor and the source editor as you work, depending on your preference.

   Tip: Rule Function Editor Preference

   To set which mode the editor uses upon first opening, go to **Window > Preferences > TIBCO BusinessEvents > Rules** and select or deselect the desired checkbox: **Initially show ’Form’ tab in Rule Function Editor**.

3. In the Form editor Configuration section, add or edit an alias and a description as desired. (In the source editor the description appears in the `* @description` line of the comments at the top of the editor and the Alias appears in the `attribute` list.).

4. If you did not do so in the Wizard, use the Return Type to select the return type of the rule function.

5. If you did not do so in the Wizard, set the Virtual checkbox according to your need. Select the checkbox if you are creating a virtual rule function (to be implemented by a decision table).

   Tip: In the source editor, the signature of a virtual rule function is:

   ```
       virtual void rulefunction folder.RFName
   ```

   Do not add code to the Body block in the source editor of a virtual rule function. If you do, you see error messages if you try to save or to switch to the form-based editor.

6. From the Validity drop-down list, select the value that specifies where the rule function can be used (source editor `attribute` equivalents shown in parentheses):

   - Action (`validity=ACTION`)
   - Action and Condition (`validity=CONDITION`)
   - Action, Condition and Query (`validity=QUERY`)

     Virtual rule functions have a non-editable validity setting of Action.

7. If the rule function returns a value, specify the Return Type, otherwise leave this field set to `void`. (Appears in the signature of the rule function in the Source editor.)

   Virtual rule functions have a non-editable return type of Void.

8. In the Scope section (`scope` statements in the source editor) you define the arguments of the rule function. Drag entities into the Scope area from TIBCO BusinessEvents Studio Explorer, or perform the following actions.
   a. Click **Add** to add resources that you are using in your rule function. You see the Select Rule Function Scope Arguments dialog.
   b. In the upper half of the Select Rule Function Scope Arguments dialog, select the type you want to use.
   c. To specify an array, select the **isArray** checkbox. (You can specify a variable array in the source editor in the usual way, for example, `int[] myArr`.)
   d. If the type you select is an ontology type, in the lower half of the dialog, select a resource from the filtered ontology tree.
   e. Click **OK**.

      Your selection appears in the list. TIBCO BusinessEvents assigns an alias to it. You can edit the alias.

      Add more entities as needed.
9. Add more arguments as needed, and use the up and down arrows to order the arguments as needed.
10. In the Body section (`Body` statements in the source editor), use the TIBCO BusinessEvents rule language to implement the function. (Virtual rule functions have only a signature, and no implementation at design time.)

    See Using Variables and Functions in the Rule Editor for more information on working in the rule editor.
11. Save the resource.

Rule Function Resource Reference

Rule Function resources enable you to write rule functions that you can use in rules, as startup and shutdown actions, and as preprocessors.

Virtual rule functions are decorated with a V.

Property	Description
Configuration Section	
Name (Wizard only)	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements.
Description (Editor and Wizard)	Short description of the resource.
Alias	Optionally, enter an alias for the rule function. Used as a short way to refer to the rule function. You can use the alias, for example, to make query strings shorter.
Virtual (Editor and Wizard)	If set to yes, the rule function is a virtual rule function. Virtual rule functions have arguments but no body. The **Body** section and the **Return Type** field are disabled. The body is provided by a decision table. See *TIBCO BusinessEvents Decision Manager User Guide*.
Validity	Specifies where the rule function can be used. Possible values are as follows: **Action** Indicates that this rule function can be used only in the Action block of a rule. **Action and Condition** Indicates that this rule function can be used in the Action and Condition blocks of a rule. **Action, Condition and Query** Indicates that this rule function can be used in the Action and Condition blocks of a rule, and can also be used in the text of a query (The query language features are available only in TIBCO BusinessEvents Enterprise Suite).

Property	Description
	Note: Only Action rule functions can be used as startup rule functions or shutdown rule functions.
Return Type (Editor and Wizard)	If the rule function returns a value, specify the Return Type, otherwise leave set to void.
Scope Section	
Term	The type of the argument. Arguments and return type can be any of the following, including arrays of these datatypes: • Primitive, that is any of: `String`, `int`, `long`, `double`, `boolean`, `DateTime`, Object • Concept • Event • Specific type of Concept • Specific type of Event The Object data type is used to pass parameters between standard and user-defined functions and external Java sources.
Alias	Each argument requires a type and an alias. Names must be valid identifiers.
Body Section	
	List of statements that will be executed when the rule function executes.

Using Variables and Functions in the Rule Editor

This section provides some tips on working in the rule editor. The rule editor is used for TIBCO BusinessEvents rules, rule functions, and state machine rules.

Using Catalog Functions in the Rule Editor

To use catalog functions in an editor choose one of the following methods.

Procedure

1. When adding code in the rule editor, perform one of the following actions.
    - Type the function name, starting with the folder path to the function.

      Do not enter the catalog name (which appears in the list of catalog functions in the Catalog Functions view, and looks like a top-level folder). Type a period at the end of the folder name. A popup window shows all folders and functions within the folder whose name you typed. For example type `Database.` to see a list of all functions in the Database folder.
    - Open the Catalog Functions view. To open the view click as follows:

      Window > Show View > Other > TIBCO BusinessEvents > Catalog Functions.

      Drill down on categories within the catalog to expand to lists of functions and drag the desired function to the rule editor.
2. Provide parameter values as indicated by the tooltip. You can hover the cursor over a function to display a tooltip showing the function’s arguments. You can also see the tooltip contents in the online reference, TIBCO BusinessEvents Functions Reference.

Result

See *TIBCO BusinessEvents Architect Guide* for descriptions of the various function catalogs, and an explanation of the decorations that appear on many function names. For information about configuring mapper functions see Using the Function Argument Mapping Wizard.

Using Global Variables in the Rule Editor

To use a global variable in the rule editor, use one of the `System.getGlobalVariableAs*` functions. For example:

```
System.getGlobalVariableAsString("Hostname", "Localhost")
```

Where Hostname is the name of the variable and Localhost is an optional literal value to use if the variable is not found.

Do not use this syntax: *`%%Global.Variable.Name%%`*.

See Working with Global Variables for more details about global variables.

Using the Function Argument Mapping Wizard

For functions known as mapper functions you can use the Function Argument Mapping wizard to map inputs from a source to the function arguments.
See Mapping and Transforming Data for a reference to using the Function Argument Mapping wizard.

Note: In the TIBCO BusinessEvents language, array indexes start from zero (0). However, in XSLT and XPath languages, they start from one (1). It's important to remember this difference when using the rule language in the rule editor, and when working in the XSLT mapper and the XPath builder.

Procedure

1. Open the Function Argument Mapping wizard in either of the following ways:
    - In the rule editor, type the category of function you want to use and select a mapper function from the list of functions in that category. The function argument area contains the text `"xslt://"`. Ctrl+click the text `"xslt://"` to open the Function Argument Mapping wizard.
    - Type the name of a mapper function category into the **Actions** or **Conditions** areas, then type an open parenthesis ("(") TIBCO BusinessEvents displays `"xslt://"`. Ctrl+click the text `"xslt://"` to open the Function Argument Mapping wizard.
    - Type the entire string to specify the function category path and name, followed by `("xslt://")`. Then control-click the text `"xslt://"` to open the Function Argument Mapping wizard.
2. In the Function Argument Mapping Wizard, under the Function section, take appropriate action. For example, if you are using `Instance.createInstance()`, click the **Browse** button to the right of the **Entity Path** field and select a concept type so you can create an instance of it.

3. Under the **Input** section, on the left side, TIBCO BusinessEvents displays a list of variables associated with the project, and on the right, a list of properties for the selected resource type.
4. Drill down and expand the lists on both sides to expose the variables and properties.

5. Drag variables from the left to the appropriate property on the right.

Figure 8: The Function Argument Mapping Wizard in XPath 1.0 Project

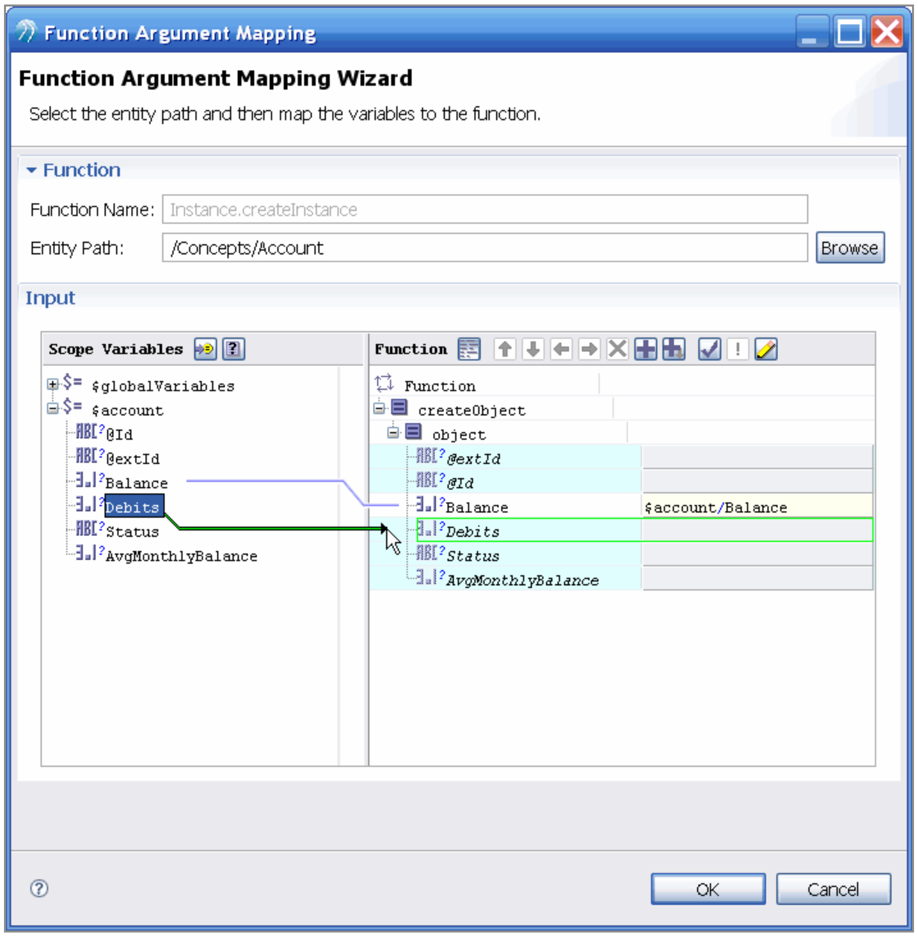
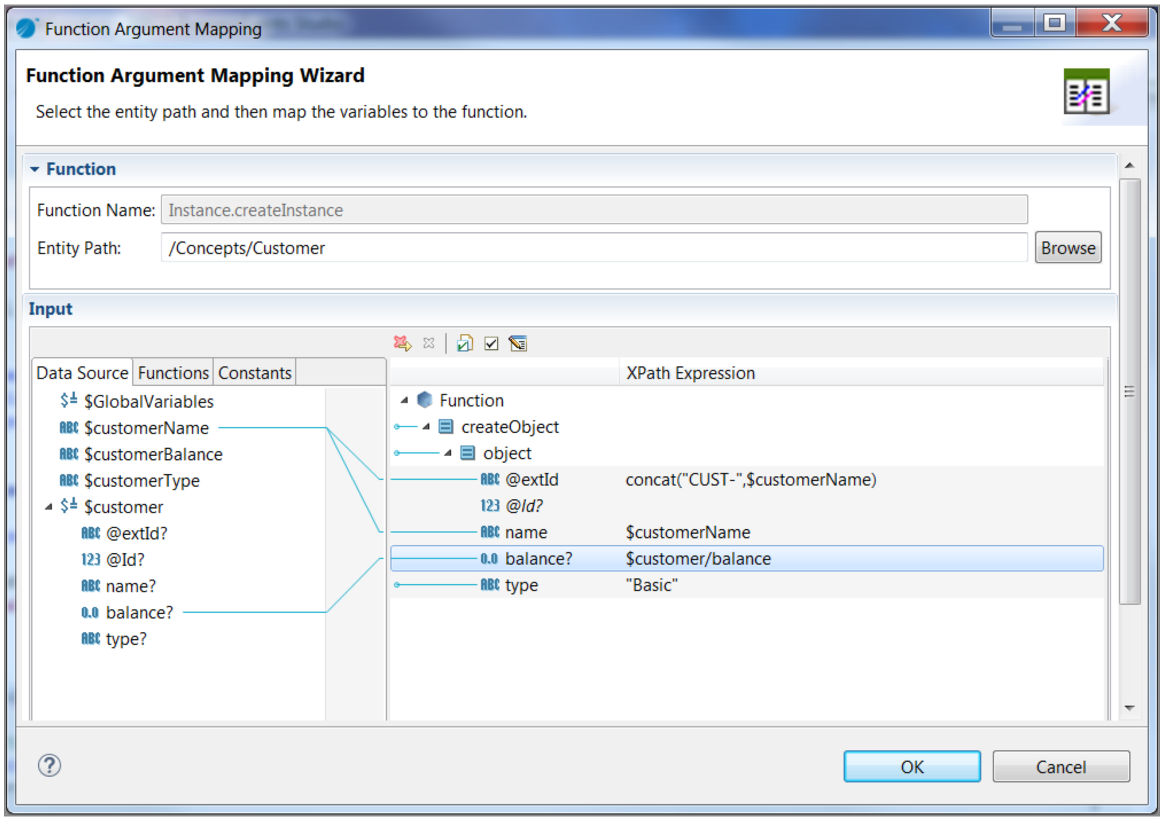


Figure 9: The Function Argument Mapping Wizard in XPath 2.0 Project



	XPath Expression
@extId	concat("CUST-",$customerName)
@Id?	
name	$customerName
balance?	$customer/balance
type	"Basic"

6. (XPath 1.0) If you want to define the arguments using more complex logic, type the code or click the **XPath Formula Builder** icon to use the XPath Formula Builder. See XPath Formula Builder For XPath 1.0.

7. (XPath 2.0) If you want to define the arguments using more complex logic, type the code in the **XPath Expression** column or click the **Show/Hide Edit Tab** icon to display the **Edit Statement** tab. On the left side, click the **Functions** tab to see the available mapper functions and then drag and drop the functions to the **XPath Expression** text box. See XPath Formula Builder For XPath 1.0.

8. Click **OK**.

Using Priority and Rank to Control Order of Rule Execution

For each RTC, the rule agenda is sorted by priority and then within priority by rank, for those rules that use the same ranking. Use of priority and rank is optional. You can also use priority without using rank.

In the rule’s **Rank** field (or rule attribute, in the source view), you specify a rule function that returns a double. The larger the return value, the higher the ranking. You can specify the same rule function in different rules to perform ranking across tuples of those rules. Here are the requirements:

- The rule function must have a Validity setting that includes Condition (that is, do not set it to Action Only).
- You can assign the same rule function to different rules as long as the following is true:
    - The scope of the rule function includes only parameters found in all the rules that use the same function. Rank rule function scope must match the rule declaration exactly and in the same order(As with rules, primitives are not allowed.) For example, if the rule has EventA and ConceptB as the scope, the rule function must also have EventA and ConceptB (in that order) as the only two parameters.
    - The parameters must be used in the same order as they appear in the rule declaration.
- The rule function must return a double value. (The default value for the **Rank** field is 0.0.)

Examples

For example, suppose that two rules declare CustName and SupportLevel. You assign the same rule function to both rules. The function returns 3.0 for Gold support level, 2.0 for Silver, and 1.0 for Bronze. As a result, among the rule tuples with priority 1, those for customers with Gold support run before those for customers with Silver support, which run before those for customers with Bronze support.

Below is an example showing how rule priority and ranking determine the sort order. Suppose you assign the same rule function for ranking rules 1, 2, and 3. At run time these are some rule tuples to be sorted in the agenda for an RTC:

Rule	Priority	Rank
Rule 1 (Tuple X)	Priority: 5	Rank: 10.0

Rule	Priority	Rank
Rule 2 (Tuple A, B)	Priority: 1	Rank: 1.0
Rule 2 (Tuple A, C)	Priority: 1	Rank: -1.0
Rule 2 (Tuple A, D)	Priority: 1	Rank: -2.0
Rule 3 (Tuple A)	Priority: 4	Rank: 0.0

They are sorted and run as follows. (This could change during a conflict resolution cycle depending on the effect of rule actions.)

Rule	Priority	Rank
Rule 2 (Tuple A, B)	Priority: 1	Rank: 1.0
Rule 2 (Tuple A, C)	Priority: 1	Rank: -1.0
Rule 2 (Tuple A, D)	Priority: 1	Rank: -2.0
Rule 3 (Tuple A)	Priority: 4	Rank: 0.0
Rule 1 (Tuple X)	Priority: 5	Rank: 10.0

Reverse Order Example

If you want the rule with the smallest return value to be ranked highest, multiply by negative one (-1.0) to reverse the size of the values. For example, suppose you want rules that were asserted earlier to run before those that were asserted later. The time values returned by the three rules are 100, 150, and 300. Using -1.0 * *`time`*), they are fired in the desired order: -100.0 (the largest value), then -150.0, then -300.0.

Using the Quick Fix Feature in the Rule Editor

The Quick Fix feature enables you to create concepts, events, and rule functions without leaving the rule editor. It also enables you to add properties to existing concepts and events. The feature is available when an unknown reference appears in the rule or rule function code.

Note: This feature is also used in the state modeler feature, available in TIBCO BusinessEvents Data Modeling, wherever the rule editor is used.

If a qualified name has multiple unresolved references, for example, `Concepts.Something.Somethingelse`, the Quick Fix feature only applies to the first unresolved reference in the name.

Using the Quick Fix Feature

Procedure

1. Type the code in the rule editor as if the entity, property, or rule function you want to use already exists. See Quick Fix Feature Options for some examples.

   You see a lightbulb icon in the left margin on any line where there are unknown references that you can configure using Quick Fix.

2. To use the available Quick Fix options, click the lightbulb icon or press Ctrl+1. The options are explained in Quick Fix Feature Options:

Result

Quick Fix Feature Options

Unknown Reference	Quick Fix Editor
Unqualified name. Example: `SomeName s;`	A list of appropriate wizards appears so you can create a new concept, simple event, or time event. The wizard you select opens so you can define the entity.
Unqualified name followed by parentheses. Example: `SomeName()`	A link to the rule function creation wizard appears. Click it to open the wizard and define the rule function.

Unknown Reference	Quick Fix Editor

Qualified name. Examples:
`Rules.SomeName();`
`Concepts.SomeName s;`
	A Quick Fix option appears so you can automatically create the entity or rule function in the given folder.
	The folder must already exist in the project.

Qualifier is a concept or event and reference is unknown. Example:
`Concepts.Person person;`
`person.someProperty = "123"`
	A Quick Fix option appears with the name: Create a property definition in the entity'*`entityName`*'. Click the option to open the New Property Definition dialog, and configure the details of the property.
	In the example, the `someProperty` property does not already exist as a property of `Concepts.Person`.

The following table explains the arguments.

Argument	Notes
Object *`txns`*	You can turn this argument into an `Object[]` within the rule function, for example: `Object[] array = txns;` The resulting array can be used to obtain useful data: • `array[0]` is a `Concept[]` containing all the Concept objects that: ◦ have been created in the current RTC, ◦ and have not been deleted. • `array[1]` is a `Concept[]` containing all the Concept objects that: ◦ existed before the current RTC, ◦ have been modified in the current RTC, ◦ and have not been deleted.

Argument	Notes
	• `array[2]` is a `Long[]` containing the id of all the Concept objects that: ◦ existed before the current RTC, ◦ and have been deleted in the current RTC. • `array[3]` is a `SimpleEvent[]` containing all the SimpleEvent objects that: ◦ are going to be added to the cache. • `array[4]` is a `Long[]` containing the id of all the Event objects that: ◦ existed before the current RTC, or arrived and started the RTC, ◦ and have been deleted in the current RTC.
int *`errorType`*	Value can be one of: `-1` for errors happening during database operations. `-2` for errors happening when sending an event.
int *`errCode`*	The error code, which is dependent on `errorType`: • For database related errors: ◦ The error code as returned by `SQLException`, if available ◦ Else `-100` if TIBCO BusinessEvents determines that the database is not available ◦ Else `-200`, which means that this is an unknown error • For sent event errors, the value is `-1`, meaning an internal error condition.
String *`errMsg`*	The associated exception message.
long *`retryCount`*	The number of times this transaction has been retried before the present call to the callback.

Event Preprocessors

Event preprocessors are rule functions with one argument of type simple event.

Event preprocessors are not used for time or advisory events, and they are multithreaded.

An event preprocessor is assigned to a destination and acts on all events arriving at that destination. Event preprocessors perform tasks after an incoming message arrives at the destination and is transformed into a simple event, but before it is asserted into the Rete network (if it is — events can be consumed in the event preprocessor).

Note:

- If you are using Cache Only mode, take care when designing rules that execute as a result of a time event. For example, a rule that has a join condition using a time event and a concept would not execute if the concept is not loaded in the Rete network and so should not be used in an event preprocessor if the concept is configured as Cache Only mode.
- Events consumed in a preprocessor are acknowledged immediately (and not after the post RTC phase).

You can aggregate events, edit events, and perform other kinds of event enrichment in a preprocessor. You can also use preprocessors as explained below.

You must set locks in the preprocessor when concurrency features are used to protect concept instances during RTC. Locking ensures that updates to concept instances during an RTC do not conflict with another set of updates to the same concept instances in another RTC. Locks are released at the end of the RTC.

If you are using the Cache Only mode for any entities, you must also load the relevant entities from the cache using an event preprocessor.

You can also use preprocessors to improve performance by avoiding unnecessary RTCs in the inference engine. For example you can consume events that are not needed. Another way to use the preprocessor for efficient processing is to transfer an event’s contents to a new concept that is not processed by the agent's set of locally active rules. Such a concept is automatically asserted, and does not trigger rules. It is saved into the cache (depending on OM configuration) where it is available for processing by any agent as needed.

Transaction Error Handler Rule Function

You can create a transaction error handler Callback rule function that enables you to identify which post RTC transactions failed or which events were not sent out during the post RTC phase. The transaction error handler rule function is invoked each time a database transaction exception occurs, and each time a send event exception occurs.

The behavior of the transaction error handler is governed by running the rule function in a separate thread. The rule function is run in the separate thread in the following conditions:

1. The `Agent.`*`agentclassname`*`.enableParallelOps` property is set `true`. See *TIBCO BusinessEvents Configuration Guide*.
2. The `Agent.`*`agentclassname`*`.enableParallelOps` property is set `false` but the **Concurrent RTC** settings are enabled. See *TIBCO BusinessEvents Configuration Guide*.

The error handler rule function provides full details of the failed transactions. The transaction details provided in this Callback function can be used for audit trail purposes. The engine continues to retry the transaction or action as usual.

To Register the Rule Function

To register the rule function, add the following property to the be-engine.tra / project CDD file at an appropriate level (for example, at the cluster level to affect all deployed engines). Set the value to the project path of the error handler rule function:

```
be.engine.txn.error.function=projectPathToErrorHandlerRuleFunction
```

Rule Function Signature

The required signature for the error handler rule function is:

```
int Name (Object txns, int errorType, int errCode, String errMsg, long retryCount)
```

The return value must be 0 (zero).

The arguments are explained in Transaction Error Handler Rule Function Reference.

Transaction Error Handler Rule Function Reference

This table contains references for the Transaction Error Handler rule function.

Argument	Notes
Object *txns*	You can turn this argument into an `Object[]` within the rule function, for example: `Object[] array = txns;` The resulting array can be used to obtain useful data: • `array[0]` is a `Concept[]` containing all the Concept objects that: ◦ have been created in the current RTC, ◦ and have not been deleted. • `array[1]` is a `Concept[]` containing all the Concept objects that: ◦ existed before the current RTC, ◦ have been modified in the current RTC, ◦ and have not been deleted. • `array[2]` is a `Long[]` containing the id of all the Concept objects that: ◦ existed before the current RTC, ◦ and have been deleted in the current RTC. • `array[3]` is a `SimpleEvent[]` containing all the SimpleEvent objects that: ◦ are going to be added to the cache. • `array[4]` is a `Long[]` containing the id of all the Event objects that: ◦ existed before the current RTC, or arrived and started the RTC, ◦ and have been deleted in the current RTC.
int *errorType*	Value can be one of: `-1` for errors happening during database operations.

Argument	Notes
	`-2` for errors happening when sending an event.
int *errCode*	The error code, which is dependent on `errorType`: • For database related errors: ◦ The error code as returned by `SQLException`, if available ◦ Else `-100` if TIBCO BusinessEvents determines that the database is not available ◦ Else `-200`, which means that this is an unknown error • For sent event errors, the value is `-1`, meaning an internal error condition.
String *errMsg*	The associated exception message.
long *retryCount*	The number of times this transaction has been retried before the present call to the callback. Handling Post-RTC action errors: When Post-RTC actions fail, such as acknowledging or sending a message, action handlers try only 10 times with `500` milliseconds sleep. The following parameters can be configured: • `be.engine.txn.action.retrycount`, default is `10`. • `be.engine.txn.action.sleeptime`, default is `500` milliseconds. Handling Post-RTC database transaction errors: In the case of a database exception, the number of tries and waits between each try are handled by the following configuration parameters: • `be.engine.txn.database.retrycount`, default is `maximum integer`. • `be.engine.txn.database.sleeptime`, default is `5000` milliseconds. After the number of tries are exhausted, the transaction is canceled. You can also perform additional actions by registering an error handler rule function as before:

Argument	Notes
	See `be.txn.error function=projectPathToErrorHandlerRuleFunction`. **Note:** When the number of tries are exhausted and the transactions are canceled, it is likely that the cache and database are in an inconsistent state. Practice care when setting a limit to the number of retries. Logs should also be monitored for database related exceptions and immediate action should be taken as soon as possible. It is also advised to disable parallel operations if a limit is set to the number of retries. In this case, multiple transactions are committed to the database together and a single database failure can affect multiple of them.

Rule Template and Rule Template View

A rule template is a specialized type of rule, and a rule template view puts a user-friendly interface around the rule template for use in Web Studio.

Rule templates and rule template views enable non-technical Web Studio users to define executable rules (called *business rules*) within limits defined in the rule template.

Adding a Rule Template View

First, add a rule template. One rule template can be associated with multiple views. .

Procedure

1. Right-click the folder where you want to store the rule template view, and select **New > Other > TIBCO BusinessEvents > Rule Template > View**. You see the New Rule Template View Wizard.
2. In the **Rule Template** field, click **Browse** and select the rule template whose bindings you want to use in this view.
3. In the **Rule Template View Name** field, type a name for the rule template view. In the **Description** field, type a description.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename**. See Element Refactoring Operations for more details.

4. Click **Finish**. You see the Rule Template View editor.
5. Add or edit the description.
6. As needed, browse and select a different rule template

7. As needed, click the Rule Template link to edit the selected rule template, for example to add new bindings.

   Bindings defined in the rule template appear in the Bindings section of the **Presentation** section.

8. In the **Presentation** tab, use plain text or HTML to define an input form. Drag and drop the bindings into the Presentation section, or type them.

   The syntax for a binding is as follows (the closing tag is required):

   ```
   <binding id="bindingName"></binding>
   ```

   You can apply styling and JavaScript to these HTML view bindings. For more details about this, see Rule Template Views.

   You can click the **Preview** tab to see how the view appears in WebStudio.

   Note: Currently, when you click on the **Preview** tab, the preview of the CSS and JavaScript styling is not displayed.

9. Click the **HTML** tab and click **Browse** to select an HTML file, which can be used for the input form.

   Use the binding ID in the HTML file for creating the text box or drop-down for the binding. The syntax for the binding is same as specified earlier step. You can apply styling and JavaScript to these HTML view bindings.

   Note: Currently, when you click on the Preview tab, the preview of the HTML file is not displayed.

10. Save the resource.

The Rule Template Editor

Comparing similar sections of the Rule Template editor and the Rule editor highlights the unique features of rule templates.

Configuration Section

The Configuration section is the same as the Configuration section in a rule. Priority, ranking, and forward chaining features work the same at runtime whether a rule is created using the Rule Editor or the Rule Template Editor.

Declarations and Variables Section

The Declarations and Variables section (`declare` in the source view) is similar to the Declaration section of a rule. However, in rule templates you can declare primitive types (as you can in rule functions). You can also define initial expressions for primitive types, for example, `int j = 50;`.

Pre-conditions Section

The Pre-conditions section (`when` in the source view) is similar to the Conditions section of a rule. However, WebStudio users can add additional conditions when defining individual business rules. The Pre-conditions section specifies conditions that must be met in *all* instances of a rule template before the rule's actions execute.

This feature enables complex calculations to be done before the definition of the condition. The Pre-conditions section could also contain conditions that might appear in any rule, such as the following:

```
con.prop1 > con2.prop1;
```

Action Context Section

The Action Context section (`actionContext` in the source view) has a similar purpose to the Actions section of a rule.

The Action Context section (`actionContext` in the source view) has a similar purpose to the Actions section of a rule. The Action Context section, however, defines *all possible actions* that can be taken by a business rule (after all conditions are met). Only the action context statements that the WebStudio user selects and defines as commands in the business rules are actually taken (depending on rule evaluation at runtime).

Defining the superset of all possible actions simplifies and restricts the configuration of business rules in WebStudio.

The actions are not completely defined in the rule template. Completing the definition of an action is a WebStudio user task. If bindings are used (and a view) then in WebStudio, the business rule writer has to enter only the binding values to complete the definition.

Action context statements are of three types: create, modify, and call, plus arbitrary actions, as explained next.

Create Statements

Create statements enable business rule developers to create new concepts or events. Create statement uses fully qualified name for the concepts and events instead of simple name. For example:

```
create Concepts.Bulk bulk;
```

In the rule template, that is as far as you go. It’s up to the business rule developer to assign the specific constructor values for the entity type.

Modify Statements

Modify statements enable business rule developers to modify concepts or events. For example:

```
modify bulk;
```

Only entities in the scope of the rule can be used. As with create statements, it is up to the business rule developer to determine which properties to change and assign values to those properties.

Call Statements

Call statements enable business rule developers to call rule functions. Call statement uses fully qualified name for the concepts and events instead of simple name. Call statement uses a variable to store the return value. Call statement uses this return variable even if the return type is void. For example:

```
call RuleFunctions.matchFound matchfound;
```

The business rule developer assigns the parameters for the rule function.

Arbitrary Actions

Rule template developers can add arbitrary actions as well as create, modify, and call statements. For example:

```
create Concepts.Bulk bulk;
if (mybinding > 10) {
   MyRFs.myRF(otherbinding);
   mycon.prop1 = thirdbinding;
}
```

Bindings Tab

The form editor for rule templates provides an additional tab, the Bindings tab, with two sections that are not found in the Rule editor: Bindings and Views (`bindings` and `views` in the source view).

Bindings

A binding is used like a local variable whose value is assigned by WebStudio users when defining business rules.

When defining a rule template, you can use bindings in condition checks and in action statements. For example, suppose you define a binding called `dollarAmount`. You could then define a condition as follows:

```
output.UNIT_PRICE > dollarAmount;
```

At runtime, the `UNIT_PRICE` is checked against the specific dollar amount defined in the business rule.

Similarly, you could use the binding in an action statement as follows:

```
RuleFunctions.matchFound(match,null,dollarAmount);
```

When executed, the engine calls the `matchFound` rule function and passes the value of `dollarAmount` as defined in the business rule.

Bindings are primitive types. You can optionally provide an initial value, and the value can be defined using a domain model. You can specify a domain model by clicking in the Domain Model cell, and then clicking the **Browse** icon. If a domain model is specified for a binding, the view defined for a rule template displays the input field for the binding as a drop-down box. In the following example, the String binding `day` is initially assigned to the value of `Monday`, and it is tied to the domain model `RuleFunctions.WeekDays`.

```
String day = "Monday" (RuleFunctions.WeekDays);
```

Views

In the Views section of the Bindings tab, you select a view that you have already defined. The selected view is used in WebStudio to present the rule template to the users and enable them to define the business rules (that is, executable rules).

If you do not associate a view with a rule template, a builder-based interface is used.

Rule Template Editor Reference

The rule template, rule, and rule function editors are similar. This section focuses on the form-based editor. You can adapt the information to apply to the different blocks of code in the source editor.

Property	Description
Configuration Section	
Same as the Rule Editor Configuration section. See Rule Editor Reference for details.	
Declarations and Variables Section	
Type	Types you add define the scope. As in the Term section of a rule, you can specify concept or event types in the project that you will use in your rule. In addition you can specify variables using primitive types. For example (in the source view):

Property	Description

	`Concepts.Accounts.CheckAccount checkacc;`
	`int i = 123;`
	
	It is not necessary to add scorecards in the declaration in order to use them in the rule or rule template.

Alias	A name used to refer to the scope element or variable in the Pre-conditions (`when`) and Action Context (`actionContext`) sections of the rule. Aliases must be valid identifiers See Identifier Naming Requirements.
Expression	If you specify a primitive you can also specify an expression.
Pre-conditions Section	
	Each line in the Pre-conditions area is a single expression. Expressions can be local variable declarations, method calls and so on, as well as expressions that evaluates to `true` or `false` such as are found in regular rules. Each line is joined to the others with an implicit `and` an operator. For the OR operator, use a double pipe (||) on the same line. TIBCO BusinessEvents evaluates single conditions from left to right. TIBCO BusinessEvents optimizes the evaluation of multiple conditions (refer to the "Order of Evaluation of Rule Conditions" section in *TIBCO BusinessEvents Architect Guide*.)
Action Context Section	
	Defines the list of all possible actions that can be executed when a rule instance based on this template is fired. Only the subset of these action context statements that are defined as commands in the business rules are actually considered. When you add an action context statement, first select one of the action types, then select an entity (create or modify action types) or rule function (call action type).

Rule Template Views

A rule template view defines a visual presentation of the rule template to make it easy for the WebStudio user to define a business rule.

The Rule template view has three tabs **Presentation**, **HTML File**, and **Preview** to help you create an HTML form for the business rule.

Presentation

The View uses rule template bindings to define an easy-to-use form-like interface where the WebStudio user assigns values to the bindings, for example, threshold values. You can use plain text or HTML to define an input form. You can drag and drop the bindings into the Presentation section, or type them. The syntax for a binding is as follows (the closing tag is required):

```
<binding id="bindingName"></binding>
```

For example:

```
Require that all applicants have a minimum income of <binding
id="minimumIncome"></binding>, a minimum age of <binding
id="minimumAge"></binding>, and restrict the credit type to <binding
id="creditType"></binding>
```

If the binding is associated with a domain model, the WebStudio interface displays the input widget as a drop-down list. Otherwise, an input widget of the appropriate type is displayed, for example, a checkbox for a boolean field.

You can also apply the styling to the input form by using the CSS class, JavaScript, or inline CSS.

CSS Class Styling

To apply the styling to the HTML elements corresponding to bindings, you can create a CSS class by using the following syntax:

```
RuleTemplateName_bindingID
```

For example, for the `Applicant_PreScreen` rule template and the `minimumAge` bindingID, the corresponding class name of HTML element is `Applicant_PreScreen_minimumAge`.

```
.Applicant_PreScreen_minimumAge
{
background: rgba(255,255,255,0.1);
border: none;
font-size: 16px;
}
```

Inline CSS

You can add the style attribute with HTML styling options to the binding tags. This applies the specified styling to the binding. For example:

```
<binding id="minimumIncome", style="height:100px; text-
align:center"></binding>
```

JavaScript

You can also apply styles to elements using JavaScript. JavaScript uses the BindingID to form a ID which the script uses to link style to the binding. In the script, apply the style to a particular item using the following syntax:

```
%RTIName%_bindingID
```

where, `%RTIName%` is replaced with the business rule name in the run time

For example, to apply blue color to `minimumAge` binding, you can use the following code in the script:

```
document.getElementById("%RTIName%_minimumAge").style.color ="red";
```

The following sample code displays all three method of styling applied to the presentation text:

```
<head>
<style>
.Applicant_PreScreen_minimumAge{
       height : 100px;
       text-align : center;
}
</style>
<script>
```

```
try {
    document.getElementById("%RTIName%_minimumIncome").style.color=
"blue"
       document.getElementById("%RTIName%_minimumAge").style.color ="red";
}
catch(e){
       console.log(e);
}
</script>
</head>
<body>
Require that all applicants have a minimum income of <binding
id="minimumIncome" ,style="height:100px; text-align:center"></binding>,
a minimum age of <binding id="minimumAge"></binding>, and restrict the
credit type to <binding id="creditType"></binding>, testing area for
string <binding id="testingTextArea"></binding>, testing checkbox
<binding id="testBoolean"></binding> and testing date <binding
id="testDate"></binding> .
</body>"/>
```

HTML File

You can also use an HTML file for business rule form. In the HTML file you can use the HTML styling for bindings using the same syntax as used on the **Presentation** tab.

Preview

Show preview of the business rule form created in the **Presentation** tab. The **Preview** tab shows preview of only plain text form and not for HTML styling. Thus, you cannot view the preview for the uploaded HTML file or applied CSS and JavaScript.

Functions

TIBCO BusinessEvents provides various built-in function to perform the standard tasks; however you can also create your custom function to perform the task as per your requirement.

Note: TIBCO BusinessWorks catalog functions are not supported in TIBCO BusinessEvents.

Catalog Functions

The functions registry includes various catalogs of functions provided with the product, and each catalog organizes its functions into categories. You can use functions in rule conditions and actions and in rule function bodies.

Some functions work together. For example, the Standard catalog function `String.append` requires that you first use the function `String.createBuffer`, to create a buffer to which strings can be appended.

To View the Catalog functions

All catalogs appear in the Catalog Functions view. To open the view navigate to **Window > Show View > Other > TIBCO BusinessEvents** and select Catalog Functions. The catalog view appears on the right, by default.

To View Documentation for Functions

Documentation for functions is provided in tooltips you can access while using the Catalog Functions view. Hover the mouse over the function name to see the tooltips.

You can also access this documentation using the online functions reference, available in the HTML version of the documentation, but not in PDF. To access the HTML version of documentation, visit https://docs.tibco.com/products/tibco-businessevents and open the **Web Help**.

Expand the contents panel to **Online References > Online References**, and on the right select the *TIBCO BusinessEvents Functions Reference* link.

Built-in Functions

For all the built-in functions, this section lists the main categories in each function catalog (but not sub-categories).

See *TIBCO BusinessEvents Functions Reference* for full details.

Standard Functions

The standard functions catalog contains the most used functions. The standard function catalog includes the following categories:

Channel

These functions return information about destinations, and can resume and suspend a destination.

Cluster

Cluster functions help with multi-engine functionality.

DataGrid

These functions are for use with Cache object management. See Cache Related Functions.

Collections

Collections functions allow you to deal with various collections structures. The functions names are:

- Comparator
- Iterator
- List
- Map
- Set
- SortedMap

- SortedSet

Date

Date functions allow you to compare two DateTime values using only the date portion of the value.

DateTime

DateTime functions allow you to perform the date/time-related tasks and more: add units of time to DateTime, compare, retrieve, and format dates and times.

Engine

Engine functions allow you to retrieve information about the engine, for example, available memory or the number of rules fired. The functions names are:

- Locale
- Profiler
- Rtc
- Variable

Event

Event functions allow you to assert, create, and send simple events and perform other event-related tasks, for example, return the default destination URI of a simple event.

Exception

Exception functions enable you to create exceptions.

File

File functions provide various useful functions used when working with files.

HTTP

HTTP functions are used with the HTTP channel. The Functions that belong to this group are named

- Servlet functions

Instance

Instance functions allow you to create and delete concept instances and perform other instance-related tasks, for example, return an instance given an internal ID. The functions belonging to this group are named:

- PropertyArray
- PropertyAtom
- Reflection
- StateMachine

JSON

JSON functions are used to parse JSON content and perform various operations on JSON node.

Log

Log functions allow you to write statements in logs.

Math

Math functions allow you to perform advanced mathematical operations.

Metric Function

Creates a new Metric instance based on the data provided in the XSLT Mapper and adds it to the working memory. Adding the instance to the working memory causes any rule conditions that depend on the concept to be evaluated.

Number

Number functions allow you to perform type conversions from and to numbers and return the maximum and minimum values for a numeric type.

SOAP

SOAP functions enable you to work with SOAP messages sent through an HTTP channel.

String

String functions allow you to perform comparisons, searches, conversions, and other operations with strings.

System

System functions allow you to send messages to a debug log, retrieve global variables, retrieve system properties, and write data to a file. The functions that belong to this group are named:

- ID functions
- IO functions, which allow the writing and closing of specific files.

Temporal

Temporal functions allow you to examine and perform calculations on values stored in a property's history. For information about using temporal functions. The functions that belong to this group are named:

- Calculus
- History
- Numeric
- Statistic

Util

Util functions category has one sub-category for working with HashMaps.

VRF

VRF functions (that is, Virtual Rule Function functions) allow you to work with decision tables. See *TIBCO BusinessEvents® Decision Manager User Guide* for details.

XPath

XPath functions allow you to evaluate XPath expressions.

Legacy ActiveSpaces Functions

These functions allow you to interact with Legacy ActiveSpaces metaspaces. They are used with the Legacy ActiveSpaces channels to perform operations on the Legacy ActiveSpaces metaspace and spaces connected by the channel.

CEP Load Balancer Functions

Load balancer functions are used to configure and work with the load balancer. There are two types of load balancer functions:

- Receiver functions, which allow you to create or discard a receiver, and retrieve the receiver's membership details.
- Router functions, which allow you to create or discard a router, and send an event to a remote receiver.

CEP Pattern Functions

Pattern functions are used with the pattern matcher language for identifying patterns in events. See *TIBCO BusinessEvents® Event Stream Processing Pattern Matcher Developer Guide* for details.

CEP Query Functions

Query functions are used with the query language for querying data in the cache. See *TIBCO BusinessEvents® Event Stream Processing Query Developer Guide* for details.

CEP Store Functions

You can use the CEP Store catalog functions to interact with TIBCO ActiveSpaces data store and external Apache Ignite cache.

The basic CEP store functions are as follows:

- `connect` or `disconnect` to data store
- `open` or `close` a container
- `update`, `put`, `get`, `delete`, or `query` items from the container
- `putAll`, `getAll`, and `deleteAll` to work on items from all containers
- `closeQuery` to exit a query explicitly

Apart from the basic functions, other functions are grouped into the following categories:

- **Metadata** - Functions to get details of store, container, and fields.
- **ConnectionInfo** - Functions to set up connection details for data store and cache with a provision to write customized configuration functions.
- **Container** - Functions to set up connection details for data store and create, configure and interact with external cache such as Apache Ignite.

  The container functions `lock` and `unlock` are supported only for Apache Ignite cache.
- **Item** - Functions to set or get data from store items.
- **QueryOptions** - Functions to cover prefetch size, snapshot consistencies, reuse for ActiveSpaces store type and functions to cover execution of query on local node, colocated query, query timeout for external Apache Ignite.

- **Transactions** - Functions to enable or disable the transactional behavior. Additionally, if the transactional behavior is enabled, commit or rollback transaction.
- **Util** - Functions to check client version, edit log configurations, and set up a custom logger.

For more information on these catalog functions, see **TIBCO BusinessEvents > CEP Store > Store** in *TIBCO BusinessEvents Functions Reference*.

Note: Depending on the table creation in Apache Ignite you must use the specific catalog functions to load and update the rows in the external Ignite table:

- If external table is created by using an SQL statement (JDBC tool) then use the catalog function `Store.executeUpdate()` to insert or update rows with an SQL statement.
- If the table is created by using the catalog function `Store.Container.Ignite.createContainer()` then create or update the table data with catalog function `Store.put()`.

Hawk Functions

Functions to interact with Hawk micro-agents methods.

DBMS Functions

Database functions are provided for working with database concepts. See TIBCO BusinessEvents Data Modeling for more on database concepts.

Security Functions

These functions are used internally by the TIBCO BusinessEvents WebStudio, for authentication.

Studio Functions

These functions allow you to use the Studio Util functions to build classes and EAR files.

Custom Functions

Using TIBCO BusinessEvents you can define custom functions and expose them as catalog functions or mapper functions. By exporting the project containing the custom Java functions to a project library, the custom Java functions can be used in other projects.

To use custom functions in the mapper, set the property `com.tibco.datamodel.xml` to `true` in the CDD file. This property must be set to true to force XPath or XSLT to use XML rather than the new JSON native data model. If your incoming or outgoing messages are exclusively XML based, setting this property to true avoids conversions from a separate internal data model to XML and use XML natively.

When the `com.tibco.datamodel.xml` property is true, set the property `be.serialization.notation=standard` to print numbers in decimal or standard format. For example, to print number 3.0.
But if the property `be.serialization.notation` is not set or `be.serialization.notation=scientific` (default value), then the numbers are serialized in scientific format. For example, to print 34.45E1.

Adding Custom Java Functions

Using TIBCO BusinessEvents, you can define custom functions and expose them as catalog functions. By exporting the project containing the custom Java catalog functions to a project library, the custom Java functions can be used in other projects.

Procedure

1. Define a simple Java class under the `JavaSrc` folder. For example, `MyFunction.java`.
2. To expose Java methods as functions, declare them as public static variables:

```
public static int sum(int a, int b)
{
       int x = a+b;
       return x;
}
```

3. The annotation @BE Package is a class level annotation: add the class-level annotation @BEPackage before any classes are declared.

   ```
   @BEPackage(catalog = "arithmeticOperations", category =
   "arithmetic.operations", synopsis = "Performs few arithemtic
   operations")
   ```

4. The annotation @BEFunction defines the synopsis, signature, parameters, return types, descriptions, examples and other relevant information about the function. The information provided in the annotation is displayed in the tooltip for the catalog function.

   ```
   @BEFunction(name = "SummationTest",
                          description="Adds two given numbers",
                          signature="int sum(int a, int b)",
                                 freturn =
   @com.tibco.be.model.functions.FunctionParamDescriptor(name = "",
   type = "int", desc = "Returns the summation of a and b."),
                          params = {
   @com.tibco.be.model.functions.FunctionParamDescriptor(name = "a",
   type = "int", desc = "First arg"),
   @com.tibco.be.model.functions.FunctionParamDescriptor(name = "b",
   type = "int", desc = "Arg two")
                          },
                          fndomain = {ACTION, CONDITION, QUERY, BUI},
   version="1.0",see="",cautions="none", example=" ")
   ```

5. Save the file.

Result

Once you add and save the annotations, the custom Java functions are available in the Catalog Functions view.

Adding Libraries

TIBCO BusinessEvents core libraries are automatically added to the Java Build Path.

If the project library contains Java source files, then those Java source files must be added to the Java build path source section. This ensures that the Java classes present in the project library are accessible from the Java source file of the current project.

Importing Old Projects

When older projects are imported to a latest environment, the project nature is updated to include the Java nature. The `.project` file contains the following entries:

```
         <natures>
                <nature>com.tibco.cep.studio.core.studioNature</nature>
                <nature>org.eclipse.jdt.core.javanature</nature>
        </natures>
```

Compiling Project and Building an EAR File

There is no change in the process to compile and build an EAR file.

Click **Project > Build Enterprise Archive** from the menu to build an EAR file.

Debugging

While debugging the projects, you can use Java breakpoints to step into the custom functions code. Java debug breakpoints and rulefunction breakpoints work seamlessly, allowing you to step into the code after hitting a breakpoint and inspect the values of the variables as they change.

Annotations Reference

Reference for @BEPackage

Field	Description
enabled	This annotation allows the user to specify if an associated annotation is enabled or disabled for use.
catalog	Name of the catalog where the custom function resides.
category	Category for the custom functions defined in the class.

Field	Description
synopsis	Brief description for the group of custom functions defined in the specified directory.

Reference for @BEFunction

Field	Description
enabled	This annotation allows the user to specify if an associated annotation is enabled or disabled for use.
deprecated	This annotation element identifies if its associated method is deprecated or not.
name	Name of the custom function.
synopsis	Brief description of the custom function.
signature	Signature for the custom function.
params	List of parameters for the custom function.
freturn	Return type of the custom function.
version	The TIBCO BusinessEvents version from which a given catalog function became public API.
see	Refers to the other custom functions or java doc URLs.
mapper	Used for functions that involve invoking a mapper UI on **Ctrl-Click**.
description	Detailed description of the custom function.
async	A boolean value indicating if the use of this function causes the rules to be evaluated again.
reevaluate	A boolean value indicating if the use of this function causes the rules to be evaluated again.

Field	Description
cautions	List any cautions that may be applicable when using the custom function.
domain	The specified domain, from which Java methods are accessible. Examples of domain are `ACTION`, `CONDITION`, `QUERY`, `BUI`.
fndomain	Describes an enumerated function execution domain.
example	Example of the custom function.

Considerations when Defining the Functions

Java methods defining the functions can have primitive data types, or Concept, Event and Object. They can return any primitive data types or Concept, Event and Object.

If the custom functions have primitive data types only, it is also visible and usable in the mapper. In that case, the category attributes of the @BEPackage annotation is used as the folder for your custom functions.

Note: The interface has some limitations as it does not allow static methods. Do not use annotations when using interfaces. Create an interface as a normal java interface without any annotations and implement it in a class without any annotations. To see these interface methods under the custom catalog function, create a static overloading method with annotations.

For example:

```
public interface InterfaceProjectLibTest
{
           public abstract String[] getAllDestinations();

                  public abstract String getClusterName();
}

public class interfImpl implements InterfaceProjectLibTest
{
           static int count=0;
           @Override
           public String[] getAllDestinations()
           {
                       Object[] ars= null;
       RuleSession rs = RuleSessionManager.getCurrentRuleSession();
String[] activdest=
(String[])rs.invokeCatalog("Channel.getAllDestinations", ars);
                       count = activdest.length;
                       return activdest;
           }

           @Override
           public String getClusterName() {
                      // TODO Auto-generated method stub
                      Object[] ars= null;
RuleSession rs = RuleSessionManager.getCurrentRuleSession();
                      String clustername;

clustername=(rs.invokeCatalog("Cluster.getClusterName", ars)).toString
();
                      return clustername;
           }

           @BEFunction(name = "getNoOfDests", signature= "getNoOfDests",
description="Returns Number of destinations of the current rulesession")
           //@BEFunction
           public static int getNoOfDests()
           {
                      interfImpl im = new interfImpl();
                      im.getAllDestinations();
                  +    System.out.println("count: " + count);
                      return count;
           }

           @BEFunction(name = "ReturnAllDestinations", signature=
"getAllDest",
description="Returns all destinations of the current rulesession")
           public static String[] getAllDest()
           {
                      interfImpl im = new interfImpl();
                      String[] activedests = im.getAllDestinations();

                      return activedests;

           }

           @BEFunction(name = "ReturnClusterName", signature=
"getClusName()", description="Returns the clustername")
       public static String getClusName()
```

```
            {
                        interfImpl im = new interfImpl();
                        String clustername=im.getClusterName();
                        System.out.println("ClusterName is: " + clustername);
                        return clustername;
            }
}
```

Ontology Functions

Ontology functions are generated by TIBCO BusinessEvents based on the concepts, events, and rules in your project.

There are three types of ontology functions:

Constructors

They allow you to create a simple event or concept instance.

Time events

They allow you to create and schedule a time event. See Working With Time Events.

Rule functions

They allow you to invoke a rule function.

The Ontology Functions area uses the same folder structure as the project (or rather, a subset of that structure).

Various ontology functions appear depending on the project and the add-on products installed:

- The ontology functions catalog lists all the entity types in a project. Each type has a function for creating an instance of that type.
- Each rule function also has an ontology function enabling it to be invoked in rules.

Enabling Extended Functions

Extended functions (sometimes called hidden functions) may be made available by TIBCO Support to address customer-specific use cases. They are also sometimes used to make legacy features available to customers who wish to continue using them. To make them visible in the Catalog Functions view, perform the following actions.

Procedure

1. Open the following file for editing:

   ```
   BE_HOME/studio/eclipse/configuration/studio.tra
   ```

2. Add the appropriate property, using the specified name. For example, the following formats are typically used:

   ```
   TIBCO.BE.function.catalog.function-catalog-name=true
   TIBCO.CEP.modules.function.catalog.function-catalog-name=true
   ```

3. Save the file and restart TIBCO BusinessEvents Studio.

Function Tooltips and Decorations

When you float the cursor over a function in the registry, TIBCO BusinessEvents displays the description and syntax in a tooltip next to the cursor. Functions that can be used with various product features are decorated with small symbols that indicate useful information about use of the function.

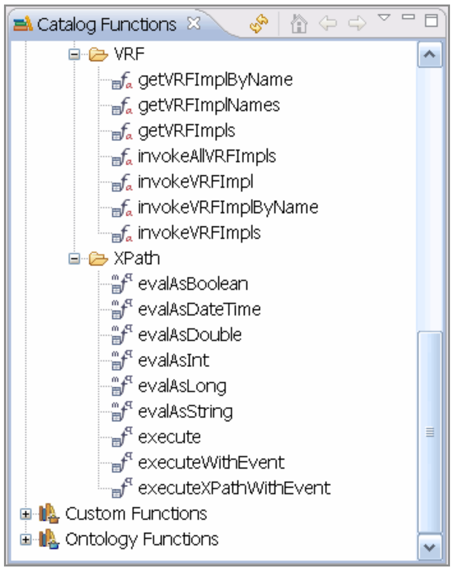
Tooltips

Tooltips also form the online reference to the function catalogs. See *TIBCO BusinessEvents Functions Reference*.

Tip: You can create your own tooltips for custom functions. You can also turn off the tooltip display too using **Window > Preferences > TIBCO BusinessEvents preferences**.

Decorations Indicating Where Functions can be Called

Some functions have limited application, and some can be used with various product features such as in queries. The second ones have decorations that cluster around the "f" next to the function name in the function catalog. A function can have zero, one, or more decorations. The following figure shows all the available decorations. They are described in the sections after the figure:


Catalog Functions

- VRF
  - getVRFImplByName
  - getVRFImplNames
  - getVRFImpls
  - invokeAllVRFImpls
  - invokeVRFImpl
  - invokeVRFImplByName
  - invokeVRFImpls
- XPath
  - evalAsBoolean
  - evalAsDateTime
  - evalAsDouble
  - evalAsInt
  - evalAsLong
  - evalAsString
  - execute
  - executeWithEvent
  - executeXPathWithEvent
- Custom Functions
- Ontology Functions

Action-Only Functions

Some of these functions have side effects, for example they can change values. Other functions are limited to actions for other reasons. These action-only functions are identified by a small `a` at the bottom right of the icon . For example: setDateTime.

Functions that can be used both in actions and in conditions have no decoration. They are considered to have the default validity.

Mapper Functions

Functions that bring up the XSLT mapper and XPath builder are identified by a small `m` at the upper left of the `i` , for example: evalAsString.

Functions That Can Be Used in TIBCO BusinessEvents Decision Manager

Functions that can be used in TIBCO BusinessEvents Decision Manager are marked with a small table icon, getVRFImpls for example (which is also an action-only function).

Functions That Can Be Used in Queries or with Pattern Matcher

Functions that can be used in queries or with Pattern Matcher are marked with a blue *q* for example, nanoTime. You can call such functions in a query string and in Pattern Matcher.

They are also valid in rule actions and conditions. See *TIBCO BusinessEvents® Event Stream Processing Pattern Matcher Developer's Guide* and *TIBCO BusinessEvents® Event Stream Processing Query Developer Guide* for details about Pattern Matcher and query features.

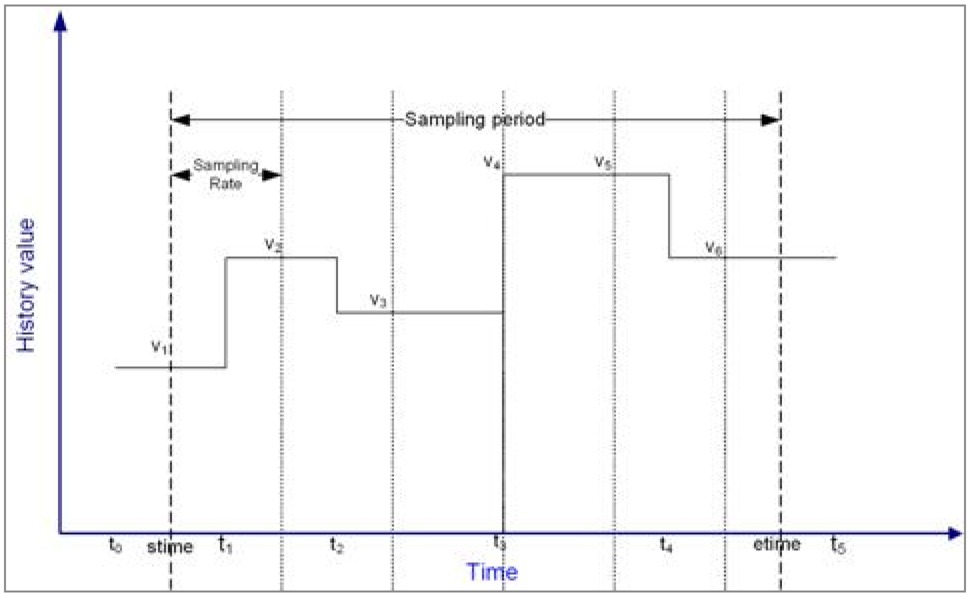
Temporal Functions and Their Parameters

The set of Standard functions that come with TIBCO BusinessEvents includes functions that allow you to perform calculations on numeric values sampled over time.

These functions are called *temporal* functions and they work exclusively with concept properties that store numeric values. Temporal functions make use of the history ring buffer to sample a property's values over time.

Note: Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.

Figure 10: Temporal Functions Parameters



All temporal functions include these parameters, illustrated in Temporal Functions and Their Parameters:

property

The property for which you want to sample values.

stime

The time from which you want to begin sampling values (the start time) measured in milliseconds since 00:00:00 UTC on January 1, 1970.

etime

The time at which you want to stop sampling values (the end time) measured in milliseconds since 00:00:00 UTC on January 1, 1970.

sample_rate

The number of milliseconds between samples.

bound_by_stime

A flag indicating whether the start-time is flexible:

- `True` indicates that if the start time you provide is earlier than the timestamp for the oldest available value, you want to perform the calculation starting with the oldest available value.
- `False` indicates that if the start time you provide is earlier than the timestamp for the oldest available value, you want to abort the calculation.

Virtual Rule Functions and VRF Catalog Functions

The VRF category of functions (within the standard functions) are used only with TIBCO BusinessEvents Decision Manager.

The VRF category of functions provide flexibility when you are working with virtual rule function implementations. Virtual rule functions are implemented by decision tables.

Note: When you deploy multiple implementations (tables) for one virtual rule function, but use a function that doesn't specify the implementation by name, for example if you use `Functions.`*`MyVirtualRuleFunction`*`()`, the default implementation is used. The default implementation is whichever was the last implementation to be deployed. However, if you use hot deployment, it may not be possible to determine which implementation was deployed last.

The VRF category of functions contains the following:

```
getVRFImplByName()
getVRFImplNames()
getVRFImpls()
invokeAllVRFImpls()
invokeVRFImpl()
invokeVRFImplByName()
invokeVRFImpls()
```

Tip: Defining the execution order of multiple decision tables for one VRF: When a VRF has multiple implementations (decision tables), the order in which the decision tables execute can be defined using each decision table's Priority setting, which is set in the decision table Properties tab.

Adding a Virtual Rule Function

To add a virtual function, perform the following steps.

Procedure

1. Right-click the folder where you want to store the virtual rule function and select **New > Rule Function**. You see the New Rule Function Wizard.
   a. In the **Rule Function Name** field, type a name for the rule function.
   b. In the **Description** field, type a description. (In the source editor the description appears in the `* @description` line of the comments at the top of the editor).
   c. Select the **Virtual** checkbox.

      Note: You cannot change a new resource name after you click Finish. (You can change the description, however.)

2. Click **Finish**.

   Click the tabs at the bottom of the editor to switch between the form-based editor and the source editor as you work, depending on your preference. These instructions use the form editor and mention the equivalent settings in the source editor.

3. In the Form editor Configuration section, add or edit a description as desired. (In the source editor the description appears in the `* @description` line of the comments at the top of the editor).

4. If you did not do so in the Wizard, select the Virtual checkbox.

   Tip: In the source editor, the signature of a virtual rule function is:

   ```
      virtual void rulefunction folder.RFName
   Do not add code to the Body block in the source editor of
   a virtual rule function. If you do, you see error messages
   if you try to save or to switch to the form-based editor.
   ```

   In virtual rule functions, the Validity field is set to Action and the Return Type is set to Void. column, select where the rule function can be used (source editor equivalents shown in parentheses:

5. In the Scope section (`scope` statements in the source editor), drag an ontology entity into the Scope area, or perform the following actions:
    a. Click **Add** to add resources that you are using in your rule function. You see the Select Rule Function Scope Arguments dialog.
    b. In the upper half of the Select Rule Function Scope Arguments dialog, select the type you want to use.
    c. If the type you select is an ontology type, in the lower half of the dialog, select a resource from the filtered ontology tree.
    d. Click **OK**.

        Your selection appears in the Declarations list. TIBCO BusinessEvents assigns an alias to it. You can edit the alias.

        Add more entities as needed.
6. Save the project.

Result

For example, here is the source view for a simple virtual rule function:

```
/**
 * @description Action to take when account is suspended
 */
virtual void rulefunction Rules.FollowUp {
  attribute {
    validity = ACTION;
  }
  scope {
    Concepts.Account account;
  }
  body {
  }
}
```

VRF Function Arguments

The VRF functions use various subsets of the following common arguments:

Common Arguments for VRF Functions

Name	Type	Notes
`vrfURI`	String	The universal resource identifier (URI) for the virtual rule function. This is typically the full path to the virtual rule function within the project directory. For example, in the `CreditCardApplication` example, the virtual rule function `Person_VirtualRuleFunction()` has the following URI: `/Virtual_RF/Person_VirtualRuleFunction`
`vrfImpl`	Object	An object representing a virtual rule function implementation. This argument is required when invoking specific virtual rule function implementations.
`implName`	String	The name of a decision table (also known as a virtual rule function implementation). For example, in the `CreditCardApplication` example, the virtual rule function `BankUser_VirtualRuleFunction` has an implementation (decision table) called `bankUser`. The `implName` argument is used to retrieve a corresponding implementation object, or to execute an implementation.
`args`	Object array	The arguments to be passed to one or more virtual rule function implementations on invocation. These objects consist of the concepts, events, scorecards, and so on, that are needed by the implementation or implementations. For example, the `processApplication` implementation in the `CreditCardApplication` example project requires concepts of type `Application`, `BankUser`, and `CreditCardApplication` to be passed as arguments. In order to invoke the `processApplication` implementation, an instance of each concept type must be passed in the `args` array.

Name	Type	Notes
`returnValues`	Object array	*Not used in this release*. This argument is used only for the `invokeVRFImpls` function. When invoking multiple implementations, the return value of each implementation is stored in this array. The array will contain a null entry for each implementation that does not return a value.

Cache Related Functions

Various standard functions in the Standard catalog `Cluster.DataGrid` category enable you to work with objects in the cache.

Cache load functions load items into the Rete network so they are available.

Tool tips in the user interface (and reproduced in the *TIBCO BusinessEvents Functions Reference*) explain the details of how to use the functions. This section explains certain patterns of use.

These core functions are used with both the Oracle and TIBCO cache providers.

Note: The previous versions of these core functions (which began with `C_`) are deprecated. Migrate projects created in earlier releases to use the current names.

Query functions are not available for the Legacy ActiveSpaces provider.

Functions for Loading Entities to Rete from Cache and Backing Store

```
CacheLoadConceptByExtId
CacheLoadConceptsByExtId
CacheLoadConceptById
CacheLoadConceptsById
CacheLoadEventByExtId
CacheLoadEventById()
CacheLoadParent()
```

`CacheLoad*()` functions are required for working with cache only cache mode. They load entities from the cache (or backing store if not found in the cache) into the Rete network. See Loading Cache Only Objects into the Rete Network in *TIBCO BusinessEvents Architect Guide* for details.

Use the `CacheLoad*()` functions in an event preprocessor. Only use them in rules for cases where the `ID` or `ExtId` is not known in advance (in the preprocessor).

After cache-only concepts are loaded in this way, you can then use `Instance.getByExtId()` in rules.

Never use `Instance.getByExtId()` unless you have first loaded the concept. `Instance.getByExtId()` does not assert the concept but just returns it for use in rules, for example, as read-only reference data.

Note: When querying by extid, you must use the functions `Cluster.DataGrid.CacheLoadConceptByExtIdByUri` and `Cluster.DataGrid.CacheLoadEventByExtIdByUri`.

Locking Functions

```
Lock()
UnLock()
```

In the event preprocessor, use the `Lock()` function to prevent other threads or engines from operating on the same entity. The lock is automatically released at the end of the RTC.

Use `Unlock()` only in a preprocessor and only to handle cases where you need to release the lock immediately instead of at the end of the RTC, for example because some information is missing that would be required to go forward.

See Using Locks to Ensure Data Integrity Within and Across Agents in *TIBCO BusinessEvents Architect Guide* for details about use of locks.

Indexing Function

`Index()` creates an index on the specified property, which is useful when you run queries. However, a simpler way is to use Present in Index checkboxes in the CDD.

Specialty Functions Not for General Use

The following functions are used only in certain applications:

```
CacheLoadConceptIndexedByExtId()
CacheLoadEntity()
CacheReevaluate()
EnableCacheUpdate()
```

Indexing for More Efficient Cache Queries

When you use cache OM, you can index concept and event properties to make searches faster. You can index more than one of an entity type’s properties.

You can create the indexes in the following ways.

Creating an Index Using a Domain Object Override Setting

You can create an unordered index in the project’s Cluster Deployment Descriptor (CDD) using a domain object setting.

Procedure

1. Open the project CDD in TIBCO BusinessEvents Studio and go to **Cluster tab > Object Management > Domain Objects > Overrides**.
2. Edit or create an override entry as needed for the desired entity or entities
3. In the override entry’s Properties Metadata section, select the **Present in Index** checkbox for the property you want to index.
4. Save the CDD file.

Enabling Explicit Tuple Format for Legacy ActiveSpaces Cluster Fields

You can set a property in the CDD so that they are instead stored as tuples. That is, TIBCO BusinessEvents attempts to create explicit space structures for each entity type.

In Shared Nothing mode, explicit tuple format is assumed internally. You should not set this property in Shared Nothing mode.

You can still use query language queries (as explained in Query the Cache Using BQL Queries) if entities and their properties are stored as blobs. However, performance is improved if they are stored as tuples.

Only certain datatypes can be stored as tuples:

- Primitive datatypes

  ```
  int, long, boolean, string, datetime, contained, reference
  ```

  Are stored as the equivalent Legacy ActiveSpaces types:

  ```
  integer, long, boolean, string, calendar, long, long
  ```

- Concepts
  - Concept property arrays are stored as blob columns
- Events
  - Payload is a byte array or blob
- State Models

If an error occurs, for example due to an unsupported datatype, the item reverts to its original blob format.

Procedure

You have to set the "`Store Properties As Individual Fields`" field to `true` to enable explicit tuple format.

Query the Cache Using BQL Queries

To run the queries, you must first enable a dynamic query session.

Note:
- The term BQL is an acronym indicating the TIBCO BusinessEvents Event Stream Processing query language.
- This is the only method that works with Legacy ActiveSpaces.

Procedure

1. Add a rule function in the inference agent that contains the following function: `Query.Util.startDynamicQuerySession()`.

   Configure this as a startup rule function (in the CDD file). For more on dynamic query sessions, see *TIBCO BusinessEvents Event Stream Processing Query Developer's Guide*.

2. Add another rule function. Configure it as an event preprocessor (in the CDD file).

3. In the event the preprocessor function, create a query string that selects entities from the cache. You can use any valid BQL query.

4. Use the query string in the following function:

   ```
   Query.Util.executeInDynamicQuerySession(queryString, null, true)
   ```

   The function returns a list of entities from the cache.

5. Use the list returned as needed. For example, iterate over the list and load the entities, as shown in the following example.

   Note: You can also do cache lookups using `@id` and `@extId` attributes.

Ungrouping Query Results

By default, TIBCO BusinessEvents tries to club the calls to the Callback rulefunction when multiple events qualify for a callback at the same time. For example, a query that counts the arriving events may callback with counts like 300, 303, 308 instead of 300, 301, 302 and so on.

To disable such grouping and to trigger a Callback rulefunction for each event, set the `be.query.max.coalesce.results` property to `0` in the CDD file at query agent level.

Example Rule Function for a Legacy ActiveSpaces Cache

The following example preprocessor rule function code shows the techniques used for querying a Legacy ActiveSpaces cache. You can then use standard functions to work with the entities returned by the query.

```
/**
 * @description
 */
void rulefunction RuleFunctions.InputEventPreprocessor {
  attribute {
    validity = ACTION;
  }
  scope {

    Events.InputEvent inputevent;
  }
  body {
    System.debugOut("Now starting InputEventPreprocessor");

    Concepts.Misc misc = Instance.getByExtId("misc");
    String[] EXTIDS_COLLECTION = Instance.PropertyArray.toArrayString
(misc.EXTID_LIST);
    String queryString =
              "select c" +
              "\n from /Concepts/TestConcept as c" +
              "\n where c@extId in"
              + "(\""
              + EXTIDS_COLLECTION[0]
              + "\", \"" + EXTIDS_COLLECTION[1]
              + "\", \"" + EXTIDS_COLLECTION[2]
              + "\", \"" + EXTIDS_COLLECTION[3]
              + "\", \"" + EXTIDS_COLLECTION[4]
              + "\")";
    System.debugOut("Executing query: \n" + queryString);
    Object resultList = Query.Util.executeInDynamicQuerySession
(queryString, null, true);
    System.debugOut("Query returned: " + Query.Util.sizeOfList
(resultList) + " rows");

    while(Query.Util.sizeOfList(resultList) > 0){
      Concepts.TestConcept c = Query.Util.removeFromList(resultList, 0);
      Cluster.DataGrid.CacheLoadEntity(c);

      System.debugOut("Now loading concept " + c + " in the pre-
processor");
    }
  }
}
```

Native Queries

You can use native query to improve the query performance. Native queries are run directly on the cache or store provider that is configured in the project CDD. Native querying is supported for Apache Ignite.

A native query by default returns EntityType results that are the same as a BQL query. All the functions including column name and type are the same as a BQL query.

For a native query to return EntityType results, a new parameter `isNativeQueryRawResults` is added. The default value of this parameter is false.

On importing the projects from older BusinessEvents versions, prior to version 6.3.1, a native query returns raw results. So, the value of `isNativeQueryRawResults` parameter is true for imported projects. However, the native query can be set to return the EntityType results by setting this parameter to false in the following `Query` catalog functions:

- Query.Util.executeInDynamicQuerySession()
- Query.Util.executeInQuerySession()
- Query.create()

For details about the functions, see *TIBCO BusinessEvents Event Stream Processing Query Developer’s Guide*.

A schema name can be provided in a native query unlike a BQL query. For example,

```
select * from NATIVEQEXT.BE_GEN_Concepts_Common_Customer as cu where
cu.age=4";
Schema name: NATIVEQEXT
```

In the Query.ResultSet.get() method to get columns for complex columnTypes, such as Contained concept or Reference concept or Array returns the same results as a BQL query.

For example, the Get() method for the Contained concept returns the Concept object and the columnType returns the EntityType results.

The following table describes the property that can be used to modify the native query results if required:

Properties	Returns	Description
`be.engine.native.query.getAll.columns`	Boolean	To view all the columns including those which are not part of the Ignite Entity properties, set the value of the property to `true`. The default value is `false`. For example, _parentId, _rrf, and _version. **Note:** The columns that are not part of the Ignite Entity properties are excluded in a BQL query. So, they are excluded in Native queries too.

To use native querying, prefix the usual query with the `native-query:` string:

```
native-query: SELECT column1, column2 FROM table_name;
```

For example, refer to the following native query example with the Apache Ignite as a cache provider:

Original Query	Native Query
`SELECT age, count FROM /Concepts/Common/Customer;`	`native-query: SELECT age, count FROM be_gen_Concepts_Common_Customer;`

Note:

- The tables must be present internally with the provider in use.
- The native query format must conform to the query syntax of the provider in use.
- Native queries work from both the BQL console and project code.
- Native queries only work for snapshot queries.
- Native queries run on the cache for cache object management mode irrespective of the persistence configuration.
- For limited cache, native queries run on the limited records that are available in the cache.
- If the `be.engine.native.query.rawResults.enabled` property is set to `false`, the DISTINCT, GROUP BY, and *aggregate functions* are not supported.

Structure of a Function Catalog

A function catalog is an XML file that conforms to the schema file `function_catalog.xsd`.

This allows TIBCO BusinessEvents to integrate your custom functions with the function registry in TIBCO BusinessEvents Studio. The function catalog must be in the XML format shown below and described in Elements in the Function Catalog to map properly to the schema.

Note:

- Name the function catalog `functions.catalog`.
- Place `functions.catalog` in the root folder of the required Java archive resource (`.jar`) file.

Example Function Catalog

This example shows two functions from the standard functions catalog as an example to follow.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<catalog name="Standard">
   <category>
      <name>System</name>
         <function>
            <name>currentTimeMillis</name>
            <class>com.tibco.be.functions.System.SystemHelper</class>
            <args></args>
            <async>false</async>
            <reevaluate>true</reevaluate>
            <isValidInQuery>true</isValidInQuery>
            <isValidInBUI>true</isValidInBUI>
            <helpUrl/>
         </function>
   </category>
   <category>
      <name>Event</name>
         <function>
            <name>createEvent</name>
            <class>com.tibco.be.functions.event.EventHelper</class>
            <args>stylesheet, entityArray</args>
            <isActionOnly>true</isActionOnly>
            <desc>Create an event using XSLT Mapper.
                  This returns a event entity</desc>
            <async>false</async>
            <mapper>
               <enable>true</enable>
               <type>xslt</type>
               <inputElement>
                  <xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                     <xsd:element name="createEvent">
                        <xsd:complexType>
                           <xsd:sequence>
                              <xsd:element name="event"
type="xsd:anyType"
                                          minOccurs="1" maxOccurs="1"/>
                           </xsd:sequence>
                        </xsd:complexType>
                     </xsd:element>
                  </xsd:schema>
               </inputElement>
            </mapper>
            <helpurl></helpurl>
            <isValidInBUI>true</isValidInBUI>
      </function>
   </category>
```

```
</catalog>
```

Elements in the Function Catalog

The Function Catalog consists of various elements and subelements described in this table.

Function Catalog Elements

Element Name	Sub-Elements	Description
`<catalog name="`*`name`*`">`		The root element. Attribute: `name="`*`name`*`"` where *name* is a name that you provide for this functions catalog. `Example: <catalog name="custom">`
`<category>`		This is a subelement of `<catalog>`. `<category>` is a nesting container for a set of related functions within this function catalog.
	`<name>`	A name you provide for this category.
`<function>`		A container for the information about a single function.
	`<name>`	The name of the function.
	`<class>`	The java class that implements the function.
	`<desc>`	Optional. A description of the function.
	`<async>`	Set to `true` if the function runs asynchronously. Set to `false` if the function runs synchronously.
	`<helpurl>`	Not used in this release.
	`<args>`	A comma-separated list of descriptive names for the

Element Name	Sub-Elements	Description
		function's arguments. TIBCO BusinessEvents takes the argument type from the function itself.
	`<isActionOnly>`	If this function has side effects, for example, if it can modify values, you can only use it in action rules. Set this parameter to `true` to alert TIBCO BusinessEvents that this function has side effects and is not valid in these conditions. Otherwise, set to `false`. Valid values: `true`, `false`.
	`<isValidInBui>`	If this function can be used in decision tables, set this element to `true`. Otherwise, set to `false`. Valid values: `true`, `false`.
	`<isValidInQuery>`	If this function can be used in queries, set this element to `true`. Otherwise, set to `false`. Valid values: `true`, `false`.
	`<reevaluate>`	Relevant only when a function is used in a condition. Valid values: `true`, `false`.

Using the Reevaluate Element

The `<reevaluate>` element of a function catalog is relevant only when a function is used in a condition.

Its effect is as follows:

- If set to `true`:
    - TIBCO BusinessEvents does not memorize the result of the evaluation of the condition that contains this function.
    - If any of the conditions is reevaluated, then this function is also reevaluated.

      For example, `<reevaluate>` is set to true for `currentTimeMillis()`. Given this condition:

      ```
      stock.price > 10.0;
        currentTimeMillis() - stock.time > 600000;
      ```

      If the condition `stock.price > 10.0` is reevaluated, the `currentTimeMillis()` is also reevaluated.
- If set to `false`:
    - TIBCO BusinessEvents calls the function during the first evaluation and stores the result that is stored and used for subsequent condition evaluations.
    - TIBCO BusinessEvents Studio reevaluates the condition only if another part of the same condition changes.
    - In the above stock price example, if `<reevaluate>` is set to false, then the condition is reevaluated only when `stock.time` changes.

Rule Language Grammar

There are some basic concepts that you need to know about working with the rule language grammar.

Whitespace

Whitespace is used to separate tokens (identifiers, keywords, literals, separators, and operators) just as it is used in any written language to separate words. Whitespace is also used to format code.

These are whitespace characters, excluding line terminators:

- the ASCII SP character, also known as *space*
- the ASCII HT character, also known as *horizontal tab*
- the ASCII FF character, also known as *form feed*

Line terminators include these characters:

- the ASCII LF character, also known as *newline*
- the ASCII CR character, also known as *return*
- the ASCII CR character followed by the ASCII LF character

Comments

Comments are used to comment a rule.

Comment rules are as follows:

Single Line Comment

`//` *text*
The text from `//` to the end of the line is ignored

Multi-line Comment

`//` *text*
The text from `//` to the end of the line is ignored

Separators

Separators are used to separate statements, expressions, or arguments in a function.

The following tokens are used for separators:

Tokens	Description
;	Statement separator for conditions and actions.
(	Expression Grouping begin, or function argument list begin.
)	Expression Grouping end or function argument list end.
,	Argument list separator.

Identifier Naming Requirements

An identifier (or *name*, to use the user interface label) is an unlimited-length sequence of letters and digits, the first of which must be a letter. Letters include uppercase and lowercase ASCII Latin letters A-Z, a-z, and the underscore (_).

Here are some guidelines for naming of identifiers:

- Do not use the dollar sign ($).
- Identifiers are case sensitive.
- Identifiers cannot have spaces (except shared resource identifiers).
- Identifiers may not be the same as any literal, keyword, or other reserved word. See Keywords and Other Reserved Words and Literals.

Letters and digits may be drawn from the entire Unicode character set, which supports most writing scripts in use in the world today, including the large sets for Chinese, Japanese, and Korean. This allows programmers to use identifiers in their programs that are written in their native languages.

Digits include the ASCII digits 0-9, while two identifiers are the same only if they have the same Unicode character for each letter or digit. Note that some letters look the same even though they are different Unicode characters. For example, a representation of the letter A using \u0041 is not the same as a representation of the letter A using \u0391.

Two example identifiers: `new_order E72526 creditCheck`

Tip: Here is a more succinct way for programmers to understand the requirements:

```
<Identifier>  :=  [ <ID_START> { <ID_PART> }* ]
<ID_START> :=  except '$', any character for which
java.lang.Character.isJavaIdentifierStart() returns true
<ID_PART> :=  except '$', any character for which
java.lang.Character.isJavaIdentifierPart() returns true
```

Local Variables

Local variables of certain types are used in rule actions and rule functions.

The following types can be used:

- Primitives
- Concepts
- Events

Primitive Arrays

Primitive arrays with a fixed length can also be used when working with a rule language grammar .

Here are examples of array declaration, initialization, and array creation expressions:

- Array declaration and initialization:

```
    int i;                // int
    int[] ii = {1,2,i};   // array of int
```

- Array creation with initialization expression:

```
    ii = int[] {1,2,3};
```

- Array creation without initialization expression:

```
    int[] arr = int[5] {};
    arr = int[5]{};
```

- Getting the length of the array:

```
    int arrLength = arr@length;
```

Literals

Certain literals are used for the rule language grammar.

The TIBCO BusinessEvents rule language supports these literals:

int

One or more digits without a decimal. May be positive or negative.

Examples:

```
4   45   -321   787878
```

long

An integer literal suffixed with the letter L. The suffixed L can be either upper or lower case, but keep in mind that the lower case L (l) can be difficult to distinguish from the number one (1).

Examples:

```
0l   0777L   0x100000000L   2147483648L   0xC0B0L
```

double

A number that can represent fractional values. D suffix is optional unless there is no decimal point or exponent.

Examples:

```
1D   1e1   2.   .3   0.0D   3.14   1e-9d   1e137
```

String

Zero or more characters enclosed in double quotes (""). The string must be on one line. Use `\n` for newlines. Use the plus sign (+) to concatenate string segments.

Examples:

- Empty string: `""` (quotes with no space).
- Space character: `" "` (quotes with one or more spaces).
- Strings with values:

  ```
  "P0QSTN3" "The quick brown fox had quite a feast!"
  ```

- Strings spanning multiple lines:

  ```
  "The quick brown fox " +
  ```

  ```
  "had quite a feast!"
  ```

boolean

One of these two values: `true` `false`

Null

This value: `null`

Escape Sequences

Escape sequences are used to represent characters when working with the rule language grammar.

You can represent characters in literals using these escape sequences:

Escape Sequences

Character	Escape Sequence
backspace	`\b`
horizontal tab	`\t`
linefeed	`\n`
form feed	`\f`
carriage return	`\r`
double quote	`\"`
single quote	`\’`
backslash	`\\`

Operators

Certain operators are defined for use with the rule language grammar.

Operators in this list are also used in the Java language and work the same as the Java operators:

Operators in the TIBCO BusinessEvents Rule Language

Operator	Notes
`++ --`	increment, decrement
`+ -`	unary plus, unary minus
`* \ %`	multiplication, division, remainder
`+ -`	addition, subtraction
`> < >= <=`	greater than, less than, greater than or equal to, less than or equal to
`=`	assignment
`== !=`	equality, inequality (Does deep string comparison, unlike Java.)
`&& || !`	boolean AND, OR, NOT
`-= += *= /= %=`	combined operation and assignment. As an example, `expr += 2`; is the same as *expr* = *expr* + 2; `+=` works on strings as well as numbers. For example, you have a variable *String s* = *"abc"*; and then you use `s+= "def"`; and so the value of `s` becomes `"abcdef"`

Operator	Notes
`instanceof`	Tests whether an object is an instance of specified type. Restricted to use with concepts and events. Example: `boolean b = customer instanceof USCustomer;`
`.`	property access
`@`	attribute access

Attributes

TIBCO BusinessEvents provides attributes that you can use in rules to access information of various kinds. Use the @ operator to access attributes.

Attributes

Entity	Attributes	Type	Returns
SimpleEvent	`@id`	`object`	The event’s unique internal ID.
	`@extId`	`string`	The event’s unique external ID.
	`@ttl`	`long`	The time to live of the event as specified in the configuration. (This is not the time-to-live remaining.)
	`@payload`	`string`	The payload as a string value.
Repeating TimeEvent	`@id`	`object`	The time event’s unique internal ID.
	`@closure`	`string`	null.
	`@interval`	`long`	The number of units between creation of successive time events.
	`@scheduledTime`	`dateTime`	The time scheduled for asserting into the Rete network.
	`@ttl`	`long`	0.
Rule-Based TimeEvent	`@id`	`object`	The time event’s unique internal ID.
	`@closure`	`string`	A string that was specified when

Entity	Attributes	Type	Returns
			the event was scheduled.
	`@interval`	`long`	0.
	`@scheduledTime`	`dateTime`	The time scheduled for asserting into the Rete network.
	`@ttl`	`long`	The time to live of the event as specified when scheduling the event. (This is not the time-to-live remaining.)
Advisory Event	`id`	`object`	The advisory event's unique internal ID
	`extId`	`String`	Null
	`category`	`String`	Broad category of advisory, for example, an exception.
	`type`	`String`	Type of advisory within the category.
	`message`	`String`	Message for the user.
Concept	`@id`	`object`	The concept instance's unique internal ID.
	`@extId`	`string`	The concept instance's unique external ID.
ContainedConcept	`@id`	`object`	The contained concept instance's unique internal ID.
	`@extId`	`string`	The contained concept instance's unique external ID.

Entity	Attributes	Type	Returns
	`@parent`	`concept`	The parent concept instance. (This is treated as a concept reference in the language.)
PropertyAtom	`@isSet`	`boolean`	True if the property value has been set. Otherwise, false.
PropertyArray	`@length`	`int`	The number of PropertyAtom entries in the array.

Note: The internal ID is automatically generated by TIBCO BusinessEvents. You cannot set it.

Exception Handling

The TIBCO BusinessEvents Rule Language includes `try`, `catch`, and `finally` blocks and has an error type attribute `@errorType`.

The `try`, `catch`, and `finally` blocks behave like their same-name Java counterparts.

Note: You can also use the special AdvisoryEvent event type to be notified of exceptions that originate in user code but that are not caught with a `catch` block. To use the AdvisoryEvent, click the plus sign used to add a resource to the declaration. AdvisoryEvent is always available in the list of resource.

Syntax

Exception handling syntax defines which combinations can be used.

These combinations are allowed:

- `try/catch`
- `try/finally`
- `try/catch/finally`

try

```
try {try_statements}
```

catch

```
catch (Exception identifier ) {catch_statements }
```

finally

```
finally {finally_statements}
```

Examples

Exception handling syntax for the rule language grammar is presented through some examples.

try/finally Example

```
String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} finally {
    //If readStatus() throws an exception,     //MyScorecard.status will
be set to "default status"
    //but the exception won't be caught here.
    //Otherwise MyScorecard.status will be set to the
    //return value of readStatus()
    MyScorecard.status = localStatus;
}
```

try/catch/finally Example

```
String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} catch(Exception exp) {
    System.debugOut("readStatus() threw an exception with message"
                   + exp@message);
} finally {
    //If readStatus throws an exception,      //MyScorecard.status will
be set to "default status"      //Otherwise MyScorecard.status will be
set to the      //return value of readStatus()
    MyScorecard.status = localStatus;
}
```

try/catch Example

```
String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} catch(Exception exp) {
    System.debugOut("readStatus() threw an exception with message "
               + exp@message);  }
    //If readStatus throws an exception,      //MyScorecard.status will
be set to "default status"      //Otherwise MyScorecard.status will be
set to the      //return value of readStatus()
MyScorecard.status = localStatus;
```

try/catch Example with Checking errorType and rethrow

```
String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
catch(Exception exp) {
    if (exp@errorType == "java.lang.NullPointerException")
         throw exp;
    System.debugOut("readStatus() threw an exception with message " +
exp@message);
}
//If readStatus throws an exception other than NullPpointerException,
//MyScorecard.status will be set to "default status"
//If readStatus throws NPE, MyScorecard.status will be set to "default
status" and NPE will be re-thrown.
//Otherwise MyScorecard.status will be set to the return value of
readStatus()
MyScorecard.status = localStatus;
```

Flow Control

The TIBCO BusinessEvents rule language includes commands to perform conditional branching and iteration loops.

if else

The `if/else` command allows you to perform different tasks based on conditions.

Syntax:

```
 if(condition){
code_block;
 }
 else{
code_block;
 }
```

for

The `for` command allows you to create a loop, running a code block until the condition you specify is false.

Syntax:

```
for(initialization; continue condition; incrementor){
code_block;
[break;]
[continue;]
}
```

where:

`break` allows you to break out of the loop.

`continue` allows you to stop running the code block but continue the loop.

For example:

```
    for(int i=1; i<10; i=i+1){
    System.debugOut("Hello World!");
    }
```

This example prints "Hello World!" to `debugOut` 10 times.

while

The `while` command allows you to perform one or more tasks repeatedly until a given condition becomes false.

Syntax:

```
while(condition){
    code_block;
    [break;]
    [continue;]
}
```

where:

`break` allows you to break out of the loop.
`continue` allows you to stop running the code block but continue the loop.

Rule Language Datatypes

Rule language datatypes are defined with conversion tables, with information about operators and types, and with information about how TIBCO BusinessEvents handles inconstancy problems with datatypes.

Concept Properties to XML Datatype Conversions

Concept Properties to XML Datatype Conversions

Property Type	`Int`	`Long`	`Float`	`Double`	`Boolean`	`String`	`DateTime`	`ComplexType`	`@ref`
`int`	L	L	L	L		L			
`long`	N	L	N	N		L			
`double`	N	N	N	L		L			
`String`	L	L	L	L	L	L	L		
`boolean`					L	L			
`Datetime`						L	L		
`ContainedConcept`								D	
`ConceptReference`									ID

N - Numeric conversion, loss of information is possible (see note below).

L - Shallow copy—Copies only the current value.

D - Deep copy—Copies the entire structure of the contained concept (current values of all properties only).

ID - Basically a shallow or reference-only copy. The copy refers to the same instance of the concept.

Note:

- History is never copied.
- Data loss is possible for conversions from string to a number datatype if the string represents a very large number that would have to be clipped.
- Datatype conversion tables for events are in the TIBCO Enterprise Message Service documentation.

Compatibility of Operators with Types

Rule language grammar defines operators compatibility with types.

Compatibility is described as follows:

Operator Matrix

	Right Side of Operator								
		str	**int**	**lon**	**du**	**boo**	**ent**	**obj**	**dat**
Left Side of Operator	`str`	=, +, eq, cmp, inst	+	+	+	+	+	=, +, eq, cmp, inst	+
	`int`	+	=, math, eq, cmp	=, math, eq, cmp	=, math, eq, cmp			=, math, eq, cmp	
	`lon`	+	=, math,	=, math,	=, math,			=, math,	

	Right Side of Operator								
		str	int	lon	du	boo	ent	obj	dat
			eq, cmp	eq, cmp	eq, cmp			eq, cmp	
	`dou`	+	=, math, eq, cmp	=, math, eq, cmp	=, math, eq, cmp			=, math, eq, cmp	
	`boo`	+				=, eq		=, eq	
	`ent`	+					=, eq, inst	=, eq, inst	
	`obj`	=, +, eq, cmp, inst	=, math, eq, cmp	=, math, eq, cmp	=, math, eq, cmp	=, eq	=, eq, inst	=, eq, inst	=, eq, inst
	`dat`	+,						=, eq, inst	=, eq, cmp, inst

Abbreviation	Meaning and Notes
`boo`	Boolean.
`cmp`	Comparison operators: <, >, <=, >=

Abbreviation	Meaning and Notes
`dat`	Date/Time
`dou`	Double
`ent`	Entity. Type includes Concept, Event, and Scorecard. Both operands must either be of the same type or have a subtype-supertype relationship
`eq`	Equality operators: ==, !=
`inst`	instanceof
`int`	Integer
`lon`	Long
`math`	Numerical operators: unary +, unary -, =, - , *, /, %
`obj`	Object
`str`	String

Correcting Inconsistencies of Type

TIBCO BusinessEvents attempts to correct inconsistencies of type whenever possible by converting expressions to the appropriate type.

It converts expression types in the following cases:

- An expression uses the plus sign (+) with a string operand.
- An arithmetic expression includes numbers of differing types.
- The value of an expression is assigned to a variable of a different type.
- The value of an expression is passed to a function that declares a different type.

There are some inconsistencies of type that TIBCO BusinessEvents cannot correct. For example, all expressions within conditions must be of type boolean. If an expression within a condition evaluates to anything other than boolean, it would be illogical for TIBCO BusinessEvents to convert the expression to boolean. In cases like this, TIBCO BusinessEvents returns an error at compile time.

String Operands

When an expression uses the plus sign (+) with a string operand, TIBCO BusinessEvents treats the expression as a request for concatenation rather than addition. It converts the second operand to a string and concatenates the two strings.

For example:

```
"area code:  " + 650
```

becomes

```
"area code:  650"
```

Arithmetic Expressions

The following information applies to these operators:

```
*  /  %  +  -  <  <=  >  =  ==  !=
```

When an expression uses one of the above arithmetic operators with two numbers of different numeric types, TIBCO BusinessEvents promotes one of the two operands to the numeric type of the other. It makes these promotions as follows:

- If either operand is a double, TIBCO BusinessEvents promotes the other to a double.

- Otherwise, if either operand is a long, it promotes the other to a long.

Assignment Conversion

If the value of an expression is assigned to a variable, TIBCO BusinessEvents converts the expression’s type to that of the variable. This might include, for example, converting a double to an int, or converting a generic model type to a more specific model type.

Function Argument Conversion

Conversions of function arguments are handled in the same way as assignment conversions.

Mapping and Transforming Data

Variables in the scope of a rule or rule function can be mapped to arguments of a function used in that rule or rule function.

See also XPath Formula Builder For XPath 1.0 for related information, as well as Rules and Functions.

The Function Argument Mapper allows you to supply the data that a function expects as input. For instructions on accessing the Function Argument Mapping wizard in the rule editor, see Using the Function Argument Mapping Wizard.

XPath 1.0 and XPath 2.0

TIBCO BusinessEvents supports XPath 1.0 and XPath 2.0. The default is XPath 2.0 for the new projects. TIBCO BusinessEvents uses the Standard Widget Toolkit (SWT) mapper to support XPath 2.0 and XSLT 2.0.

The following BusineeEvents functions SWT mapper, and the underlying implementation of these functions uses the related XML runtime:

- Event.createEvent
- Instance.createInstance
- All XPath.evalAs functions, for instance XPath.evalAsString

Using the SWT mapper for XPath 2.0, you can perform the mapping in the same way you can do for XPath 1.0. However, the mapper uses the XPath 2.0 expressions. In contrast to XPath 1.0, the type casting from one data type to another data type is not done implicitly in XPath 2.0. In XPath 2.0, you must perform the explicit typecasting using the constructor function. BusinessEvents also provides the option in the Function Argument Mapping wizard to autofix the typecasting error. You can fix the common issues using the Mapper Function Migration wizard. See Migrating Mapper Functions from XPath 1.0 to XPath 2.0 for more details about how to common mapping issues. You can also use the `studio-tools` utility to fix those common issues. See Migrating Mapper Functions to XPath 2.0 Using the CLI for more details.

In XPath 1.0, the mapper has two different windows the Function Argument Mapping and XPath Function Builder. However, in XPath 2.0 both windows are now combined in the same Function Argument Mapping wizard. So, if your project uses XPath 1.0 then you can see the Function Argument Mapping wizard and XPath Formula Builder in different windows. However, if your project uses XPath 2.0 then you see the Edit tab inside the Function Argument Mapping wizard. See Function Argument Mapping Wizard for XPath 2.0.

The mapper for XPath 1.0 had some TIBCO provided functions that were not part of the XPath 1.0 specifications, for instance `format-dateTime`. These functions are currently unavailable with the SWT mapper for XPath 2.0, and the old XPath expression may give an error, in such case you have to fix those errors manually.

All the following sections provide information on mapping data with respect to the XPath 1.0 user interface; however, you can perform all those functions in XPath 2.0. See Function Argument Mapping Wizard for XPath 2.0 for reference on the user interface for XPath 2.0.

Function Argument Mapping Wizard for XPath 1.0

Using the Function Argument Mapping wizard for XPath 1.0 you can map the source data to the input arguments of the mapper function. The wizard consists of two sections: Function and Input.

Function Section

The function section, in the upper part of the dialog, shows the view-only name of the function you are working with and the editable entity path to the item whose properties and attributes you want to map to the function arguments.

Input Section

In the input section, there are two panels:

- Scope Variables Panel

  The scope variables panel shows the list of properties and attributes available to the function, as well as global variables defined in the project.

- Function Panel

  The function panel uses an Extensible Stylesheet Language Transformation (XSLT) template that specifies how scope variables should be transformed to provide the expected input. Normally, you do not need detailed knowledge of XSLT to specify a function’s expected output. However, if you are familiar with XSLT and you wish to see the actual code, you can right-click on any item in the Function panel and choose Copy from the popup menu. Then open a blank text document and choose Paste. The XSLT is displayed in your text document.

  You can also use your own XSLT templates to perform transformations instead of using the techniques described in the following sections. You can paste XSLT into your function input fields, or into the XPath Formula field in XPath Formula Builder. You cannot, however, paste XSLT directly into the function argument in the rule editor.

The Input tab consists of several toolbar buttons, popup menus, and icons to help in performing mapping of data for XPath 1.0. See Buttons Menus and Icons.

Function Argument Mapping Wizard for XPath 2.0

Using the Function Argument Mapping wizard you can map the variable of the source to the argument of a mapper function.

Functions Section

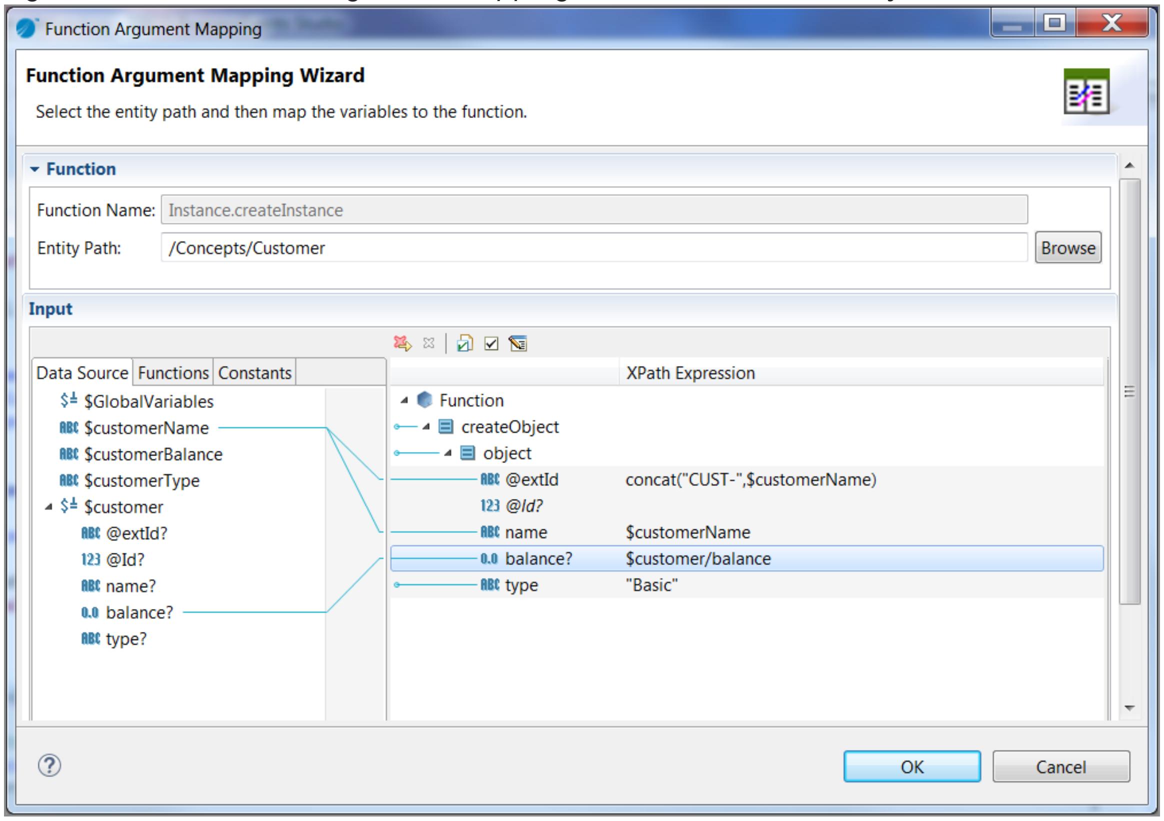
The **Functions** section is similar to the Function Argument Mapping wizard for XPath 1.0. It displays the view-only name of the function you are working with and the editable entity path to the item, whose properties and attributes you want to map to the function arguments.

Input Section

The **Input** section in the Function Argument Mapping wizard for XPath 2.0 is a combination of the Input section of the Function Argument Mapping wizard for XPath 1.0 and XPath Function Builder. For XPath 2.0, there is no separate XPath Function Builder but it has been merged in the Function Argument Mapping wizard. However, the Function Argument

Mapping wizard still performs the same functions and in the same way with only the user interface changes.

Figure 11: The Function Argument Mapping Wizard in XPath 2.0 Project


Function Argument Mapping
Function Argument Mapping Wizard
Select the entity path and then map the variables to the function.
Function
Input
Data Source
Functions
Constants
XPath Expression
Browse
OK
Cancel

Function Name: Instance.createInstance

Entity Path: /Concepts/Customer

```
$GlobalVariables
$customerName
$customerBalance
$customerType
$customer
    @extId?
    @Id?
    name?
    balance?
    type?
```

```
Function
  createObject
    object
      @extId       concat("CUST-",$customerName)
      @Id?
      name         $customerName
      balance?     $customer/balance
      type         "Basic"
```

In the Input section, on the left side, there are three tabs: Data Source, Functions, Constants.

The **Data Source** tab is similar to the Scope Variables panel in the Function Argument Mapping wizard for XPath 1.0. It lists the properties and attributes available to the function, as well as global variables defined in the project.

The **Functions** and the **Constants** tab are the same tabs which are available in the XPath Formula Builder for XPath 1.0. However, the Functions tab now contains the XPath 2.0 mapper functions. Some of these new functions were provided by TIBCO in XPath 1.0, which are now part of the XPath 2.0 function library. See The XPath Formula Builder for more details about these tabs.

On the right side, similar to the Function panel in the Function Argument Mapping wizard for XPath 1.0 the XSLT template is present which specify how the Data Source would be transformed to provide the expected input. There is a toolbar which consists of the tools show the edit tab and auto-fix the errors.

Function Argument Mapping Wizard for XPath 2.0 Toolbar

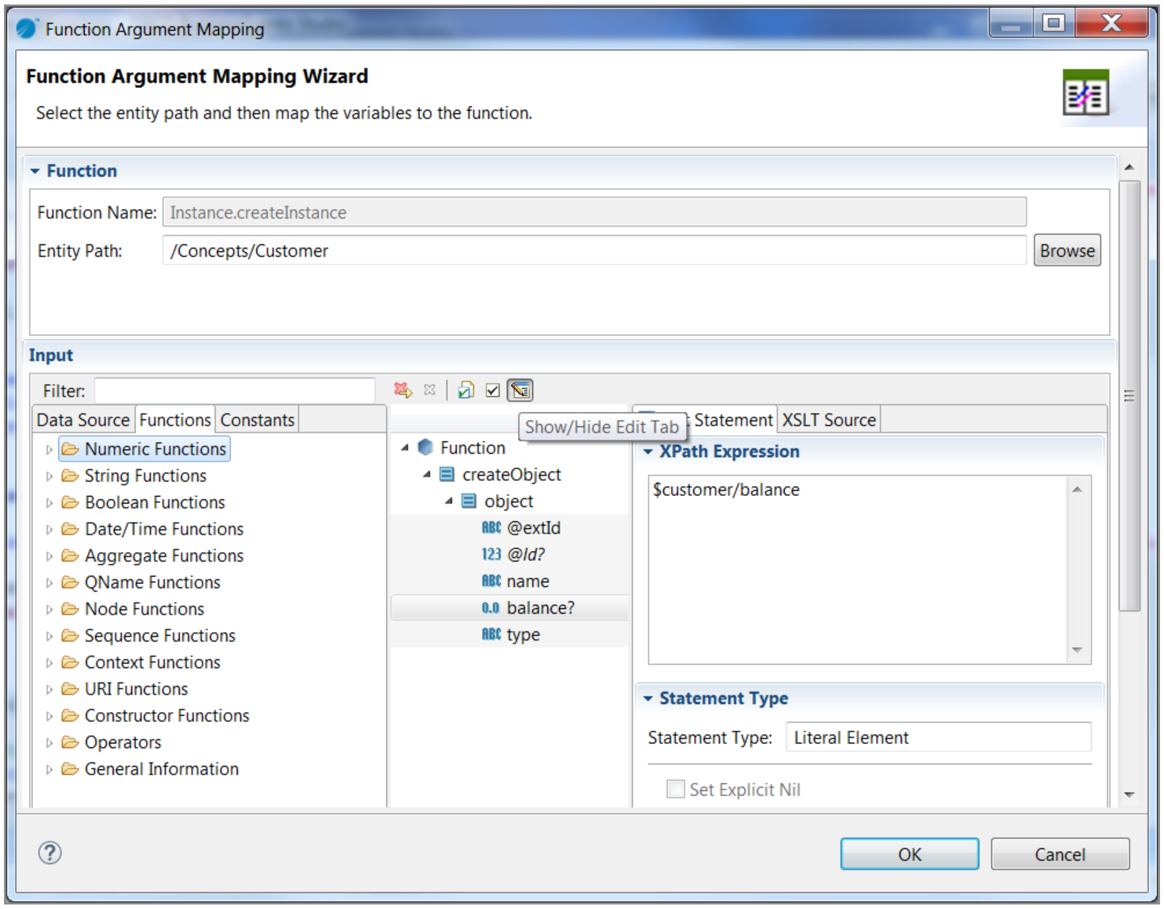
Icon	Tool	Description
	Remove Mapping	Removes the selected mapping from the selected attribute.
	Show Check and Repair	Displays mapper check and repair window similar to XPath 1.0. It verifies the XSLT template you have created in the Function panel against the expected input. A list of errors and warnings appear and you can choose which items you wish to fix. TIBCO BusinessEvents attempts to fix simple problems such as adding missing items that are expected and typecasting errors.
✓	Fix Typecasting Error	Fixes the typecasting error in XPath 2.0 where explicit casting is not done. For BusinessEvents, if the XPath expression elements cannot be implicitly cast, an error is displayed. For instance, if the XPath expression is $price > 1000, and $price is defined as a String, then an explicit cast must be done (the XPath 1.0 mapper does this casting implicitly).
	Show/Hide Edit Tab	Toggles the XPath expression editor for building complex XPath statement for the selected input argument. See XPath Formula Builder For XPath 2.0. This builder functions in the same way in which XPath Function Builder was working in XPath 1.0. See XPath Formula Builder For XPath 1.0 for more details.

XPath Formula Builder For XPath 2.0

XPath 2.0 mapper do not have a separate window for XPath Formula Builder. The expression editor is now part of the Function Argument Mapping wizard and can be shown by the Show/Hide Edit Tab button.

Click the **Show/Hide Edit Tab** icon to display or hide the **XPath Statement** and **XSLT Source** tabs where you can build the complex XPath expression. You can use the data scope variables, mapper functions and constants from the respective tab on the left. In XPath 2.0, the behavior of the formula builder remains the same as XPath 1.0. See XPath Formula Builder For XPath 1.0 for more details about its functioning in XPath 1.0.

Figure 12: XPath Formula Builder for XPath 2.0



Mapping and Transforming Data to Function Input

Note: Assigning an empty string `("")` to a field in a mapper function results in a null string.

To map data, select an item in the Scope Variables panel, then drag and drop that item into the desired schema element in the Function panel.

Simple mappings appear in the formula area to the right of the input element after you release the mouse button. For more complex mappings, click the **Edit Statement** icon (XPath 1.0) or **Show/Hide Edit Tab** icon (XPath 2.0).

Most options in the Edit Statement dialog are straightforward. However, there are some complex scenarios that require multiple steps.

You may also wish to refer to XSLT Statements for a reference of XSLT statements when deciding which XSLT statement can be used to achieve the result you desire.

You can specify XPath formulas to transform an element if you need to perform more complex processing. The XPath Formula Builder allows you to easily create XPath formulas. For more advanced use of XPath, see XPath Formula Builder For XPath 1.0. There are also a variety of third-party books and resources about XSLT and XPath.

The datatypes of the function's arguments display as hints. Once a mapping or formula is specified, a hint becomes an XSLT statement. See Statements Hints and Errors for more information about hints and statements.

Statements Hints and Errors

When you display the Function tab, the existing statements are examined, and any input elements that do not have a statement are displayed as hints. Hints are reminders that you can specify a statement for the input element, but they are not stored as part of the XSLT template for the function's input. Hints are displayed in italics with a light blue background. Once you specify a mapping or a formula for a hint, the input element becomes a statement. You can also drag the hint to the left past the dividing line between the panels and the hint becomes a blank statement.

Once you specify a statement in the Function panel and click OK, it becomes part of the XSLT template used to create the input data. Statements are only deleted if you manually delete them using the delete button, or if you use the **Mapper Check and Repair** (✓) button to automatically fix errors. Therefore, if the input schema for the function changes, your statements may no longer be valid. See Tester Preferences for more information about using the **Mapper Check and Repair** button to fix statements in the Function panel.

Any statement or hint that has an error is displayed in red. A hint is only displayed in red if it is a required input element. All required input elements must have statements specified. The **Mapper Check and Repair** button can help you automatically fix some errors. See Incorrect Mappings for more information about fixing errors.

Buttons Menus and Icons

The Input tab contains several toolbar buttons, popup menus, and icons. This section describes the various graphical elements of the Input tab.

Toolbar and Right-Click Menu On the Input Section

The Scope Variables panel and the Function panel have several buttons for performing various functions. There is also a popup menu when you right-click on the elements in each panel. The following table describes the buttons and right-click menu items available in the panels of the **Input** tab.

Input tab toolbar buttons

Button	Right-Click Menu	Description
Scope Variables Panel		
		Coercions. It allows you to specify a type for Scope Variables elements that are not a specific datatype. For example, a choice element can be coerced into one of the possible datatypes for the element, or an element of a datatype `any` can be coerced into a specific datatype.
		Type Documentation. It allows you to specify or view documentation for schema elements.
	Expand	This menu item has two submenus: Content and All. **Expand > Content** expands the current element so that all elements that are currently used in a mapping are visible. **Expand > All** expands all subelements of the currently selected element.
	Show Connected	Expands the elements in the Function area to display elements that are mapped to the currently selected element or its subelements.
	Delete	Deletes the selected element.

Button	Right-Click Menu	Description
	Copy	Copies the selected element. The element can be later pasted.
Function Panel		
☰		Shows or hides the mapping formulas for the input elements.
⇧		Move Up. Moves the selected element up in the Function tree.
⇩		Move Down. Moves the selected element down in the Function tree.
⇦	Move Out	Move Out. Promotes the selected element to the next highest level in the Function tree.
⇨	Move In	Moves the currently selected element into a new statement. This displays the Move Into New Statement dialog that allows you to choose the statement you wish to move the element into. See XSLT Statements for more information about XPath statements.
✕	Delete	Deletes the mapping for the selected element. If no mapping is defined, the element itself is deleted (along with all child elements). **Note:** Elements are predefined. Do not delete elements. Deletion of an element causes mapper validation errors due to the mismatch of the right panel's content with its schema.
✚		Insert. Click Insert to pop up a New XSLT Statement dialog where you can define an XSLT statement. The statement is inserted in the function input schema on the same level of the hierarchy as the currently selected element. You can add one XSLT statement at a time with this button. The right-click menu item Statement provides a shortcut for multi-line statements, such as Choice or If. See the description of the

Button	Right-Click Menu	Description
		Statement menu item below for more information. **Note:** Elements are predefined. Do not add new elements. Doing so causes a mismatch of the right panel's contents with its schema. See XSLT Statements for more information about XSLT statements.
✚		Add Child. Adds a statement for a child element to the currently selected element.
✓		Mapper Check and Repair. Verifies the XSLT template that you have created in the Function panel against the expected input. A list of errors and warnings appear and you can choose which items you wish to fix. TIBCO BusinessEvents attempts to fix simple problems such as adding missing items that are expected. See Incorrect Mappings for more information.
!		Edit Statement. It allows you to modify an XSLT statement for the element. See XSLT Statements for more information about XSLT statements.
✏	Edit	XPath Formula Builder. It invokes the XPath formula builder. You can use this editor to create an XPath statement for this input element. See XPath Formula Builder For XPath 1.0 for more information about XPath and the XPath formula builder.
	Expand	This menu item has three submenus: Content, Errors, and All. **Expand > Content** expands the current element so that all subelements that have a mapping or expression are visible. **Expand > Errors** expands the current element so that all subelements that have an error in their expression are visible.

Button	Right-Click Menu	Description
		Expand > All expands all subelements of the currently selected element.
	Show Connected	Expands the elements in the Scope Variables panel to display elements that are mapped to the currently selected element or its subelements.
	Statement	This menu item contains shortcuts that allow you to add easily one or more desired XSLT statements with one menu item instead of adding one or more statements with the **Insert** button. See Statement Menu Options for a description of the subitems of this menu.
	Undo *operation*	Rolls back the last operation performed. The name of the last operation is shown.
	Redo *operation*	Performs the last operation that was undone with the **Undo** menu item. The name of the last operation is shown.
	Cut	Deletes the selected element. The element can later be pasted to a new location.
	Copy	Copies the selected element.
	Paste	Pastes the last element that was copied or cut.

Icons for Schema Element Datatypes

Schema elements also have a set of associated icons to indicate their type. The following table describes the icons used for schema items.

Tip: You can use the **Type Documentation** button to obtain any available documentation on any node in the Scope Variables or function input schema trees.

Icons for schema items

Icon	Description
▤	Complex element that is a container for other datatypes. This is also called a *branch* in the schema tree.
ABC	Simple string or character value.
123	Simple integer value.
3.1	Simple decimal (floating point) number.
0/1	Simple boolean value.
▦	Simple Date or Time. This can be any of the following datatypes: • Time • Date • Date & Time • Duration • Day • Month • Year • Month & Year • Day & Month
01	Simple binary (base 64) or hex binary value.
■	Represents a schema item that can be any datatype. Data in this schema element can be any datatype.
◈	Choice. Specifies that the actual schema element can be one of a specified set of datatypes.

Qualifier Icons

Schema elements can have additional icons that specify qualifications. The qualifier icons have different meanings depending upon where they appear. For example, a question mark icon signifies an element is optional in the Scope Variables schema or in a hint in the Function panel. However, in an XSLT statement, the question mark signifies the statement is "collapsed" and an implicit "if" statement is set, but not displayed in the Function panel.

The following table describes the additional qualifiers that appear next to the name of schema items.

Additional icons for hints

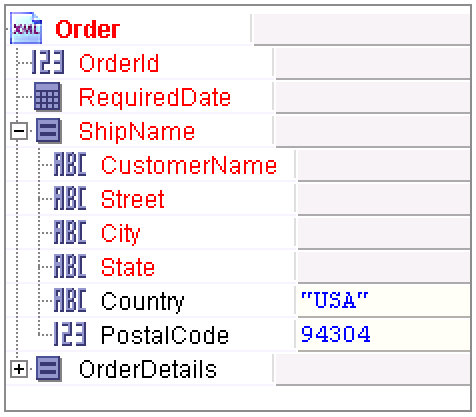
Qualifier	Scope Variables or Hint	Statement
	No qualifier indicates the element is required.	N/A
▨	A question mark indicates an optional Item.	An implicit "if" statement is set for this statement. This occurs when you map an optional element from the Scope Variables to an optional element in the function input schema or if you specify Surround element with if test on the **Content** tab of the **Edit Statement** dialog.
*	An asterisk indicates the item repeats zero or more times.	N/A
+	A plus sign indicates the item repeats one	N/A

Qualifier	Scope Variables or Hint	Statement
	or more times.	
⊠	A null sign indicates the item may be set to null.	A null sign indicates the item is explicitly set to null. You can set an element explicitly to null by clicking the **Edit Statement** button for the element, then checking the Set Explicit Nil field on the **Content** tab of the Edit Statement dialog.

Specifying Constants

For each element in the Function input schema tree, you can specify a constant. Constants can be strings or numeric values. To specify a string, enclose the string in quotes. To specify a number, type the number into the schema element's mapping field. The following illustrates specifying the string "USA" for the `Country` item and 94304 for the `PostalCode` item of a function input schema.

Figure 13: Specifying Constants



```
Order
  OrderId
  RequiredDate
  ShipName
    CustomerName
    Street
    City
    State
    Country        "USA"
    PostalCode     94304
  OrderDetails
```

Constants can also be used in functions and search predicates. To learn more about complex XPath expressions that use functions and search predicates, see XPath Formula Builder For XPath 1.0.

Date and Datetime Strings in Constants

In constant expressions used in bindings, datetime values are read in according to the ISO 8601 standard, as described in the XML Schema specification. For example, the value:

```
"2002-02-10T14:55:31.112-08:00"
```

is 55 minutes, 31 seconds, and 112 milliseconds after 2pm on February 10, 2002 in a timezone that is 8 hours, 0 minutes behind UTC.

If no timezone field is present, the value is interpreted in the timezone of the machine that is performing the parsing. This can lead to complications if you are processing data from different timezones. So you are always encouraged to use the timezones.

When TIBCO BusinessEvents generates datetime strings, UTC time is always used. The output for the example above is:

```
2002-02-10T14:55:31.112Z
```

which is the equivalent time in the UTC timezone.

Data Validation

Data passed as input to a function is validated to ensure that it conforms to its specified datatype.

Data Validation describes the validation behavior. Datatype validation listed with the prefix `xsd:` is defined in the namespace http://www.w3.org/2001/XMLSchema. See *XML Schema Part2: Datatypes* specification at http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/ for more information on the proper representation of these datatypes.

Datatype validation listed with the prefix `xdt:` is defined in the namespace http://www.w3.org/2003/11/xpath-datatypes. See *Xquery 1.0 and Xpath 2.0 Functions and Operators* specification at http://www.w3.org/TR/2003/WD-xpath-functions-20031112/ for more information on the proper representation of these datatypes.

Datatype validation

Data Type	Validation
`boolean`	`xsd:boolean`
`double`	`xsd:double`
`string`	`xsd:string`
`dateTime`	`xsd:dateTime`
`long`	`xsd:long`
`int`	`xsd:int`

Incorrect Mappings

Any incorrect statements are displayed in red in the Function panel.

Errors can occur for a number of reasons. For example,

- a required element has no statement, and therefore must be specified
- the function’s input schema has changed and existing statements may no longer be valid
- the XPath formula for an element may contain an error

You should correct any errors before attempting to test or deploy your process definition.

To help find potential problems in your mappings, click the Mapper Check and Repair button. This button displays a dialog with all potential problems in the specified mappings. You can select the Fix checkbox for potential errors, and TIBCO BusinessEvents attempts to fix the problem.

Some potential problems in the **Mapper Check or Repair** dialog cannot be fixed easily, and therefore there is no checkbox in the Fix column for these items. For example, if an element expects a string and you supply a complex type, the corrective action to fix the problem is not clear, and therefore TIBCO BusinessEvents cannot automatically fix the problem. You must repair these items manually.

XPath 2.0

For XPath 2.0, in addition to the earlier specified errors, you can also see the errors when there is no explicit typecasting of different data types, such as, mapping a boolean value (`$processapplication/start`) to a input of data type double. In such cases, you can fix the typecasting error in following ways:

- Run the Mapper Function Migration wizard to fix the common mapper issues. See Migrating Mapper Functions from XPath 1.0 to XPath 2.0 for more details.
- Click the **Fix Typecasting Error** ✓ icon to autofix all typecasting errors.
- Open Check and Repair window and select the Fix checkbox for the typecasting error to fix the selected error.
- Perform explicit typecasting using the constructor functions, such as, for earlier example use `xsd:double($processapplication/start)` to explicitly typecast the boolean value to double.

Repair Incorrect Mappings

If you want to return to the original expected Function input and remove all of the currently specified mappings, perform the following:

Procedure

1. Delete the root element of the function’s input by selecting it and clicking the **Delete** button.

   Click the **Mapper Check and Repair** (✓) button.
2. In the Mapper Check and Repair dialog Fix column, select the checkboxes for all items.
3. Click **OK**.

Result

Alternatively, you can simply select the root input element and press the delete key on your keyboard as a shortcut for the procedure above.

After deleting all mappings and schema items and then repairing the input schema, the function's input reverts to the state before you open the Argument Mapping Wizard for the first time for this function.

Migrating Mapper Functions from XPath 1.0 to XPath 2.0

After the project is migrated from Xpath 1.0 to XPath 2.0, to fix some common mapping errors, you can use the Mapper Function Migration wizard.

This wizard fixes the typecast errors, if the data type can be converted using the xsd: constructor functions. It also fixes the issues which the gives the error: `XSLT is out of sync with schema component properties.`

Note: This wizard cannot fix all mapper errors, but it fixes the common issues when switching from XPath 1.0 to XPath 2.0.

Important: Always back up your project before running the Mapper Function Migration wizard.

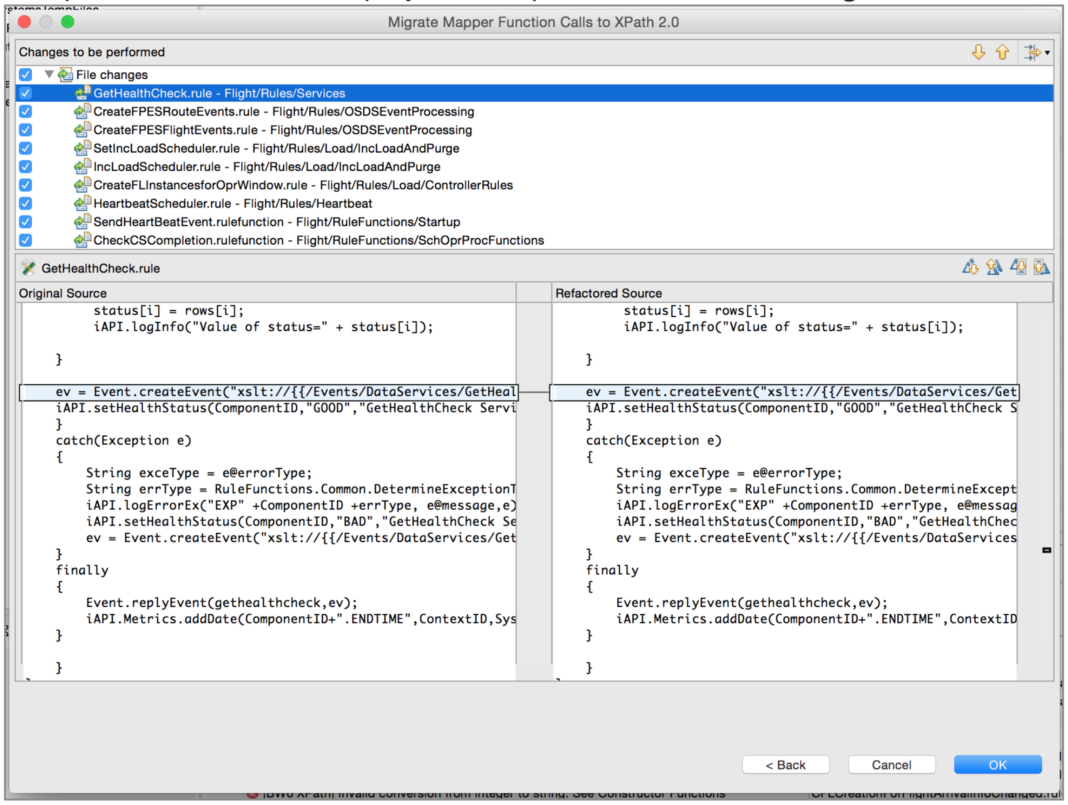
Procedure

1. In BusinessEvents Studio, right-click the project and select **Refactor > Migrate Mapper Function Calls**.

   The Migrate Mapper Function Calls to XPath 2.0 window is displayed with information on the action that is performed on the project.

2. Click **Preview** to review the changes that is done to the project.

   The preview window is displayed with preview of all the changes.



- ✓ File changes
  - ✓ GetHealthCheck.rule - Flight/Rules/Services
  - ✓ CreateFPESRouteEvents.rule - Flight/Rules/OSDSEventProcessing
  - ✓ CreateFPESFlightEvents.rule - Flight/Rules/OSDSEventProcessing
  - ✓ SetIncLoadScheduler.rule - Flight/Rules/Load/IncLoadAndPurge
  - ✓ IncLoadScheduler.rule - Flight/Rules/Load/IncLoadAndPurge
  - ✓ CreateFLInstancesforOprWindow.rule - Flight/Rules/Load/ControllerRules
  - ✓ HeartbeatScheduler.rule - Flight/Rules/Heartbeat
  - ✓ SendHeartBeatEvent.rulefunction - Flight/RuleFunctions/Startup
  - ✓ CheckCSCompletion.rulefunction - Flight/RuleFunctions/SchOprProcFunctions

GetHealthCheck.rule

Original Source

```
            status[i] = rows[i];
            iAPI.logInfo("Value of status=" + status[i]);

    }

    ev = Event.createEvent("xslt://{{/Events/DataServices/GetHeal
    iAPI.setHealthStatus(ComponentID,"GOOD","GetHealthCheck Servi
    }
    catch(Exception e)
    {
        String exceType = e@errorType;
        String errType = RuleFunctions.Common.DetermineExceptionT
        iAPI.logErrorEx("EXP" +ComponentID +errType, e@message,e)
        iAPI.setHealthStatus(ComponentID,"BAD","GetHealthCheck Se
        ev = Event.createEvent("xslt://{{/Events/DataServices/Get
    }
    finally
    {
        Event.replyEvent(gethealthcheck,ev);
        iAPI.Metrics.addDate(ComponentID+".ENDTIME",ContextID,Sys
    }

    }
```

Refactored Source

```
            status[i] = rows[i];
            iAPI.logInfo("Value of status=" + status[i]);

    }

    ev = Event.createEvent("xslt://{{/Events/DataServices/Get
    iAPI.setHealthStatus(ComponentID,"GOOD","GetHealthCheck S
    }
    catch(Exception e)
    {
        String exceType = e@errorType;
        String errType = RuleFunctions.Common.DetermineExcept
        iAPI.logErrorEx("EXP" +ComponentID +errType, e@messag
        iAPI.setHealthStatus(ComponentID,"BAD","GetHealthChec
        ev = Event.createEvent("xslt://{{/Events/DataServices
    }
    finally
    {
        Event.replyEvent(gethealthcheck,ev);
        iAPI.Metrics.addDate(ComponentID+".ENDTIME",ContextID
    }

    }
```

3. Select or clear the checkbox for each file after reviewing the changes. Click **OK** to apply the changes.

What to do next

After the wizard completes, you can revert the changes immediately using the **Edit > Undo File Changes** menu. However, after any additional changes to the resources in the workbench, you cannot use the **Undo** command.

Shortcuts

The Move In, Insert, Add Child, and Edit Statement buttons on the toolbar are ways to manually manipulate XSLT statements in the Function panel. These buttons, however, only add or modify one statement at a time. Also, there are some situations where you wish to convert a hint into a statement without performing any mapping. This section describes shortcuts for manipulating XSLT statements.

Statement Menu Options

When you select an element in the function input schema and right-click, a menu appears. The Statement menu item contains several sub-items that are useful shortcuts for creating XSLT statements.

- Surround with Choice: a shortcut for adding a choice statement and its associated conditions or otherwise statements around the currently selected element.
- Surround with If: a shortcut for adding an if statement and placing the currently selected element as the sub-element of the if.
- Surround with For-Each: a shortcut for moving the current element into a For-Each statement.
- Surround with For-Each-Group: a shortcut for moving the current element into a For-Each-Group statement and adding a Group-By grouping statement.
- Duplicate: a shortcut for creating a duplicate of the currently selected element (including any mappings or XPath formulas for the element). The duplicate is added below the currently selected element.
- Insert Model Group Content: a shortcut for inserting the contents of a selected model group into the mapper tree. The selected element in the function input schema is replaced by the contents of the model group you select.

Dragging to the Left

Dragging an element in the function input schema to the left of the divider between the two areas of the **Input** tab changes a hint into an XSLT statement.

This shortcut is useful in the following situations:

- When you have a complex element with no subelements and no content.
- When you have a choice element, dragging to the left brings up the Mapping Wizard, and you can choose a type for the element.
- When you have an element of type Any, dragging to the left brings up a dialog, and you can specify the type for the element.

Cutting and Pasting

The Function panel is an XSLT template for specifying the function’s input schema. You can choose any element in the Function panel and select Copy from the right-click menu or press the Control-C keys to copy the XSLT statement for the element. Once the XSLT is copied, you can paste it into a text editing tool to view or modify the code.

You can also paste arbitrary XSLT code into the Function panel using the right-click menu or the Control-V keys. Pasting XSLT code from the copy buffer places the code above the currently selected element in the Function panel.

Automatic Testing (at Runtime)

When you map Scope Variables elements to Function elements, the behavior of the mapping depends upon the types of elements you are mapping. In the simplest case of mapping a required element in the Scope Variables schema to a required Function element, the value of the Scope Variables element is assigned to the required Function element.

However, when elements are optional or nillable, more complex tests are necessary. When you drag the Scope Variables element to the Function element, the necessary tests are automatically placed into the Function XSLT template.

This section describes the result of mapping different types of elements. The types of mappings are described, then an example is given that illustrates these mappings and shows the XSLT code that is generated automatically when these mappings are performed at runtime.

Required to Required

Specifies that the statement should always include the required Function element and its value should be obtained from the required Scope Variables element that the element is mapped to.

Optional to Optional

Specifies that the statement should test if the Scope Variables element is present, and if so, include the optional element in the function’s input. If the Scope Variables element is not present, the optional element is omitted from the function’s input.

Nillable to Nillable

Specifies that both the Scope Variables and Function elements can be nil. Therefore, the value of the Function element is set to the value of the Scope Variables element. The value of the Function element is set explicitly to nil if that is the value of the Scope Variables element.

Optional to Nillable

Specifies that the statement should test if the optional Scope Variables element exists. If the element exists, the Function element should be created and set to the value of the Scope Variables element. If the Scope Variables element does not exist, the element is omitted from the function input schema.

Nillable to Optional

Specifies that the statement should test if the Scope Variables element has a value specified, and if so, the optional element in the Function panel should be set to the value of the Scope Variables element. Otherwise, if the Scope Variables element is nil, the optional element is omitted from the Function panel.

Optional & Nillable to Optional & Nillable

Specifies that if the optional Scope Variables element exists, then include the optional Function element in the input schema. If the Scope Variables element is nil, set the value of the Function element explicitly to nil. If the Scope Variables element is not nil, set the value of the Function element to the value of the Scope Variables element. If the Scope Variables element is not present, then omit the optional element from the function input schema.

Examples of Mappings

Some mappings require several steps to achieve the desired results. This section describes some complicated mapping scenarios and how to achieve the desired mappings using the tools available.

Note: There are many methods to insert or modify XSLT statements in the function input schema. The examples in this section illustrate the simplest procedures to obtain the desired results. However, you do not have to follow the same procedures outlined in this section to achieve the correct set of statements.

Setting an Element Explicitly to Nil

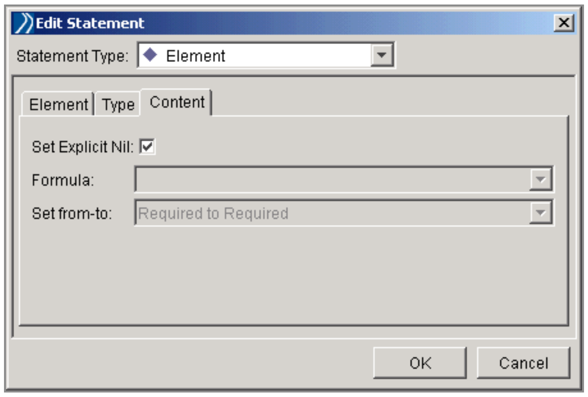
In some situations, you may wish to set an element explicitly to nil. One situation is when you wish to insert a row into a database table and you wish to supply a NULL for one of the columns.

Procedure

1. Select the input element you wish to set to nil.

   Click the **Edit Statement** icon (XPath 1.0) or **Show/Hide Edit Tab** icon (XPath 2.0) on the **Input** tab toolbar.

2. (XPath 1.0) Click the **Content** tab of the Edit Statement window and select the **Set Explicit Nil** checkbox.

   Edit Statement

   Statement Type: Element

   Element | Type | Content

   Set Explicit Nil: ☑

   Formula:

   Set from-to: Required to Required

   The element's formula becomes blank and is not editable (because nil is the value of the element) and the explicit nil qualifier icon appears next to the statement

   .

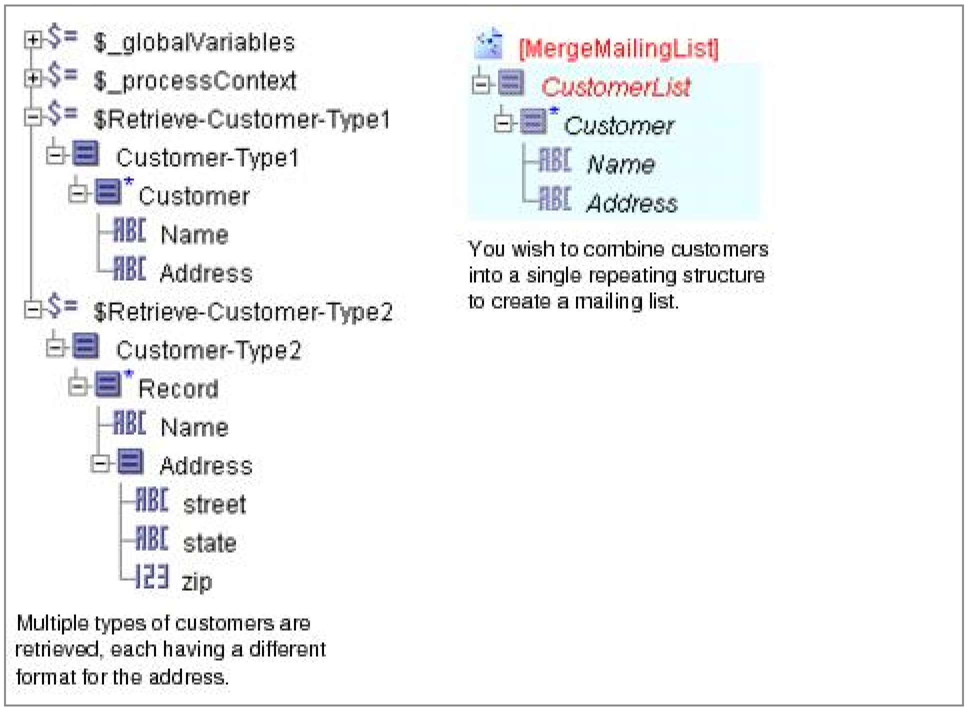
3. (XPath 2.0) Select the **Set Explicit Nil** checkbox under the **Statement Type** section under the **Edit Statement** tab.

Merging Input from Multiple Sources

You may have multiple items in the Scope Variables that you wish to map to one repeating element in the Function panel.

For example, you may have multiple formats for customer records and you wish to create a single, merged mailing list containing all customers in one format.

Figure 14: Example Schema of Merging Input From Multiple Sources

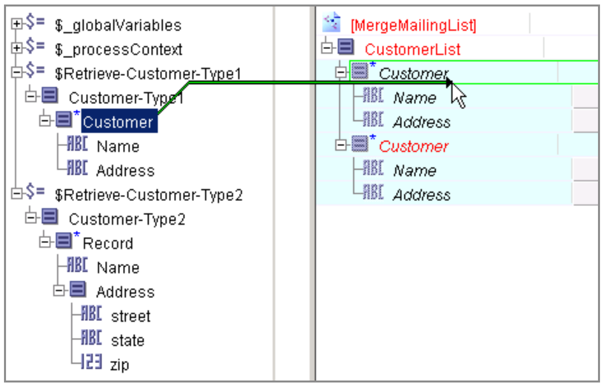


Procedure

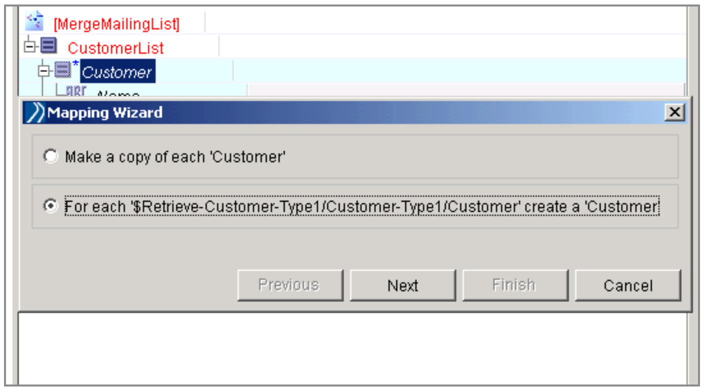
1. Select the repeating element in the Function panel, right-click, and select **Statement > Duplicate** from the menu.

   Because you are creating two different formulas for mapping, you need two copies of the repeating element, one for each format. The resulting output contains only one repeating customer element, but the two copies in the Function panel make it simpler to perform two different mappings.

2. Map one of the elements from the Scope Variables to the first copy of the repeating element in the Function. For example, map `$Retrieve-Customer-Type1/Customer` to `MergeMailingList/CustomerList/Customer`.



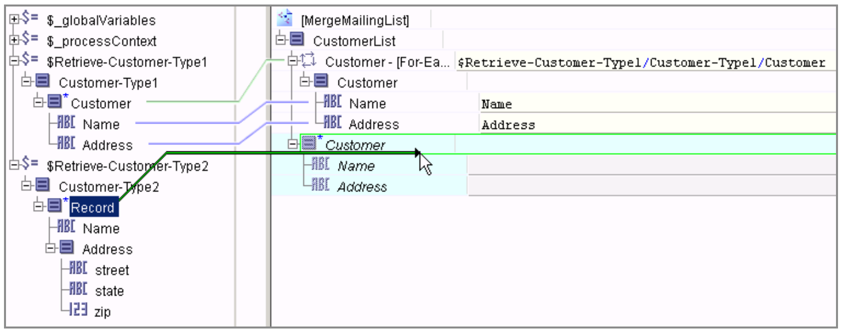
The Mapping Wizard dialog appears and presents choices for what you would like to accomplish. Choose the For Each option and click **Next**.



Make a copy of each 'Customer' [✓ For each '$Retrieve-Customer-Type1/Customer-Type1/Customer' create a 'Customer']

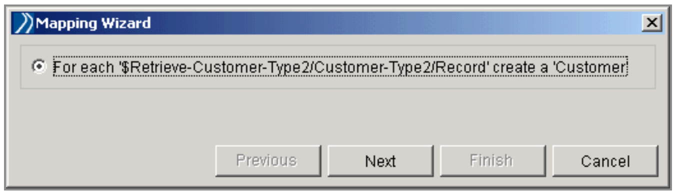
The mapping wizard asks if you wish to automatically map items with the same names. Click **Finish** to accept the default mappings.

3. Map the other element from the Scope Variables to the second copy of the repeating element in the Function. For example, map `$Retrieve-Customer-Type2/Record` to `MergeMailingList/CustomerList/Customer`.



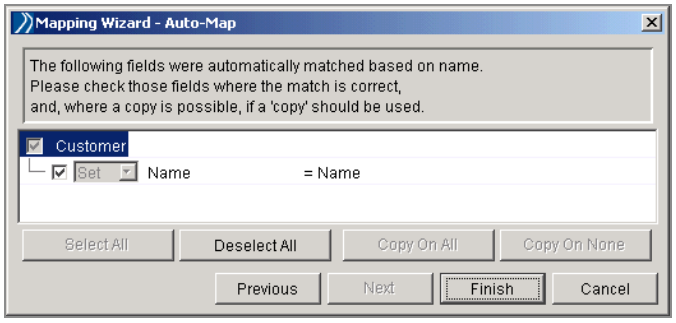
```
$Retrieve-Customer-Type1/Customer-Type1/Customer
Name
Address
```

In the Mapping Wizard dialog, choose the For Each option and click **Next**.



For each '$Retrieve-Customer-Type2/Customer-Type2/Record' create a 'Customer'

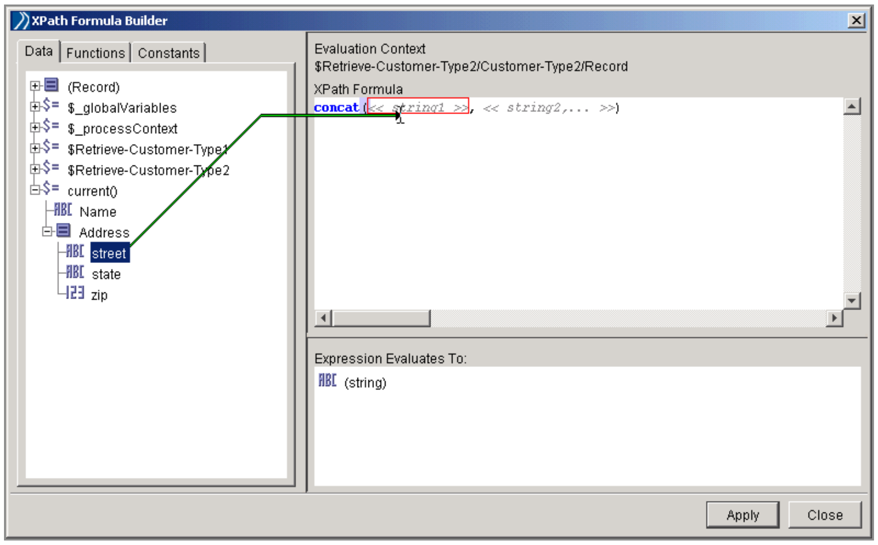
The mapping wizard presents you with an option to automatically map elements with the same name. Click **Finish** to accept the default mappings.



The following fields were automatically matched based on name.
Please check those fields where the match is correct,
and, where a copy is possible, if a 'copy' should be used.

☑ Customer
☑ Set Name = Name

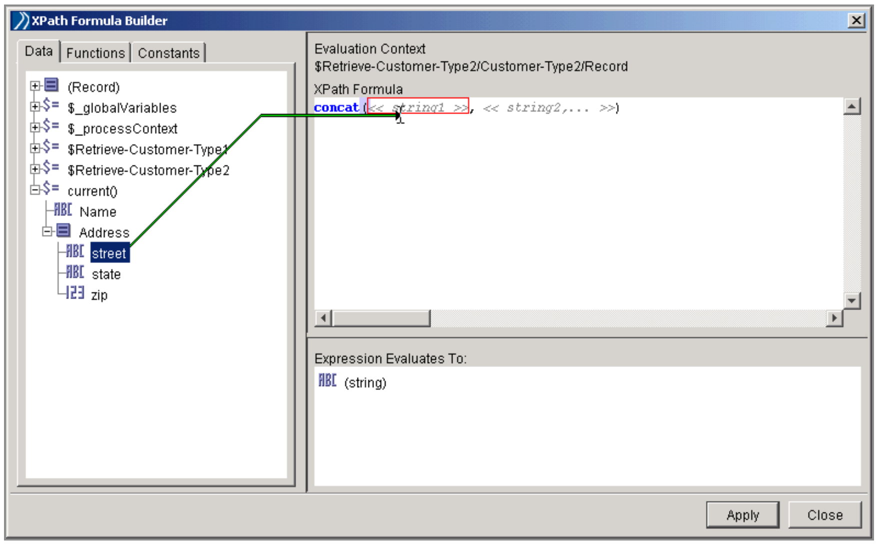
4. Select the `Address` element and click the XPath Formula Builder icon in the **Input** tab toolbar. In the XPath Formula Builder, drag a `concat()` function into the XPath Formula field. This function is used to concatenate the three elements in the `Record` element in the Scope Variables area to one `Address` element in the function's input.



Evaluation Context
$Retrieve-Customer-Type2/Customer-Type2/Record

XPath Formula
```
concat(<< string1 >>, << string2,... >>)
```

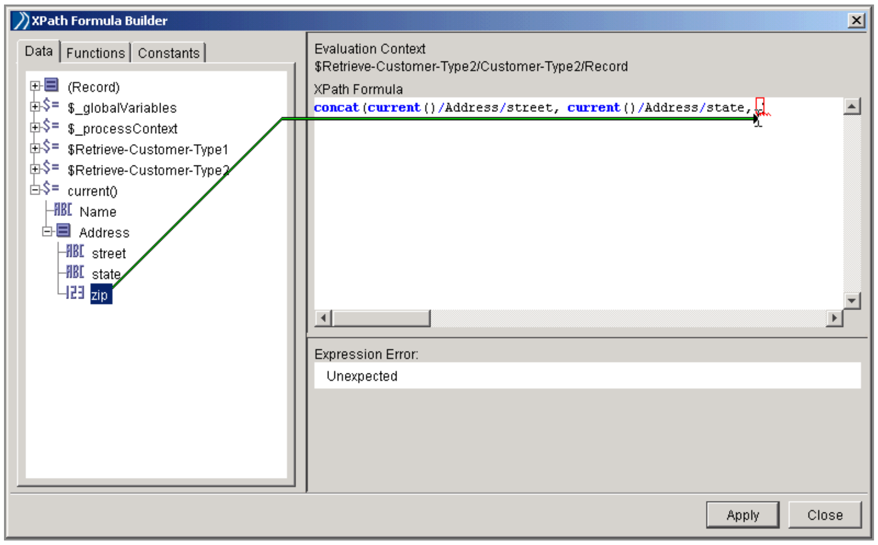
Click the **Data** tab, then drag the `$current()/Address/street` element into the `<<` *string1* `>>` placeholder in the `concat()` function.



Evaluation Context
$Retrieve-Customer-Type2/Customer-Type2/Record

XPath Formula
```
concat(<< string1 >>, << string2,... >>)
```

Drag the `$current()/Address/state` element into the `<<` *string2* `>>` placeholder in the `concat()` function. Then, add a comma to the end of the function to include a third string to concatenate. Drag the `$current()/Address/zip` element into the position of the third string in the `concat()` function.



Evaluation Context
$Retrieve-Customer-Type2/Customer-Type2/Record

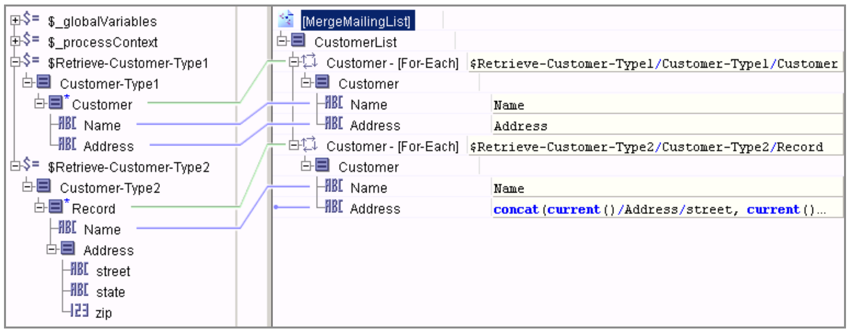
XPath Formula

```
concat(current()/Address/street, current()/Address/state,
```

Expression Error:
Unexpected

5. Click **Apply**, then click **Close** to dismiss the XPath Formula Builder dialog.

6. This results in the following mapping:



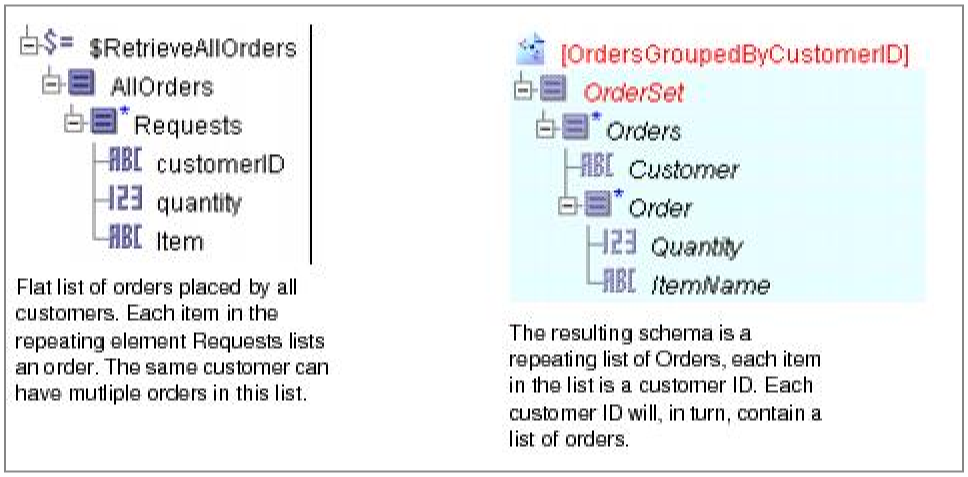
```
$Retrieve-Customer-Type1/Customer-Type1/Customer
Name
Address
$Retrieve-Customer-Type2/Customer-Type2/Record
Name
concat(current()/Address/street, current()...
```

Converting a List Into a Grouped List

You may need to convert a flat list of items into a more structured list.

For example, you may have a list of all orders that have been completed. You might want to organize that list so that you can group the orders placed by each customer.

Figure 15: Example Schema of List Conversion to a Grouped List

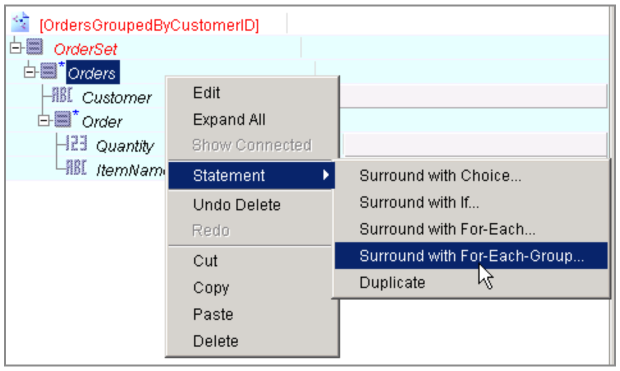


Flat list of orders placed by all customers. Each item in the repeating element Requests lists an order. The same customer can have mutliple orders in this list.

The resulting schema is a repeating list of Orders, each item in the list is a customer ID. Each customer ID will, in turn, contain a list of orders.

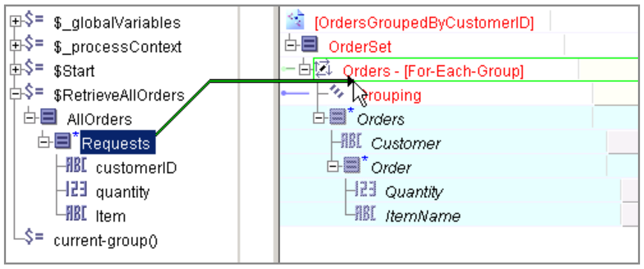
Procedure

1. Choose the repeating element in the function input schema that holds the grouped data. In this example, that element is `Orders`. Right-click on this element and choose **Statement > Surround with For-Each-Group...** from the pop-up menu. This is a shortcut to create a For-Each-Group statement with the `Orders` element as a child element and a Grouping statement to contain the element you wish to group-by.

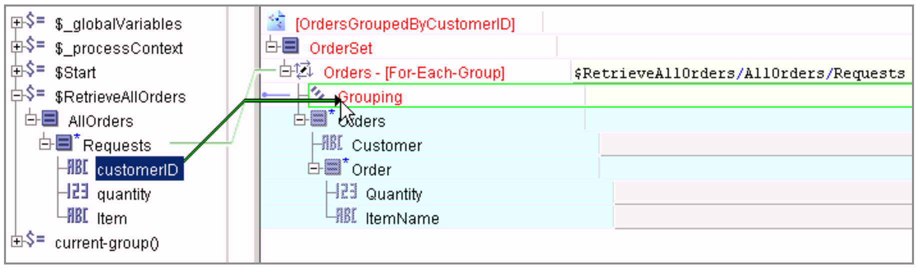


Adding the Grouping statement creates the `$=current-group()` element in the Scope Variables area. The Grouping statement creates the list grouped by the desired element, and the `current-group()` function allows you to access the items in the `Requests` repeating element that correspond to the group that is currently being processed.

2. Drag the repeating element from the Scope Variables area to the For-Each-Group statement.

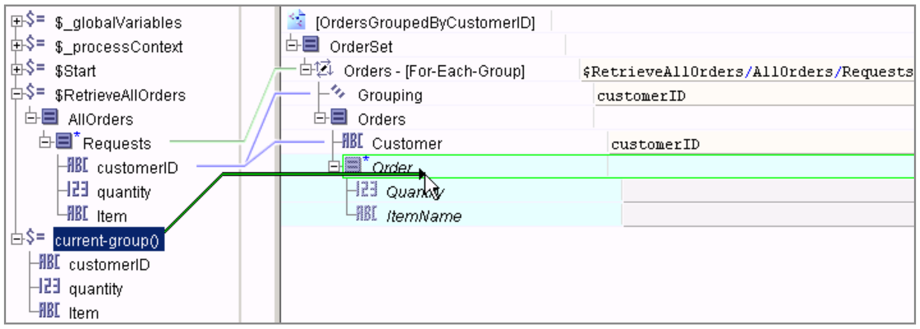


3. Drag the element you wish to group by from the Scope Variables area to the Grouping statement in the Function panel. In this example, customerID is the grouping element.



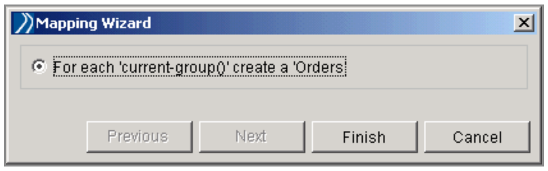
```
$RetrieveAllOrders/AllOrders/Requests
```

4. Map the `current-group()` element in the Scope Variables area to the repeating element Order under the Customer element in the Function panel.



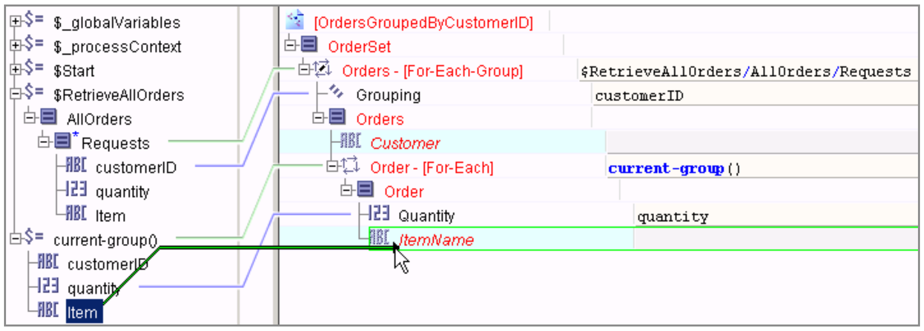
```
$_globalVariables                 [OrdersGroupedByCustomerID]
$_processContext                  OrderSet
$Start                              Orders - [For-Each-Group]   $RetrieveAllOrders/AllOrders/Requests
$RetrieveAllOrders                    Grouping                    customerID
  AllOrders                           Orders
    Requests                            Customer                  customerID
      customerID                        Order
      quantity                            Quantity
      Item                                ItemName
current-group()
  customerID
  quantity
  Item
```

The default choice in the mapping wizard for this mapping is to create a For-Each. Choose this option in the mapping wizard.



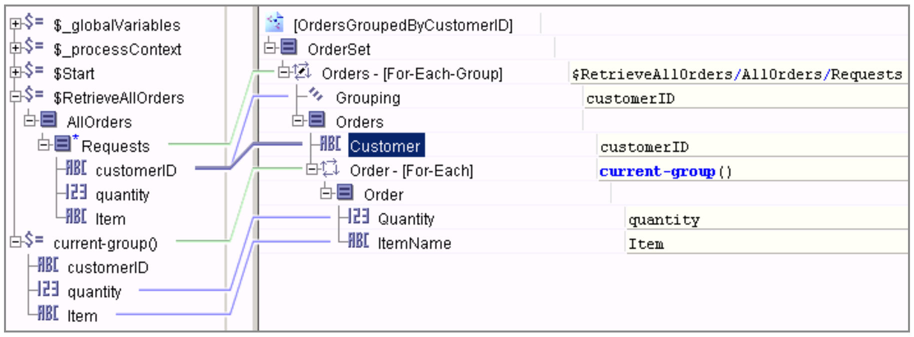
This creates an item in the `Order` list for each item in the current customer ID group that is being processed. The mapping wizard asks if you wish to map items with the same name in the current group and the orders group.

5. Map the remaining element from the `current-group()` element into the desired element in the For-Each group. In this case, `quantity` would map to `Quantity` automatically, and `Item` must be mapped to `ItemName`.



```
$_globalVariables                 [OrdersGroupedByCustomerID]
$_processContext                  OrderSet
$Start                              Orders - [For-Each-Group]   $RetrieveAllOrders/AllOrders/Requests
$RetrieveAllOrders                    Grouping                    customerID
  AllOrders                           Orders
    Requests                            Customer
      customerID                        Order - [For-Each]        current-group()
      quantity                            Order
      Item                                  Quantity              quantity
current-group()                             ItemName
  customerID
  quantity
  Item
```

6. Map the customerID element in the Requests element into the Customer element in the Function panel.



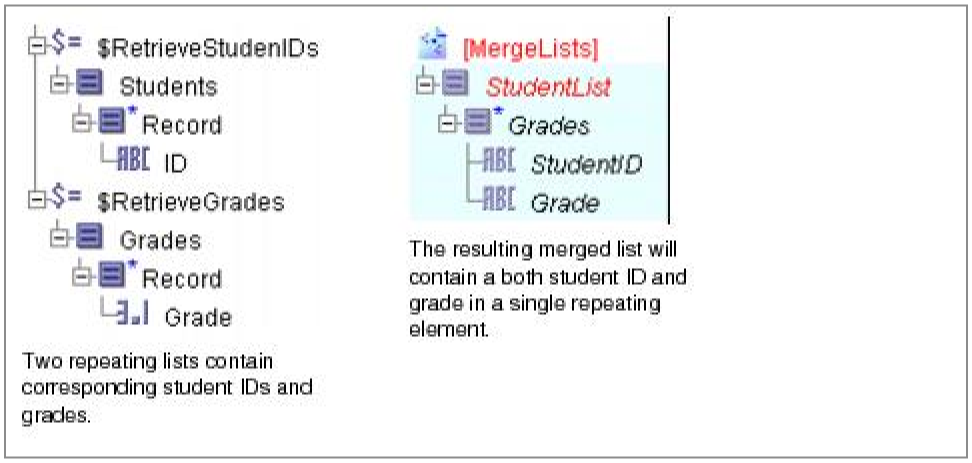
```
$RetrieveAllOrders/AllOrders/Requests
customerID
customerID
current-group()
quantity
Item
```

Merging Two Corresponding Lists

You may need to merge two lists that have corresponding items into one repeating element.

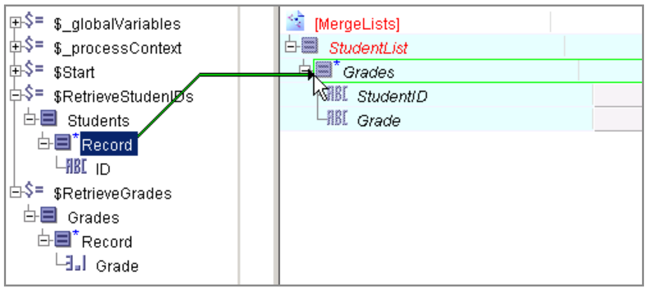
For example, you may have a list of student IDs and a list of grades, each grade corresponds to the student ID in the same position in the student ID list.

Figure 16: Example Schema of Merging of Two List

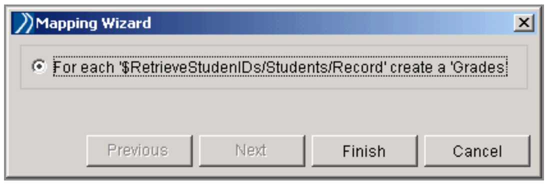

$RetrieveStudenIDs
Students
Record
ID
$RetrieveGrades
Grades
Record
Grade
Two repeating lists contain corresponding student IDs and grades.
[MergeLists]
StudentList
Grades
StudentID
Grade
The resulting merged list will contain a both student ID and grade in a single repeating element.

Procedure

1. Map the first repeating element from the Scope Variables area into the Grades repeating element in the Function panel. In this example, the `$RetrieveStudentIDs/Students/Record` is the first repeating element.

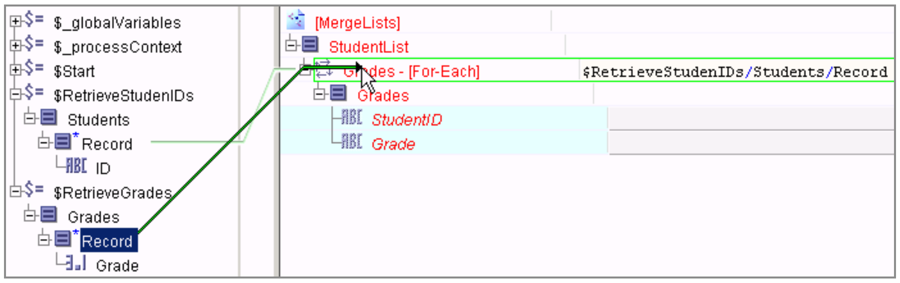


This brings up the mapping wizard with the default choice of creating a For-Each statement. Click **Finish** in the Mapping Wizard dialog to create the For-Each statement.

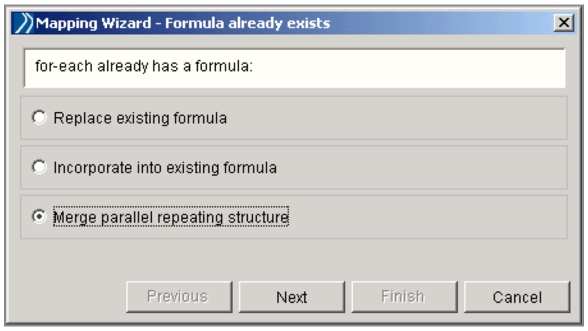

Mapping Wizard

For each '$RetrieveStudenIDs/Students/Record' create a 'Grades'

2. Drag the second repeating element into the For-Each statement.



3. The Mapping Wizard dialog appears asking you to choose an option. Choose the Merge parallel repeating structure option and click **Next**.



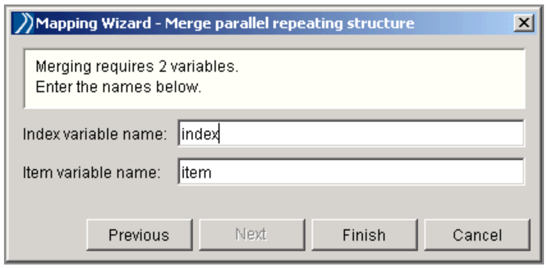
for-each already has a formula:

Replace existing formula

Incorporate into existing formula

[✓ Merge parallel repeating structure]

4. Merging two parallel repeating structures requires two variables. The mapping wizard prompts you to name these two variables. One variable is to hold the position number of the current item being processed, and the other variable is to hold the item in the second list that corresponds to the position of the item in the first list. Create the variables with the default names supplied by the mapping wizard, or choose your own names for these variables. Click **Finish** to proceed.



Merging requires 2 variables.
Enter the names below.

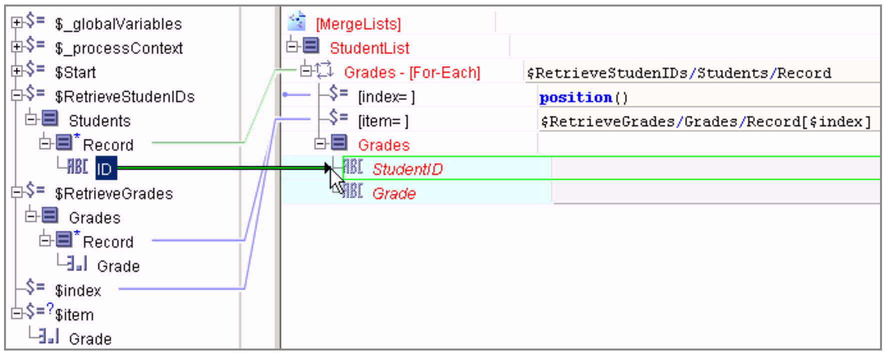
Index variable name: index

Item variable name: item

The two variables appear in the Scope Variables area once you have completed this step. The two variables also appear in the Function panel with the correct XPath statement to produce the desired result.

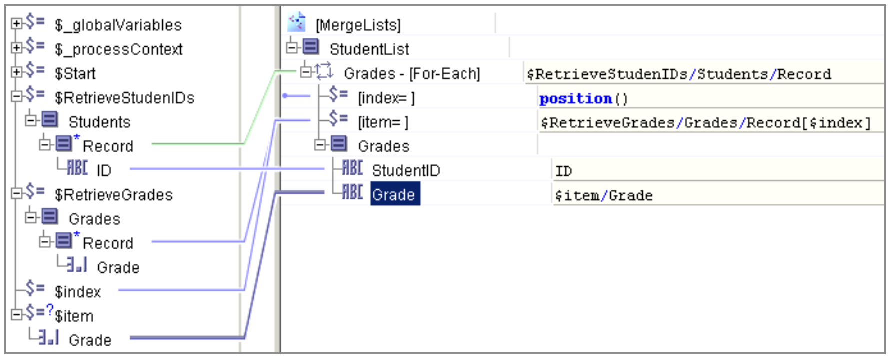
The `$=[index=]` element contains the XPath formula `position()` to set the element with the current position number of the list item being processed. The `$=[item=]` element contains a statement to retrieve the item in the second repeating element that corresponds to the position of the item in the first list that is currently being processed.

5. Map the `ID` element to the `StudentID` element in the function arguments.



```
$RetrieveStudenIDs/Students/Record
position()
$RetrieveGrades/Grades/Record[$index]
```

6. Map the `$=item/Grade` element to the `Grade` element in the Function panel.



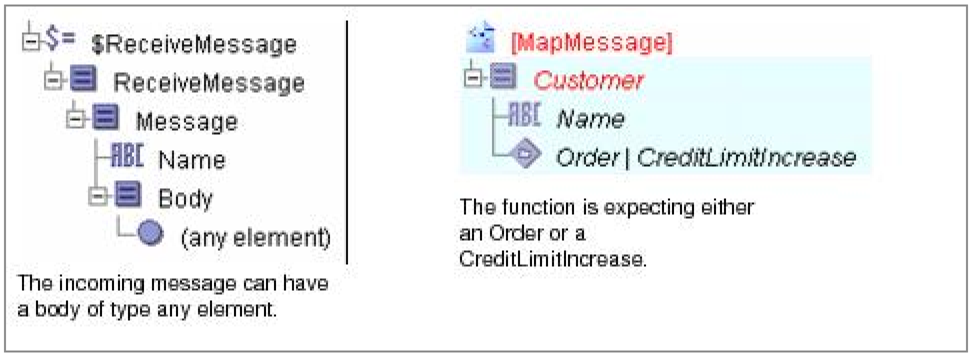
```
$RetrieveStudenIDs/Students/Record
position()
$RetrieveGrades/Grades/Record[$index]
ID
$item/Grade
```

Coercions

In some situations, the datatype of a Scope Variables element may be undefined. In these situations, you may know the data type of the element, and you can coerce the element into a specific type. Using the **Coercions** button in the **Input** tab toolbar you can create and manage your coercions.

The following example illustrates a schema with an element defined as the "any element" data type. The schema is for a generic incoming message that can have any type of body. In the example, however, the any element is coerced into an Order type so that it can be mapped to a choice element.

Figure 17: Example Schema of Coercion

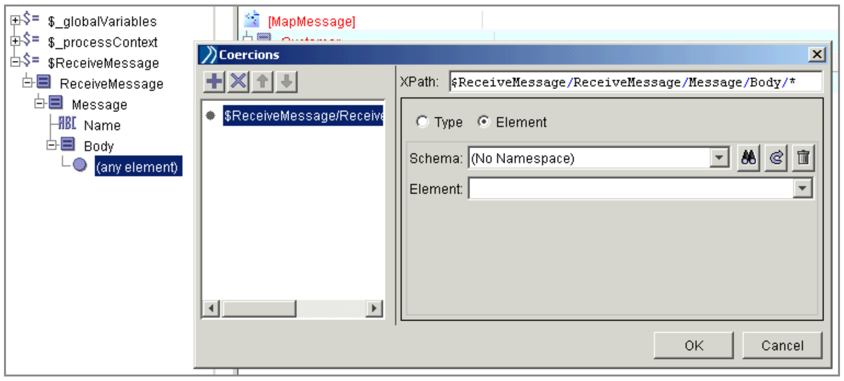

$ReceiveMessage
ReceiveMessage
Message
Name
Body
(any element)
The incoming message can have a body of type any element.
[MapMessage]
Customer
Name
Order | CreditLimitIncrease
The function is expecting either an Order or a CreditLimitIncrease.

The following procedure describes how to coerce the `Body` element of the incoming message into a specific datatype and map it to a choice element.

Note: There are many ways of accomplishing the same result as this example. This example attempts to illustrate the simplest method to achieve the desired result.

Procedure

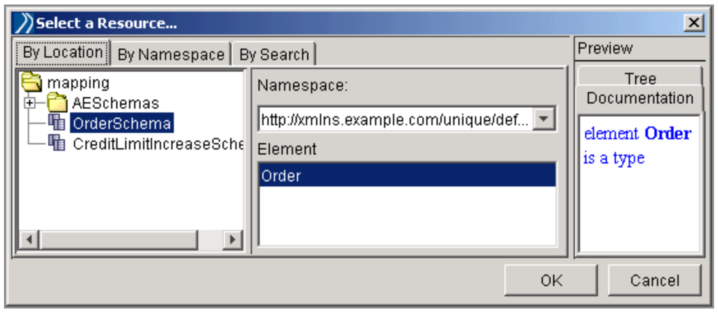
1. Select the element of type any element in the Scope Variables schema. Click the **Coercions** button in the **Input** tab toolbar. In the Coercions dialog, click the **Insert** button (+) to add a coercion for the currently selected element.



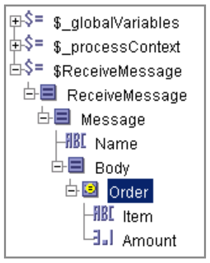
The Coercions dialog allows you to manage all of your coercions for a function in one dialog. You can create, modify, or delete coercions for any element in the Scope Variables schema using this dialog, not just the currently selected element. If you are creating a coercion for an element that is not currently selected, use the XPath field to specify the location of the element.

Click the **Element** radio button to specify that you are specifying a schema element.

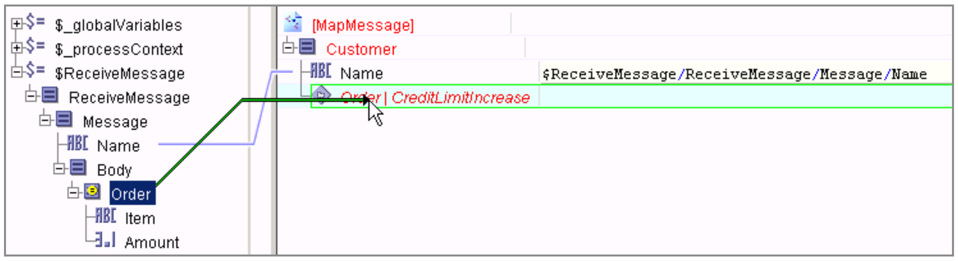
2. Click the Browse Resources button next to the Schema field to browse a list of schemas that can be used. In the Select a Resource... dialog, select the schema that you would like to specify



Click OK to coerce the element into the data type of the selected schema element. The following would be the resulting schema where the element of the datatype any element has been replaced with the `Order` schema.

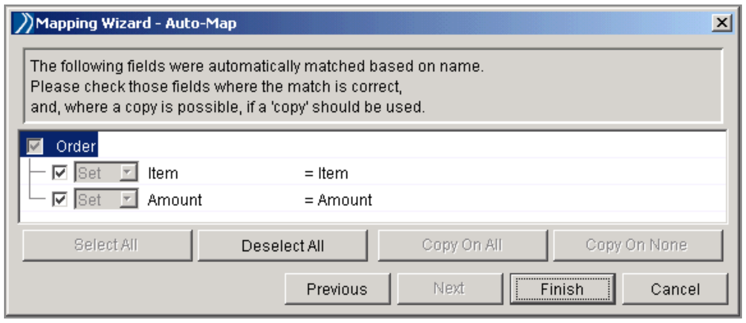


3. Map the `Name` element to the `Name` element in the Function panel. Then, map the coerced `Order` element to the choice element in the Function panel.



$ReceiveMessage/ReceiveMessage/Message/Name

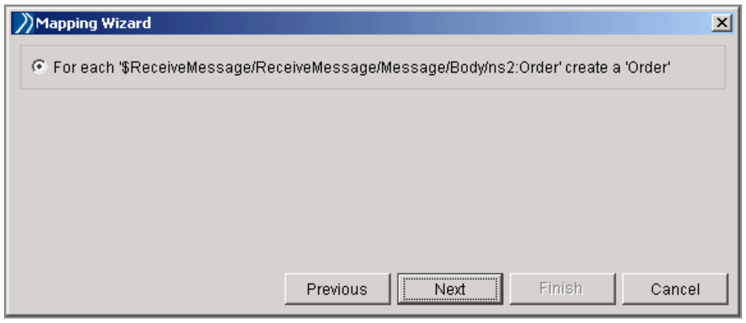
The Mapping Wizard dialog appears and asks if you wish to create an `Order` or a `CreditLimitIncrease` element. Select `Order` and click **Next**.



The following fields were automatically matched based on name.
Please check those fields where the match is correct,
and, where a copy is possible, if a 'copy' should be used.

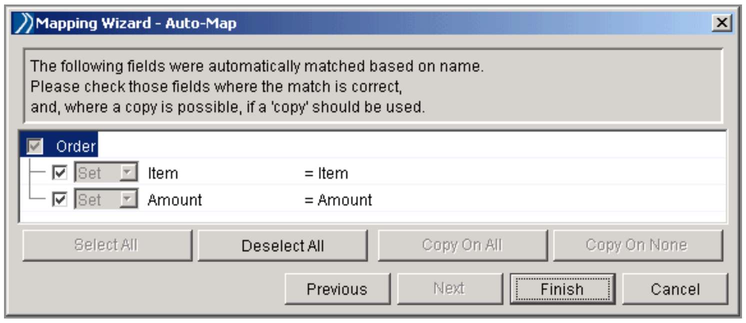
[✓] Order
[✓] Set Item = Item
[✓] Set Amount = Amount

The Mapping Wizard then asks you to create a For Each. Even though there is only one element in the Scope Variables schema (the `Message` element is not repeating), a For Each is used because this construct allows you to map the individual items of the Order element. Click **Next** to continue.



[●] For each '$ReceiveMessage/ReceiveMessage/Message/Body/ns2:Order' create a 'Order'

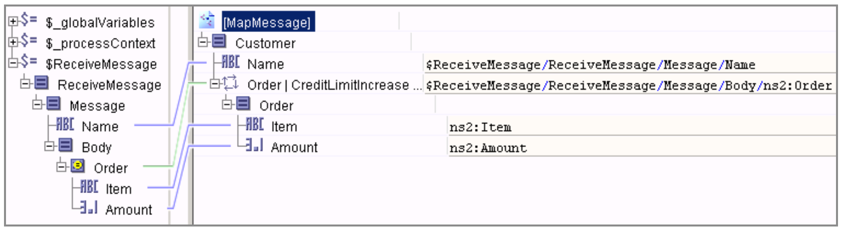
The Mapping Wizard then asks if you wish to automatically map elements with the same name. Click **Finish** to accept the default mappings.


Mapping Wizard - Auto-Map

The following fields were automatically matched based on name.
Please check those fields where the match is correct,
and, where a copy is possible, if a 'copy' should be used.

☑ Order		
☑ Set	Item	= Item
☑ Set	Amount	= Amount

4. The following is the completed mapping.



Name	$ReceiveMessage/ReceiveMessage/Message/Name
Order | CreditLimitIncrease ...	$ReceiveMessage/ReceiveMessage/Message/Body/ns2:Order
Item	ns2:Item
Amount	ns2:Amount

XSLT Statements

Using the XSLT statement you can set up the transformation of mapping.

The following sections describe the XSLT statements that you can add to your mapping. You can add or edit these statements by clicking the **Edit Statement** icon (XPath 1.0) or **Show/Hide Edit Tab** icon (XPath 2.0). These statements can be added automatically by selecting them from the dialogs that appear when you drag and drop elements from the Scope Variables tree to the function argument tree.

The following sections discuss statement types (available in the Statement Type dropdown list in the Edit Statement dialog).

Attribute

Allows you to specify an attribute, and optionally the namespace for the attribute. You can also specify the type of value for the attribute.

XSLT Equivalent

The following is an attribute named "lastname".

```
<ns:attribute namespace="mns" name="lastName"/>
```

When attributes are created, you can optionally specify the kind of value the attribute has and whether the attribute should be surrounded by an if statement. For example, you can specify the value of the last name attribute to be a constant, like so:

```
<ns:attribute namespace="mns" name="lastName"/>
   "Smith"
</ns:attribute>
```

Choose

Provides a way to select transformation to perform based on an expression. Specify the condition in the `when` element as an XPath expression. You can optionally specify an `otherwise` condition for processing all elements that do not meet any of the specified `when` conditions.

XSLT Equivalent

The following determines if the node set for FilesTransferred contains any files, and if so, performs an action. If the node set is empty (no files were transferred), a different action is performed.

```
<ns0:choose xmlns:ns0="http://www.w3.org/1999/XSL/Transform">
   <ns0:when test="$FTP-Put/FTPPutOutputFile/FileTransferred" >
      < something here ... >
   </ns:0when>
   <ns0:otherwise>
      < something here ...>
   </ns0:otherwise>
</ns0:choose>
```

Comment

Places a comment in the XSLT template. Comments are delimited by `<!--` and `-->`.

XSLT Equivalent

```
<!-- comment here -->
```

Copy

Copies the selected node to the current node in the input tree. Only the node is copied, no children of the node are copied.

XSLT Equivalent

```
<ns0:copy xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
select="$Query/resultSet"/>
```

Copy-Contents-Of

Copies the selected node’s contents. This is useful if you wish to copy an element to a new element with a different name.

XSLT Equivalent

```
<ns:element namespace="foo" name="bar">
   <ns:copy-of select="null/@*"/>
   <ns:copy-of select="null/node()"/>
</ns:element>
```

Copy-Of

Creates a copy of the selected node, including the node’s children. Both the copied node and the destination node must have the same name and structure.

XLST Equivalent

```
<ns0:copy-of xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
select=""/>
```

Element

Creates an element with the specified name.

XSLT Equivalent

```
<elementName>value</elementName>
```

For-Each

Performs the specified statements once for each item in the selected node. This is useful if you wish to process each item of a repeating element once.

XSLT Equivalent

The following iterates over the list of files transferred from a ActiveMatrix BusinessWorks FTP Put activity and outputs an element with the name of each file for each file transferred.

```
<ns:for-each select="$FTP-Put/FTPPutOutputFile/FileTransferred">
   <fileName>
      <ns:value-of select="$FTP-
Put/FTPPutOutputFile/FileTransferred/Name"/>
   </fileName>
</ns:for-each>
```

For-Each-Group

Groups the items in a list by a specified element. This statement requires a Grouping statement to specify which element to group-by.

See Converting a List Into a Grouped List for an example of using the For-Each-Group statement.

XSLT Equivalent

```
<ns0:for-each-group xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
select=""/>
```

Generate Comment

Places a comment element into the XSLT template. This comment is generated into the function’s output.

Comment elements have the following syntax:

```
<ns0:comment xmlns:ns0="http://www.w3.org/1999/XSL/Transform"/>
```

Generate PI

Places a processing instruction into the XSLT template.

XSLT Equivalent

```
<ns0:processing-instruction
xmlns:ns0="http://www.w3.org/1999/XSL/Transform" name=""/>
```

If

An if statement is used to surround other statements in an XSLT template to perform conditional processing. If the test attribute evaluates to true, the statements in the if are output, otherwise they are not output.

XSLT Equivalent

The following if statement surrounds an attribute for processing order items.

```
<ns:if xmlns:ns="http://www.w3.org/1999/XSL/Transform" test="not
(position()=last())">
    <ns:attribute name="OrderItem">
        <ns:value-of select=
  "$GetOrderInformation/OrderInformation/OrderDetails/OrderItem"/>
    </ns:attribute>
</ns:if>
```

Value-Of

Specifies a value-of statement. This is normally done implicitly by specifying the formula for an element (field) in the mapping, but you may insert this statement explicitly.

XSLT Equivalent

```
<ns:value-of xmlns:ns="http://www.w3.org/1999/XSL/Transform" select=""/>
```

Variable

Adds a local variable for use in the current mapping. You can specify the name of the variable and whether you wish the variable to have a select attribute.

When you add a local variable, it appears in the Function and Scope Variables panels. You can supply any XPath expression to the new variable in the Function panel (either through mapping or through the XPath Formula Builder).

Once the variable’s contents have been supplied, the variable (in the Scope Variables area) can be mapped to any item.

Adding a variable is useful when you wish to join two repeating elements into a single list, then map the combined list to an item. Adding a variable is also useful if you perform the same computation repeatedly. You can map the results of the computation to several items instead of recreating the computation for each item.

Variables can also improve performance of mappings for large data structures. For example, if you have a process variable with 40 sub-elements, and you map each of the sub-elements to a corresponding input item, TIBCO BusinessEvents must retrieve the current process variable for each XPath expression, in this case 40 times. If this mapping appears in a loop, the retrieval of the current process variable occurs 40 times per iteration of the loop. With a variable, the data is retrieved only once and used for all mappings containing the variable. Therefore, to improve performance, create a local variable to hold process variables with a large number of elements and use the local variable in XPath expressions instead of the process variable.

XSLT Equivalent

```
<ns0:variable xmlns:ns0="http://www.w3.org/1999/XSL/Transform"
name="var" select="$RetrieveResults/resultSet"/>
```

XPath Formula Builder For XPath 1.0

TIBCO BusinessEvents uses XPath in the XPath Formula Builder, available in the Function Argument Mapper tool. You can use XPath, for example, when defining payloads for events TIBCO BusinessEvents also uses XPath as the language for defining conditions and transformations.

In XPath 1.0 , the XPath Formula Builder opens in a separate window; however, in XPath 2.0, the XPath Formula Builder is part of the Function Argument Mapping wizard. They both function in the same manner with only difference in user interface. See Function Argument Mapping Wizard for XPath 2.0 for more details about XPath Formula Builder for XPath 2.0.

XPath (XML Path Language) is an expression language developed by the World Wide Web Consortium (W3C) for addressing parts of XML documents. XPath also has basic manipulation functions for strings, numbers, and Boolean values.

To use XPath in TIBCO BusinessEvents, you need only be familiar with the basic XPath concepts, but you may wish to learn more about XPath when building complex expressions.

For a complete description of XPath, refer to the XPath specification (which can be obtained from www.w3.org).

TIBCO BusinessEvents uses XPath (XML Path Language) to identify elements whose content may be used, for, example in an event payload. You can also use XPath to perform basic manipulation and comparison of strings, numbers, and Boolean values.

Addressing Schema Elements

All Scope Variables and Function arguments are represented as an XML schema. Regardless of where the data comes from or its format, TIBCO BusinessEvents represents the data as a schema tree. The data can be simple (strings, numbers, Boolean values, and so on), or it can be a complex element. Complex elements are structures that contain other schema elements, either simple elements or other complex elements. Both simple and complex elements can also repeat. That is, they can be lists that store more than one element of the given type.

XPath is used to specify which schema element you would like to refer to. Each schema has its own associated structure, for example, a set of simple values or simple data and other complex data.

To reference a particular data item in a schema, you start with the root node and then use slashes (/) to indicate a path to the desired data element. For example, if you wish to specify the Street attribute in the `ShipName` complex element that is in the `GetOrderInformation` node, you would use the following syntax:

```
$GetOrderInformation/ShipName/Street
```

The path starts with a dollar sign to indicate it begins with a root node, then continues with node names using slashes, like a file or directory structure, until the desired location is named.

Evaluation Context

XPath also has a method for referencing relative paths from a particular node. If you have an *evaluation context*, or a particular starting node in a schema tree, you can specify the relative path to other elements in the tree.

For example, if your evaluation context is `$GetOrderInformation/ShipName`, then you can reference the sub-items of `ShipName` without specifying the entire path. If you wish to reference `$GetOrderInformation/RequiredDate`, the relative path would be `../RequiredDate`. The path is relative to the evaluation context — `RequiredDate` is one level higher in the schema tree than the elements of `ShipName`.

Namespaces

Some schema elements must be prefixed with their namespace. The namespace is automatically added to elements that require this when creating mappings or when dragging and dropping data in the XPath formula builder.

Search Predicates

An XPath expression can have a search predicate. The search predicate is used to locate a specific element of a repeating schema item. For example, a `$GetOrderInformation/OrderDetails/OrderItem` item is a repeating element. If you wish to select only the first item in the repeating element, you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[1]
```

The [1] specifies the first element of a repeating item.

Sub-items can also be examined and used in a search predicate. For example, to select the element whose `ProductId` is equal to "3A54", you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[ProductId="3A54"]
```

You can also use functions and expressions in the search predicate. For example, if you wish to find all elements after the first, you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[position()>1]
```

See the **Functions** tab of the XPath Formula Builder for a list of available functions available and online documentation.

Testing for Nil

Some elements can be explicitly set to nil. You can test an element to determine if it is set to nil or not. For example, the following XPath expression returns true if the `$Order/Item/OnSale` element is set to nil:

```
$Order/Item/OnSale/@xsi:nil="true"
```

Comments

You can add comments to XPath expressions using the XPath 2.0 syntax for comments. The syntax is:

```
{-- <comment here> --}
```

For example, the following XPath expression contains a comment:

```
$GetOrderInformation/ShipName/Street {-- returns the street --}
```

The XPath Formula Builder

You access the XPath Formula Builder using a button in the Function Argument Mapping Wizard. First select an item in the Function Argument panel (in the Input section). Then click the **XPath Formula Builder** button ().

The XPath formula builder allows you to drag and drop schema elements and XPath functions to create XPath expressions. The schema elements, when dragged into the XPath Formula field, automatically become valid XPath location paths for the desired item. If a function is dragged into the XPath formula window, there are placeholders for each parameter of the function. You can drag and drop schema elements over the parameter placeholders to replace each placeholder.

The XPath Formula Builder describes the different areas of the XPath formula builder.

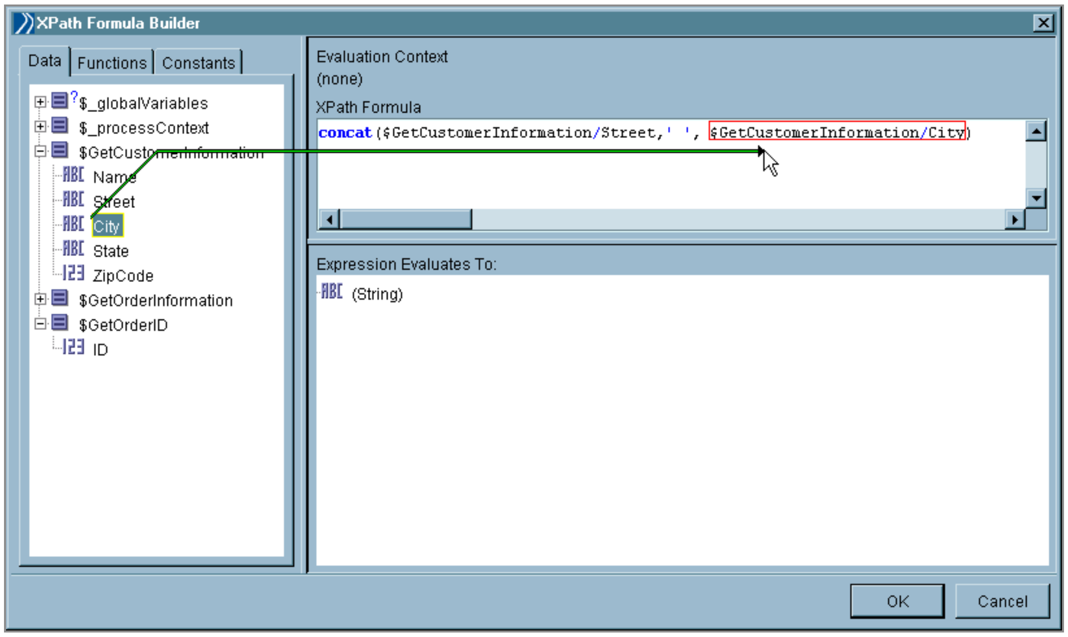
XPath Formula Builder Reference

Element	Description
Data tab	Displays the Scope Variables schema tree. All elements in this tree are available to drag and drop into the XPath Formula field.
Functions tab	Displays the available XPath functions. These are categorized into groups and each function can be dragged from the function list into the XPath Formula field. When the function is placed into the XPath formula, placeholders are displayed for the function's parameters. You can drag and drop schema elements from the **Data** tab into the function's placeholders. The result of evaluating the function is displayed in the "Expression Evaluates To" panel. If there are any errors in the expression, they are listed there as well. For more information about XPath functions, see the description of the function that is displayed when it is selected in the XPath formula builder.
Constants tab	Displays the constants available for use in XPath expressions. These are categorized into groups and each constant can be dragged from the constants list into the XPath Formula field. Constants are useful for inserting special characters, such as tabs, symbols, and so on, into XPath formulas. Constants are also defined for commonly used items, such as date formats.
Documentation panel	Appears below the Functions and Constants tabs. Describes each selected function. As you click on a function in the **Function** tab, the documentation panel gives a brief description of the function and one or more examples. Similarly documentation for constants in the Constants tab appears.
Evaluation	Displays the evaluation context of the expression field that the editor was

Element	Description
Context field	invoked from. See Evaluation Context for more information about the evaluation context.
XPath Formula field	Displays the XPath formula you wish to create. You can drag and drop items from the **Data** tab or the **Functions** tab to create the formula.
Expression Evaluates To Panel	Displays the result of evaluating the formula shown in the XPath Formula field. If there are errors in the formula, they are displayed here.

Creating an XPath formula illustrates using the XPath formula builder to create a valid function. The function concatenates the data elements `$GetCustomerInformation/Street` and `$GetCustomerInformation/City` and places a space between the two elements.

Creating an XPath formula



```
concat($GetCustomerInformation/Street,' ', $GetCustomerInformation/City)
```

String Representations of Datatypes

When data must be represented in the input or output of an activity, the data is represented as a string. This section explains the string representations of various datatypes. TIBCO BusinessEvents follows the XPath 1.0 standard for representing all numeric datatypes. TIBCO BusinessEvents follows the XML Schema canonical format for all other datatypes.

Numeric Datatypes

Numeric datatypes include all types derived from `xs:integer`, `xs:decimal`, `xs:float`, and `xs:double`.

All decimal, float, and double numbers are compressed to an integer when represented, if there are only zeros following the decimal point (for example, "`1.000`" is represented as `1`). Scientific notation is never used to represent a floating point number as a string (for example, "`xs:double('1.234E05')`" is represented as `123400`). Data is truncated if the number of digits exceeds the maximum precision for the datatype (for example, "`xs:float ('1.23456789')`" is represented as `1.2345679`).

Both zero and negative zero are represented as `0`. Positive and negative infinity are represented as `Infinity` and `-Infinity`. Not a number is represented as `NaN`.

Boolean

The boolean datatype is used to indicate a true or false state.

`xs:boolean('true')` and `xs:boolean('1')` are represented by true. The XPath function `true()` is also represented as `true`.

`xs:boolean('false')` and `xs:boolean('0')` are represented by `false`. The XPath function `false()` is also represented as `false`.

Date Datatypes

TIBCO BusinessEvents Function Argument Wizard (also known as the function argument mapper) implements dates in one of two ways. Either a date is stored as the number of milliseconds since January 1, 1970, or the date is implemented according to the XPath 2.0 or XQuery 1.0 standards as a set of normalized components (`xs:date`, `xs:time`, `xs:dateTime`, and so on) with an optional time zone offset. Activities that are associated with Java (for example, Java Code, Java Method, and so on) use the first implementation. Activities that are associated with XML (for example, Mapper, Parse XML, and so on) use the second implementation. The second implementation supports arbitrary precision of the seconds component.

Conversion between these representations may result in a loss of information either because of the difference in time zone representation or the precision of the seconds.

Date and Time Functions

There are some functions in the XPath formula builder using which you can parse or format strings that represent dates and times. These functions are:

- `format-dateTime(`*`format`*`,`*`dateTime`*`)`
- `format-date(`*`format`*`,`*`date`*`)`
- `format-time(`*`format`*`,`*`time`*`)`
- `parse-dateTime(`*`format`*`,`*`string`*`)`
- `parse-date(`*`format`*`,`*`string`*`)`
- `parse-time(`*`format`*`,`*`string`*`)`

The *`format`* parameter of these functions is based on the format patterns available for the `java.text.SimpleDateFormat` Java class. In the format parameter, unquoted alphabetic characters from `A` to `Z` and `a` to `z` represent the components of the date or time string. You can include non-pattern alphabetic characters in the string by quoting the text with single quotes. To include a single quote, use `''`. The following table describes the alphabetic characters and their associated presentation in a date or time string.

Formatting characters in date or time strings

Letter	Description	Example
`G`	Era Four or more `G`s return the full name of the era.	AD
`y`	year Two `y`s return two-digit year.	2003; 03
`M`	Month in year Three or more `M`s return text name.	August; Aug; 08
`w`	Week in year	48

Letter	Description	Example
`W`	Week in month	3
`D`	Day in year	254
`d`	Day in month	28
`F`	Day of week in month	3
`E`	Day in week Four or more `E`s return the full name of the weekday.	Friday; Fri
`a`	AM/PM marker Four or more `a`s return the full name.	AM
`H`	Hour in day (0-23)	23
`k`	Hour in day (1-24)	1
`K`	Hour in AM/PM (0-11)	11
`h`	Hour in AM/PM (1-12)	1
`m`	Minute in hour	59

Letter	Description	Example
`s`	Second in minute	48
`S`	Millisecond	456
`z`	Time zone represented as a GMT offset.	GMT-08:00
`Z`	RFC 822 four-digit time zone format	-0800
all other letters	Reserved	

For any format pattern letter that returns a numeric value (for example, `w`, `h`, and `m`), the number of letters in the format pattern represents the minimum number of digits. For formatting functions, if the date or time has fewer digits than the number of pattern letters, the output is padded with zeros. For parsing functions, when the date or time has fewer digits than the number of characters in the format pattern, the extra characters are ignored, unless they are needed to determine the boundaries of adjacent fields.

The following table illustrates some example date and time format patterns and the resulting string.

Example date and time format patterns

Date/Time Pattern	Result
`"yyy.MM.dd G ’at’ HH:mm:ss"`	`2003.3.11 AD at 09:43:56`
`"EEE, MMM d, ’’yy"`	`Tue, Mar 11, ’03`

Date/Time Pattern	Result
`"hh ’o’’clock’ a, zzzz"`	`9 o’clock AM, GMT-8:00`
`"K:mm a"`	`0:08 PM`
`"yyMMddHHmmssZ"`	`010704120856-700`

Cluster Deployment Descriptor

The Cluster Deployment Descriptor (CDD) is an XML file used to configure a project for deployment.

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.

The CDD file is explained in detail in *TIBCO BusinessEvents Configuration Guide*.

TIBCO ActiveMatrix BusinessWorks 6 Integration

With the use of the TIBCO BusinessEvents API, you can run a TIBCO BusinessEvents engine from within BusinessWorks 6 to call rule functions, create events and concepts and assert events and concepts.

See TIBCO ActiveMatrix BusinessWorks documentation for more details about BusinessWorks 6.

Note: For more information on BusinessEvents API, see online Java API Reference and MissManners and RuleFunctionAPI examples under `BE_HOME/examples/standard` to get familiar with the API.

Procedure

1. Install TIBCO BusinessEvents and TIBCO ActiveMatrix BusinessWorks 6.
2. Import the TIBCO BusinessEvents project RuleFunctionAPI/RFAPI from the standard examples in TIBCO BusinessEvents Studio.
    - Create an EAR for the same project or
    - Create a new TIBCO BusinessEvents project with Rulefunction and create and EAR for it.
3. Create a JAVA class to call the TIBCO BusinessEvents Rulefunction API.
    a. Create a JAR file from the same JAVA class.
4. Create a TIBCO ActiveMatrix BusinessWorks 6 application with the JAVA INVOKE activity with the TIBCO ActiveMatrix BusinessWorks 6 Studio.

5. Add the JAR file and the required JAR files from TIBCO `BE_HOME/lib` as listed below to add to TIBCO ActiveMatrix BusinessWorks 6 application's build path libraries.

   The following are the TIBCO BusinessEvents Library that are required:

   - `be-functions.jar`
   - `cep-common.jar`
   - `cep-kernel.jar`
   - `cep-base.jar`
   - `commons-lang3-3.2.jar`
   - `cep-ui-rt-common.jar`
   - `org.eclipse.uml2.uml.jar`
   - `org.eclipse.uml2.types.jar`
   - `TIBCOrt.jar`
   - `javassist.jar`
   - `antlr-3.2.jar`

6. Configure the JAVA Invoke Activity by using the invoke method from the *CallRuleFunction* class to call TIBCO BusinessEvents RuleFunction API.

7. Provide the required JAVA activity input parameters, such as REPO URL, TIBCO BusinessEvents engine TRA file, CDD file, PU, Rulefunction name and so on.

   These parameters are TIBCO BusinessEvents project artifacts required to call TIBCO BusinessEvents RuleFunction.

8. Go to the TIBCO ActiveMatrix BusinessWorks 6 Application Module Dependencies and import the packages as listed below required to call the TIBCO BusinessEvents RuleFunction API.

   Packages required from TIBCO ActiveMatrix BusinessWorks 6 include:

   - `org.eclipse.emf.ecore.resource.Resource$Factory$Registry`
   - `org.eclipse.emf.ecore.EObject`
   - `org.eclipse.emf.common`
   - `org.eclipse.emf.common.util.URI`
   - `org.eclipse.emf.ecore.resource.impl.ResourceSetImpl`
   - `org.eclipse.emf.ecore.impl.EPackageImpl`
   - `org.eclipse.emf.ecore.xml.type.XMLTypePackage`
   - `org.eclipse.emf.ecore.xmi.impl.EcoreResourceFactoryImpl`
   - `org.eclipse.emf.ecore.util.FeatureMap`
   - `org.eclipse.emf.common.notify.Notification`
   - `org.apache.log4j.Appender`
   - `org.apache.log4j.spi.LoggerRepository`
   - `com.tibco.xml.tns.impl.TargetNamespaceCache`
   - `com.tibco.util.StringUtilities`
   - `com.tibco.xml.schema.SmType`
   - `org.eclipse.emf.ecore.xmi.XMLOptions`
   - `com.tibco.xml.schema.impl.SmNamespaceProviderImpl`
   - `com.tibco.xml.ws.wsdl.WsException`
   - `com.tibco.xml.tns`
   - `com.tibco.xml.tns.impl`
   - `com.tibco.xml.tns.parse`
   - `com.tibco.xml.tns.parse.helpers`

- `com.tibco.xml.data.primitive`
- `com.tibco.xml.datamodel`
- `com.tibco.io.xml`
- `com.tibco.xml.schema.channel.SchemaModelProvider`
- `com.tibco.xml.validation.kernel.DefaultSchemaCache`
- `com.tibco.xml.channel.error.helpers.ErrorThrower`
- `com.tibco.xml.schema.parse.SmParseSupport`
- `com.tibco.xml.schema.build.MutableSchema`
- `com.tibco.xml.datamodel.helpers.XiChild`
- `com.tibco.xml.schema.helpers.NoNamespace`
- `com.tibco.xml.schema.flavor.XSDL`
- `com.tibco.sax.ResolverUtilities`
- `com.tibco.xml.data.primitive.values.XsBoolean`
- `com.tibco.xml.datamodel.navigation`
- `com.tibco.xml.xquery`

9. Deploy the application in TIBCO ActiveMatrix BusinessWorks 6 as "BW Application" or export the application to EAR and deploy in the TIBCO ActiveMatrix BusinessWorks domain with TIBCO Enterprise Administrator.

OpenAPI Specification Support

You can generate the OpenAPI specification for your TIBCO BusinessEvents REST API application that another TIBCO BusinessEvents application or any third-party product can easily consume. The OpenAPI specification contains all the underlying configurations of your application and exposes the capabilities of your application in an easily readable JSON or YAML format.

The consuming application can interpret the OpenAPI specification and create the project artifacts based on its contents: the concepts, events with or without payload (ontology concept ref or XSD element ref or complex element payload), destinations, channels, and shared resources.

TIBCO BusinessEvents supports the OpenAPI specification for only HTTP channel applications.

To know more about OpenAPI specification, see OpenAPI documentation.

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

Note: The following artifact fields are not supported when a TIBCO BusinessEvents project is exported using the OpenAPI Specification:

- Concept TTL and Event TTL
- Channel SSL section
- SSL fields
- Pageflow type for HTTP Channel
- Description field for a channel, destination, event, and shared connection

The following artifact fields are not supported when a TIBCO BusinessEvents project is imported using the OpenAPI Specification:

- Third-party specification having object type as array. The array property in Event is not supported.
- Event Payload that consists 'XML Element Reference' type under 'Complex Element'
- Nested properties
- When a project has an Event payload with multiple contained and referenced concepts, the import operation does not create all referenced or contained concepts if the x-ConceptPath and x-event-payload type tags are excluded from the exported specification.

Sample OpenAPI Specification

The following is the example OpenAPI specification for put employee data.

```
{
  "openapi" : "3.1.0",
  "info" : {
    "title" : "Simple API",
    "version" : "9",
    "description": "A simple API to illustrate OpenAPI concepts for put 
Employee data"
  },
 "servers" : [ {
    "url" : "http://example.io/v1:8110"
  } ],
   "tags" : [ {
    "name" : "/Channels/Emp"
  } ],
  "paths" : {
    "/Channels/Put_Employee_Data" : {
      "post" : {
        "description" : "Add Employee Data",
        "operationId" : "Channels_Put_Employee_Data_post",
        "parameters" : [ {
          "name" : "EmplID",
          "in" : "query",
          "schema" : {
            "type" : "integer"
          }
        }, {
          "name" : "EmployeeName",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "CompanyName",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "Address",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }],
        "requestBody" : {
          "content" : {
            "application/xml" : {
              "schema" : {
                "type" : "object",
                "$ref" : "#/components/schemas/Company_xml"
              }
            },
            "application/json" : {
              "schema" : {
                "type" : "object",
                "$ref" : "#/components/schemas/Company"
              }
            }
          }
        },
        "responses" : {
          "default" : {
            "description" : "Success Response"
          }
        }
      },
      "put" : {
       "description" : "Add Employee Data",
        "operationId" : "Channels_Put_Employee_Data_put",
        "parameters" : [ {
          "name" : "EmplID",
          "in" : "query",
          "schema" : {
            "type" : "integer"
          }
        }, {
          "name" : "EmployeeName",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "CompanyName",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "Address",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }],
        "requestBody" : {
          "content" : {
            "application/xml" : {
              "schema" : {
                "type" : "object",
                "$ref" : "#/components/schemas/Company_xml"
              }
            },
            "application/json" : {
              "schema" : {
                "type" : "object",
                "$ref" : "#/components/schemas/Company"
              }
            }
          }
        },
        "responses" : {
          "default" : {
            "description" : "Success Response"
          }
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "Company" : {
        "type" : "object",
        "properties" : {
          "CompanyName" : {
            "type" : "string"
          },
          "Address" : {
            "type" : "string"
          },
          "City" : {
            "type" : "string"
          },
          "State" : {
            "type" : "string"
          },
          "EmplID" : {
            "type" : "integer",
            "format" : "int32"
          }
        }
      },
      "Company_xml" : {
        "type" : "object",
        "xml" : {
          "name" : "Company"
        },
        "properties" : {
          "CompanyName" : {
            "type" : "string"
          },
          "Address" : {
            "type" : "string"
          },
          "City" : {
            "type" : "string"
          },
          "State" : {
            "type" : "string"
          },
          "EmplID" : {
            "type" : "integer",
            "format" : "int32"
          }
        }
      }
    }
  }
}
```

Exporting OpenAPI Specification

You can get the TIBCO BusinessEvents project resources as OpenAPI specification in either of the following ways.

- Export at design time from TIBCO BusinessEvents Studio, see Exporting Project as OpenAPI Specification at Design Time.
- Access at run time, see Accessing OpenAPI Specification at Run Time.

Note:

- When you export an OpenAPI specification for a TIBCO BusinessEvents application, all HTTP methods endpoints are created by default for each destination in the project.
- The variable value is exported for a global variable used in the artifact property configuration.
- Event payloads of type Complex Element and XML Element Reference are supported.
- XSD payload type does not support direct reference of an element from XSD.

Exporting Project as OpenAPI Specification at Design Time

The OpenAPI specification export utility allows you to export TIBCO BusinessEvents project artifacts in JSON or YAML format.

Procedure

1. In Studio Explorer, right-click anywhere in the project and select **Export**.

   The Export window opens.

2. Select **TIBCO BusinessEvents > OpenAPI Specification** and click **Next**.

3. In the OpenAPI Specification Export window, provide the following details.

Field	Description
Generate Yaml	Select this checkbox to generate the OpenAPI specification as a YAML file. By default, the OpenAPI specification is generated in JSON format.
Select OpenAPI Specification Location	Browse and select the output location. The project location is the default location of the OpenAPI specification.
Enter OpenAPI Specification File name	Enter a valid file name to save the OpenAPI specification.
Select Channel File	Browse and select the channel from the project. You can select only one channel at a time to export. For multiple channels, the first channel is set as the default channel and the path in the configuration is the default channel path.

4. Click **Finish**.

Result

The TIBCO BusinessEvents project is exported as an OpenAPI specification at the specified output location.

Enabling the RunTime Access to OpenAPI Specification

Accessing an OpenAPI specification for a TIBCO BusinessEvents application at run time requires the setting of some properties.

1. To access an OpenAPI specification over a server URL at run time, in the `be-engine.tra` file at *BE_HOME*`/bin` or in the project CDD file, set the following server property:

Field	Description
`be.engine.api.spec.enable`	Set to true to enable the HTTP channel to generate and serve the OpenAPI specification at run time. By default, it is `false`.

2. This step is required if you start the TIBCO BusinessEvents engine from TIBCO BusinessEvents Studio for accessing a runtime specification.

   In TIBCO BusinessEvents Studio, set the `be.engine.api.spec.enable` server property in either of the following ways:

   - In the Run Configurations window, provide the start up property file, see Adding and Working with Launch (Debug or Run) Configurations.
   - In the project CDD file, add the server property in the Properties section on the Cluster tab.

3. To access an OpenAPI specification on a SwaggerUI, in the `be-engine.tra` file at *BE_HOME*`/bin` or in the project CDD file, set the following properties:

Field	Description
`be.engine.openapi.version`	Specify the correct OpenAPI version. The OpenAPI version must match the value `3.0.n` to access the specification on a SwaggerUI page, for example `3.0.0`. The default value is `3.1.0`.
`be.channel.http.propertiesAsXML`	Set to true to include event properties such as XML or JSON in the response when replying to an HTTP request event. The default value is `false`.
`be.http.json.rootElement.ignore`	Set to true to exclude the root element and its attributes for the event with a JSON payload. The default value is `false`.

What to do next

See Accessing OpenAPI Specification at Run Time for accessing the OpenAPI specification of your TIBCO BusinessEvents application at run time.

Accessing OpenAPI Specification at Run Time

You can access the OpenAPI specification for a running TIBCO BusinessEvents application over a URL or SwaggerUI page.

Before you begin

Enable the runtime OpenAPI specification access, see Enabling the RunTime Access to OpenAPI Specification.

Procedure

1. Run the TIBCO BusinessEvents engine for a project which you want to generate an OpenAPI specification for.
2. When TIBCO BusinessEvents engine is running, in the browser, enter the following HTTP channel URL endpoint:

   The port number must be the port number used in the HTTP shared resource.

Field	Description
`http://`*`host_name:port_number`*`/openapi.json`	Enter to access the OpenAPI specification in a JSON format.
`http://`*`host_name:port_number`*`/openapi.yaml`	Enter to access the OpenAPI specification in a YAML format.
`http://`*`host_name:port_number`*`/swagger/index.html`	Enter to access the OpenAPI specification on a SwaggerUI.

   To access the OpenAPI specification for a particular channel from a project containing multiple channels, specify the correct port number of the channel in the URL.

Exporting OpenAPI Specification by Using the Command-Line Utility

You can export a project as an OpenAPI specification by using the `studio-tools` command-line utility.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.

3. On the command prompt, run the following command:

   ```
   studio-tools.exe -core exportOpenAPI [-h] [-x] [-o specificationFileName]
   [-d outputDirectory] [-genyaml generateYAMLSpecification] [-p projectDir] [-ch
   channelName]
   ```

   For example:

   ```
   studio-tools.exe -core exportOpenAPI -o httpOpenAPI -p
   ..\..\examples\standard\HTTPChannel\HTTPChannel -ch HTTP
   ```

Options

The following table describes the command parameters:

exportOpenAPI SpecificationOptions

Option	Description
`-core exportOpenAPI`	Specifies the `exportOpenAPI` operation for exporting a TIBCO BusinessEvents Studio OpenAPI specification into the workspace. The exported specification file is by default in JSON format.
`-h`	*Optional*. Displays help.
`-x`	*Optional*. Overwrites the specified output file if it exists when the value of `-x` is set to True. If the value of `-x` is set to False, then the export operation does not proceed and a different output file should be mentioned. If the value of `-x` is not specified, then a confirmation message prompts whether to overwrite the file.
`-o`	*Optional*. Specifies the exported OpenAPI specification file name. The Project name is the default specification name.
`-d`	*Optional*. Specifies the absolute path to the output directory where the OpenAPI specification is exported. The default path where the specification file exports is the source project path.
`-p`	Source project: Specifies the absolute path to the project directory of the

Option	Description
	TIBCO BusinessEvents Studio project.
`-genyaml`	*Optional*. Generates a YAML file format for the exported OpenAPI specification.
`-ch`	Specifies the channel name from the project. You can select only one channel at a time.

Importing OpenAPI Specification

TIBCO BusinessEvents Studio can consume an OpenAPI specification and automatically generate artifacts in a TIBCO BusinessEvents project by using the import utility.

Procedure

1. In Studio Explorer, right-click anywhere in the project and select **Import**.

   The Import window opens.

2. Select **TIBCO BusinessEvents > OpenAPI Specification** and click **Next**.

3. In the OpenAPI Specification Import window, provide the following details.

Field	Description
Import From URL	Select this checkbox to import an OpenAPI specification directly from an URL. You can also import an OpenAPI specification from a third-party cloud product such as Github.
Select File	Browse and select an OpenAPI specification file in JSON or YAML format from your local file system. You can drag a local OpenAPI specification file from your system and drop it to this field and it picks the file path.
Provide URL	Provide an URL to import an OpenAPI specification from. This field is enabled if you select the **Import from URL** checkbox.
User name	Username for accessing the remote repository URL.
Password	Password for accessing the remote repository URL.
Enter or select the parent folder	Select a TIBCO BusinessEvents project where you want to create artifacts.

4. Click **Finish**.

   If the imported OpenAPI specification has any error or invalid value, you see an error message and no artifacts are imported. The error details are printed on TIBCO BusinessEvents Studio console.

5. Save the project.

Result

The supported project artifacts for the HTTP channel are created under the respective artifacts folders in TIBCO BusinessEvents Studio as per the configuration mentioned in the OpenAPI specification. The hierarchy and relationships between the artifacts are maintained.

Re-importing OpenAPI Specification

When you import an OpenAPI specification, TIBCO BusinessEvents Studio creates the project artifacts based on the configuration mentioned in the specification. After importing, if any changes are made to the OpenAPI specification, you can re-import the specification to update the TIBCO BusinessEvents application.

To re-import an OpenAPI specification, follow the same procedure mentioned for importing an OpenAPI specification. For detail process, see Importing OpenAPI Specification.

When you re-import an OpenAPI specification, TIBCO BusinessEvents Studio asks to confirm to override the existing project artifacts. After confirmation, the artifacts are recreated with latest specification.

Importing OpenAPI Specification by Using the Command-Line Utility

You can import a project as an OpenAPI specification by using the `studio-tools` command-line utility.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.
3. On the command prompt, run the following command:

```
studio-tools.exe -core importOpenAPI [-h] [-x] [-u] [-pass] [-i
inputSpecificationFile] [-p projectDir]
```

   For example:

```
studio-tools.exe -core importOpenAPI -i https://raw.githubusercontent.com/user_
name/630Test1/main/http_open_st_v34.json?token=GHSAT0AAAAAAB6DBAVHH2K4JQZKBJ5YNOIKY7LDE7Q -
p C:\Users\xyz_user\workspace\630_V37\OpenGithubImport -u -xyz_user -pass -1234
```

Options

The following table describes the command parameters:

importOpenAPI SpecificationOptions

Option	Description
`-core importOpenAPI`	Specifies the `importOpenAPI` operation for importing a TIBCO BusinessEvents Studio OpenAPI specification into the workspace.
`-h`	*Optional*. Displays help.
`-x`	*Optional*. Overwrites the artifacts if they exist in the target project when the value of `-x` is set to True. If the value of `-x` is set to False, then the export operation does not proceed and a different output file should be mentioned. If the value of `-x` is not specified, then a confirmation message prompts whether to overwrite the file.
`-u`	*Optional*. Specifies the username when the OpenAPI specification is imported from a private repository URL.
`-pass`	*Optional*. Specifies the password when the OpenAPI specification is imported from a private repository URL.
`-i`	Specifies the input file name for the OpenAPI specification. The input file name can also be a remote URL.
`-p`	Target project: Specifies the absolute path to the project directory of the TIBCO BusinessEvents Studio project where the specification is to be imported.

AsyncAPI Specification Support

TIBCO BusinessEvents supports the industry-adopted AsyncAPI specification, which allows an interaction between event-driven, APIs-based applications in a consistent and standard manner. The AsyncAPI specification is a machine-readable format for defining the message exchange patterns as per consumers for event-driven APIs and works over any protocol. See AsyncAPI documentation for details.

You can export your TIBCO BusinessEvents project using the AsyncAPI specification that can be consumed by another TIBCO BusinessEvents project or a third-party product in a different ecosystem and generate resources.

TIBCO BusinessEvents can parse an imported AsyncAPI specification and create the required project artifacts based on its contents: the concepts, events (with or without payload), channels, destinations, and shared resources.

In TIBCO BusinessEvents, property-based configurations artifacts for Kafka, shared resources for HTTP and both for JMS channels can be exported and imported by using the AsyncAPI specification. For Kafka and JMS channels, there must be a one-to-one mapping between channel destination and topic, queue.

Note: The following artifact fields are not supported when a TIBCO BusinessEvents project is exported by using an AsyncAPI Specification:

- Concept TTL and Event TTL
- Channel SSL section
- SSL fields
- Pageflow type for HTTP Channel
- Description field for a channel, destination, event, and shared connection

The following artifact fields are not supported when a TIBCO BusinessEvents project is imported by using an AsyncAPI Specification:

- Third-party specification having object type as array. The array property in Event is not supported.
- oneOf section in AsyncAPI message definition
- Event Payload that consists 'XML Element Reference' type under 'Complex Element'
- Nested properties
- When a project has an Event payload with multiple contained and referenced concepts, the import operation does not create all referenced or contained concepts if the x-ConceptPath and x-event-payload type tags are excluded from the exported specification.

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

Overview of AsyncAPI Specification Objects

Servers

The Servers object is a mandatory component and must define some properties such as host, port, protocol, and so on as default. There can be multiple server objects defined in the application. However, TIBCO BusinessEvents considers only default server properties and exports the server specification into the default server.

Channels Object

Channels are the sender and receiver of messages, managed by a server. Channels contain the relative location to the server, the destination in a TIBCO BusinessEvents application, and the expected message format.

Components Object

Components are reusable objects to represent various AsyncAPI specifications characteristics and must be explicitly referenced from outside the component object. Components object contains `messages` and `schemas`.

For details about all AsyncAPI objects, see AsyncAPI documentation.

Sample AsyncAPI Specification

The following is an AsyncAPI specification example for the Streetlights Kafka API:

```
{
    "asyncapi": "2.1.0",
    "info": {
        "title": "Streetlights Kafka API",
        "version": "1.0.0",
        "description": "The Smartylighting Streetlights API allows you
to remotely manage the city lights",
        "license": {
            "name": "Apache 2.0",
            "url": "https://www.apache.org/licenses/LICENSE-2.0"
        }
    },
    "servers": {
        "default": {
            "url": "test.mykafkacluster.org:8092",
            "protocol": "kafka",
            "protocolVersion": "1.0.0",
            "description": "Test broker"
        }
    },
    "defaultContentType": "application/json",
    "channels": {
        "lightingMeasured": {
            "description": "The topic on which measured values may be
produced and consumed.",
            "subscribe": {
                "summary": "Inform about environmental lighting
conditions of a particular streetlight.",
                "operationId": "receiveLightMeasurement",
                "message": {
                    "$ref": "#/components/messages/lightMeasured"
                }
            }
        }
    },
    "components": {
        "messages": {
            "lightMeasured": {
                "name": "lightMeasured",
                "title": "Light measured",
                "summary": "Inform about environmental lighting
conditions of a particular streetlight.",
                "contentType": "application/json",
                "payload": {
                    "$ref": "#/components/schemas/lightMeasuredPayload"
                }
            }
        },
        "schemas": {
            "lightMeasuredPayload": {
                "type": "object",
                "properties": {
                    "lumens": {
                        "type": "integer",
                        "description": "Light intensity measured in
lumens."
                    },
                    "sentAt": {
                       "type": "string",
                        "format": "date-time",
                        "description": "Date and time when the message
was sent."
                    }
                }
            }
        }
    }
}
```

Exporting AsyncAPI Specification

You can get the TIBCO BusinessEvents project resources as AsyncAPI specification in either of the following ways.

- Export at design time from TIBCO BusinessEvents Studio, see Exporting Project as AsyncAPI Specification at Design Time.
- Access at run time, see Accessing AsyncAPI Specification at Run Time.

Note:
- Only the concepts and events referenced in the destination of the selected channel, are exported using AsyncAPI specification.
- The variable value is exported for a global variable used in the artifact property configuration.
- Event payloads of type Complex Element and XML Element Reference are supported.
- XSD payload type does not support direct reference of an element from XSD.

Exporting Project as AsyncAPI Specification at Design Time

The AsyncAPI specification export utility allows you to export TIBCO BusinessEvents project artifacts in JSON or YAML format.

Procedure

1. In Studio Explorer, right-click anywhere in the project and select **Export**.

   The Export window opens.

2. In the Export window, select **TIBCO BusinessEvents > AsyncAPI Specification** and click **Next**.

3. In the AsyncAPI Specification Export window, provide the following details.

Field	Description
Generate Yaml	Select this checkbox to generate the AsyncAPI specification as a YAML file. By default, the AsyncAPI specification is generated in JSON format.
Select AsyncAPI Specification Location	Browse and select the output location. The project location is the default location of the AsyncAPI specification.
Enter AsyncAPI Specification File name	Enter a valid file name to save the AsyncAPI specification.
Select Channel File	Browse and select the channel from the project. You can select only one channel at a time to export. For multiple channels, the first channel is set as the default channel and the path in the configuration is the default channel path.

4. Click **Finish**.

Result

The TIBCO BusinessEvents project is exported as an AsyncAPI specification for the selected channel at the specified output location.

Enabling the Runtime Access to AsyncAPI Specification

Follow the steps to enable accessing a specification over a server URL.

1. In the `be-engine.tra` file located at *`BE_HOME`*`/bin`, set the following server properties:

Field	Description
`be.engine.api.spec.enable`	Set to true to enable an HTTP endpoint to generate and serve the AsyncAPI specification at run time By default, it is `false`.
`be.engine.http.ping.port`	Specify the port number. The generated specification is displayed on the specified port. The default port is `8180`. When multiple engines are running on the same host, specify a range for port numbers to use the next port in line when a port is not available. For example: `be.engine.http.ping.port=8180-8185` When a single port is specified, the default port range is considered as the next 10 ports from the specified port number.

2. This step is required if you start the TIBCO BusinessEvents engine from TIBCO BusinessEvents Studio to access a runtime specification.

   In TIBCO BusinessEvents Studio, set the server properties, `be.engine.api.spec.enable` and `be.engine.http.ping.port`, in either of the following ways:

   - In the Run Configurations window, provide the start up property file, see Adding and Working with Launch (Debug or Run) Configurations.
   - In the project CDD, add the server properties in the Properties section on the Cluster tab.

What to do next

For accessing the AsyncAPI specification of your TIBCO BusinessEvents application at run time, see Accessing AsyncAPI Specification at Run Time.

Accessing AsyncAPI Specification at Run Time

You can access the AsyncAPI specification for a running TIBCO BusinessEvents application over a URL and it can be maintained in a repository on a server.

A runtime specification can synchronize a TIBCO BusinessEvents application with its AsyncAPI specification.

Before you begin

Enable the access to the AsyncAPI specification over the server URL by configuring properties, see Enabling the Runtime Access to AsyncAPI Specification.

Procedure

1. Run the TIBCO BusinessEvents engine for a project which you want to generate AsyncAPI specification for.
2. When TIBCO BusinessEvents engine is running, in the browser, enter the HTTP server URL endpoint:

   ```
   http://host_name:port_number/asyncapi
   ```

   The parameter `asyncapi` lists all available channels in the project.
3. To generate the AsyncAPI specification for a particular channel, provide the channel name to the endpoint:

   ```
   http://host_name:port_number/asyncapi?chName=/Channels/channel_name
   ```

   For example,

   ```
   http://host_name:port_number/asyncapi?chName=/Channels/KafkaChannel
   ```
4. To generate the AsyncAPI specification in a YAML format, append the parameter `generateYaml=true` to the endpoint:

   ```
   http://host_name:port_number/asyncapi?chName=/Channels/channel_name&generateYaml=true
   ```

Result

The AsyncAPI specification is generated and served on the specified HTTP port at run time.

The specification is regenerated every time the endpoint has been entered.

Exporting AsyncAPI Specification by Using the Command-Line Utility

You can export a project as a AsyncAPI specification by using the `studio-tools` command-line utility.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.
3. On the command prompt, run the following command:

   ```
   studio-tools.exe -core exportAsyncAPI [-h] [-x] [-o specificationFileName]
   [-d outputDirectory] [-genyaml generateYAMLSpecification] [-jsonpayload
   isJsonPayload] [-p projectDir] [-ch channelName]
   ```

   For example:

   ```
   studio-tools.exe -core exportAsyncAPI -o httpAsyncAPI -p
   ..\..\examples\standard\HTTPChannel\HTTPChannel -ch HTTP
   ```

Options

The following table describes the command parameters:

exportAsyncAPI SpecificationOptions

Option	Description
`-core exportAsyncAPI`	Specifies the `exportAsyncAPI` operation for exporting a TIBCO BusinessEvents Studio AsyncAPI specification into the workspace. The

Option	Description
	exported specification file is by default in JSON format.
`-h`	*Optional*. Displays help.
`-x`	*Optional*. Overwrites the specified output file if it exists when the value of `-x` is set to True. If the value of `-x` is set to False, then the export operation does not proceed and a different output file should be mentioned. If the value of `-x` is not specified, then a confirmation message prompts whether to overwrite the output file.
`-o`	*Optional*. Specifies the exported AsyncAPI specification file name. The Project name is the default specification name.
`-d`	*Optional*. Specifies the absolute path to the output directory where the AsyncAPI specification is exported. The default path where the specification file exports is the source project path.
`-p`	Source project: Specifies the absolute path to the project directory of the TIBCO BusinessEvents Studio project.
`-genyaml`	*Optional*. Generates a YAML file format for the exported AsyncAPI specification.
`-jsonpayload`	*Optional*. Specifies the JSON payload for the exported AsyncAPI specification.
`-ch`	Specifies the channel name from the project. You can select only one channel at a time.

Importing AsyncAPI Specification

TIBCO BusinessEvents Studio can consume an AsyncAPI specification and automatically generate artifacts in a TIBCO BusinessEvents project by using the import utility.

Procedure

1. In Studio Explorer, right-click anywhere in the project and select **Import**.

   The Import window opens.

2. Select **TIBCO BusinessEvents > AsyncAPI Specification** and click **Next**.

3. In the AsyncAPI Specification Import window, provide the following details.

Field	Description
Import From URL	Select this checkbox to import an AsyncAPI specification directly from a URL. You can also import an AsyncAPI specification from a third-party cloud product such as Github. The URLs that require any authentication to access them are not supported.
Select File	Browse and select an AsyncAPI specification file in JSON or YAML format from your local file system. You can drag a local AsyncAPI specification file from your system and drop it to this field and it picks the file path.
Provide URL	Provide a URL to import an AsyncAPI specification from. This field is enabled if you select the **Import from URL** checkbox.
Enter or select the parent folder	Select a TIBCO BusinessEvents project where you want to create artifacts.

4. Click **Finish**.

   If the imported AsyncAPI specification has any error or invalid value, you see an error message and no artifacts are imported. The error details are printed on TIBCO BusinessEvents Studio console.

5. Save the project.

Result

The supported project artifacts for the selected channel are created under the respective artifacts folders in TIBCO BusinessEvents Studio as per the configuration mentioned in the AsyncAPI Specification. The hierarchy and relationships between the artifacts are maintained.

Importing AsyncAPI Specification by Using the Command-Line Utility

You can import a project as an AsyncAPI specification using the `studio-tools` command-line utility.

Procedure

1. Navigate to `BE_HOME/studio/bin/`.
2. Open the `studio-tools` utility.
3. On the command prompt, run the following command:

   ```
   studio-tools.exe -core importAsyncAPI [-h] [-x] [-u] [-pass] [-i inputSpecificationFile] [-p projectDir]
   ```

   For example:

   ```
   studio-tools.exe -core importAsyncAPI -i https://raw.githubusercontent.com/user_name/630Test1/main/http_async_st_v34.json?token=GHSAT0AAAAAAB6DBAVHQOGLLH4FQUQMJCXAY7LCUBQ -p C:\Users\xyz_user\workspace\630_V37\AsyncGithubImport -u [-xyz_user] -pass [-1234]
   ```

Options

The following table describes the command parameters:

importAsyncAPI SpecificationOptions

Option	Description
`-core importAsyncAPI`	Specifies the `importAsyncAPI` operation for importing a TIBCO BusinessEvents Studio AsyncAPI specification into the workspace.
`-h`	*Optional*. Displays help.

Option	Description
`-x`	*Optional*. Overwrites the artifacts if they exist in the target project when the value of `-x` is set to True. If the value of `-x` is set to False, then the export operation does not proceed and a different output file should be mentioned. If the value of `-x` is not specified, then a confirmation message prompts whether to overwrite the file.
`-u`	*Optional*. Specifies the username when the AsyncAPI specification is imported from a private repository URL.
`-pass`	*Optional*. Specifies the password when the AsyncAPI specification is imported from a private repository URL.
`-i`	Specifies the input file name for the AsyncAPI specification. The input file name can also be a remote URL.
`-p`	Target Project: Specifies the absolute path to the project directory of the TIBCO BusinessEvents Studio project where the specification is to be imported.

Performance Profiler

You can run TIBCO BusinessEvents Performance Profiler utility to gather statistics about activities that occur during each RTC cycle. This information helps to identify bottlenecks in the project, which can often be addressed by redesigning rules or other aspects of a project.

The Profiler utility collects statistics relating to the run to completion (RTC) rule evaluation cycle in an inference agent. The utility does not collect data about object management. It also does not collect data for other types of agents. It helps you understand, for example, internal execution times and frequencies of rules.

The Profiler records time spent during each RTC on activities such as the number of times each condition or action is performed, and total time spent on each condition and action. A complete RTC includes conditions and actions, although any individual RTC might contain only conditions or only actions.

Statistics are collected for each completed RTC. When the Profiler is directed to stop during an RTC, it continues to collect data for the current RTC until that RTC is completed.

When the Profiler is turned off, it continues to write statistics for the current session until that session is completed. So the RTC in progress is always completed, even if the Profiler is directed to stop during an RTC.

After the Profiler finishes, the statistics data is written to the specified (or default) file and cleared from memory. The engine continues to run. If the engine stops before the Profiler completes, the file is not created.

You can run the Profiler and turn it off in three ways:

Using properties

The Profiler turns on when the agent initializes at system startup. Used to profile RTC time, including startup rule functions.

Using TIBCO BusinessEvents catalog functions

The Profiler turns at the beginning of the next RTC after the function call, if it has not already been enabled. (There is no effect if the Profiler is already on). Used to turn the Profiler on and off inside a rule or rule function.)

Using a TIBCO Hawk method

The Profiler turns on by invoking a TIBCO BusinessEvents microagent Hawk method. The Profiler is turned on at the beginning of the next RTC after the method call, if it has not already been enabled. (There is no effect if the Profiler is already on). Used to turn the Profiler on and off dynamically.

The Delimiter Character

The Profiler is tab-delimited by default. The delimiter character can be changed adding the following property in the CDD file:

```
be.engine.profile.delimiter
```

Specify the delimiter using a String value. Enclose the value in double quotes (the quotes are not used as part of the delimiter).

For example to use an open curly brace as the delimiter, you would specify `"{"` as the value. Do not choose a character used in rule conditions.

Use a single character if the application into which you will import the output uses a one-character delimiter. When importing the file into Excel, do not check the “Treat consecutive delimiters as one” option. Consecutive delimiters indicate a column that is empty.

Tip: Also, when importing the file into Excel, set the timestamp field to Text (and not General, which is the default).

Turning Profiler On and Off

There are different ways you can turn the Profiler on and off:

- Using Properties
- Using Functions
- Using TIBCO Hawk Methods

Using Properties

Set the following properties in the Cluster Deployment Descriptor (CDD) Processing Unit tab, for all processing units (engines) whose RTC performance you want to profile.

Profiler Configuration Properties (Sheet of)

Property	Notes
`be.engine.profile.`*`Agent_Class_ Name`*`.enable`	If set to true, enables Profiler for the specified agent class (*`Agent_Class_Name`*) when the agent initializes. Default is false.
`be.engine.profile.*.enable`	If set to true, enables the Profiler for all agents when each agent initializes, even when a specified agent class Profiler is disabled. Default is false.
`be.engine.profile.`*`Agent_Class_ Name`*`.file`	The name of output file that the Profiler writes to, for the specified agent (*`Agent_Class_Name`*). Default behavior is as follows: If `be.engine.profile.*.file` is specified and `be.engine.profile.`*`Agent_Class_Name`*`.file` is not specified, then the file name is the value of `be.engine.profile.*.file`, with the *`Agent_ Class_Name`* appended. If the properties `be.engine.profile.*.file` and `be.engine.profile.`*`Agent_Class_Name`*`.file`s are not specified the name is created as follows: `be-profile_`*`Agent_Class_Name`*`.csv`
`be.engine.profile.*.file`	The default (prefix for the) name of the output file that the Profiler writes to. In all cases, the appropriate *`Agent_Class_Name`* is appended. Default name is `be-profile.csv` and it is located under the current working directory, if file name is

Property	Notes
	not specified.
`be.engine.profile.`*`Agent_Class_Name`*`.duration`	Specifies the duration of profile data collection in seconds, for the specified *`Agent_Class_Name`*). When the duration period ends, the Profiler continues to collect statistics for the current RTC until the RTC is completed, then outputs data and stops. So the RTC in progress is always completed, even if the Profiler is directed to stop during an RTC. If you set duration to a value of zero or less (<= 0), then profiling continues until agent stops or Profiler is explicitly turned of using a function or Hawk method. Default is -1.
`be.engine.profile.*.duration`	Specifies the duration of profile data collection in seconds, for all agents. When the duration period ends, the Profiler continues to collect statistics for the current RTC until the RTC is completed, then outputs data and stops. So the RTC in progress is always completed, even if the Profiler is directed to stop during an RTC. If you set duration to a value of zero or less (<= 0), then profiling continues until the agents stop or Profiler is explicitly turned of using a function or Hawk method. When `be.engine.profile.`*`Agent_Class_Name`*`.duration` and `be.engine.profile.*.duration` are both present, the duration specified in `be.engine.profile.`*`Agent_Class_Name`*`.duration` takes precedence.

Property	Notes
	Default is -1.
`be.engine.profile.`*`Agent_Class_Name`*`.level`	Level of depth that profile data will be collected for the specified agent (*`Agent_Class_Name`*): • -1: all levels of profile data are collected, including RTC level and conditions and actions within the RTC. • 1: Only RTC level of profile data will be collected (and no condition and action data). Default is -1.
`be.engine.profile.*.level`	Level of depth that profile data will be collected for all agents: • -1: all levels of profile data are collected, including RTC level and conditions and actions within the RTC. • 1: Only RTC level of profile data will be collected (and no condition and action data). When `be.engine.profile.`*`Agent_Class_Name`*`.level` and `be.engine.profile.*.level` are both present, the level specified in `be.engine.profile.`*`Agent_Class_Name`*`.level` takes precedence. Default is -1.

Using Functions

This section assumes you understand how to use TIBCO BusinessEvents functions. It tells you that functions to use and the effect of each function.

Tip: You can turn the Profiler on using the engine properties, and turn it off using a function or Hawk method, as desired. See notes for `be.engine.`*`Agent_Class_Name`*`.profile.duration` and `be.engine.*.profile.duration` in Using Properties .

To turn the Profiler on

In your rule or rule function, use the following function to turn on the Profiler:

```
Engine.Profiler.startCollectingToFile(String fileName, int level, long
duration)
```

The above function turns on the TIBCO BusinessEvents Profiler. It starts collecting data for the specified duration for the agent in which the rule or rule function that calls this function is run. The Profiler starts collecting data at the beginning of the next RTC.

Profile data is output to the specified file in comma-separated value format at the end of the duration period, unless the Profiler is turned off before the end of the duration. In such a case, it is the output at the end of the RTC that completes after the Profiler is turned off.

Input arguments are the same as the engine properties show in Using Properties:

```
String fileName: See be.engine.profile.Agent_Class_Name.file
```

`int` *`level`*: See `be.engine.profile.`*`Agent_Class_Name`*`.level`

`long` *`duration`*: See `be.engine.profile.`*`Agent_Class_Name`*`.duration` (Here *`Agent_Class_Name`* is the current agent in which the rule or rule function that calls this function is run.)

To turn the Profiler off

In your rule or rule function, use the following function to turn off the Profiler:

```
Engine.Profiler.stopCollecting()
```

The above function turns off the TIBCO BusinessEvents Profiler. The function writes the profile data to a file for the agent in which the rule or rule function that calls this function is included (the file is output at the end of the RTC that completes after the Profiler is turned off). There is no effect if the Profiler is not on.

Using TIBCO Hawk Methods

This section assumes you understand how to use TIBCO Hawk methods. It tells you which methods to use and the effect of each method.

Tip: You can turn the Profiler on using properties, and turn it off using a function or Hawk method, as desired. See notes for `be.engine.profile.duration` in Using Properties.

Before you Begin

Ensure that the property `hawk.enabled` is set to true in the CDD at the cluster level before the TIBCO BusinessEvents engine starts.

To turn the Profiler on

Use the following method to turn on the Profiler:

```
StartFileBasedProfiler(String session, String fileName, int level, long
duration)
```

The above method turns on the TIBCO BusinessEvents Profiler for the specified agent. The Profiler starts collecting data at the beginning of the next RTC for the specified duration.

This method works the same way as the `Engine.Profiler.startCollectingToFile()` function (see Turning Profiler On and Off), except that it requires you to specify an agent class.

Input arguments are the same as the engine properties shown in Using Properties:

`String`*`session`*: If you want to monitor multiple agents, run the method once for each agent, specifying the agent class name in each case. But when there is only one agent, the session parameter is optional.

String *`fileName:`* See `be.engine.profile.`*`Agent_Class_Name`*`.file`

`int` *`level`*: See `be.engine.profile.`*`Agent_Class_Name`*`.level`

`long` *`duration`*: See `be.engine.profile.`*`Agent_Class_Name`*`.duration`

If you attempt to turn on the Profiler when it is already running, an error is returned. But the running Profiler is not affected.

To turn the Profiler off

In your rule or rule function, use the following function to turn off the Profiler:

```
StopFileBasedProfiler(String session)
```

The above method turns off the TIBCO BusinessEvents Profiler. It writes the profile data into a file for the specified agent when the current RTC is completed. You must run the method once for each session, as needed.

If you attempt to turn off the Profiler when it is already off, an error is returned. But there is no effect on the Profiler.

Profiler Reference

The table in this section explains each of the columns in the Profiler report. Data is grouped by `RTC_Stats_Type` and Description. (Description contains information about the specific RTC.) All data collected for conditions and actions performed during each RTC is listed within each RTC grouping.

Three rows of column headers for the RTC, condition, and action are listed at the beginning of the file:

- One for statistics relating to the overall RTC
- One for statistics relating to conditions
- One for statistics relating to actions.

The first column of each data line is always the statistic type, which begins with one of `RTC-`, `CONDITION-`, or `ACTION-`.

Data is also grouped, one row for the overall RTC, and zero or more rows for different conditions or actions or both, as appropriate.

Profiler Column Heading Reference (Sheet of)

Column Heading	Notes
Statistics Relating to the Overall RTC	

Column Heading	Notes
`RTC_Stats_Type`	Type of rule evaluation cycle (RTC) There are 10 different types of RTC: • `RTC-Object-Asserted` • `RTC-Object-Modified` • `RTC-Object-Deleted` • `RTC-Event-Expired` • `RTC-Execute-Rule` • `RTC-Invoke-Action` • `RTC-Invoke-Function` • `RTC-Post-Process` • `RTC-Repeat-TimeEvent` • `RTC-Reevaluate-Element`
`Timestamp`	The time at which the first RTC begins.
`Description`	Information relating to the current `RTC_Stats_Type`. For example, the description of type `RTC-Object-Asserted` is the name of the object being asserted.
`NumExecuted`	The total number of times the same RTC has run.
`TotalRtcTime`	Total time in milliseconds spent on the total number of executions of the same RTC.
`AvgRtcTime`	`TotalRtcTime` / `NumExecuted`
`MaxRtcTime`	The maximum time in milliseconds spent on a single RTC.
`MinRtcTime`	The minimum time in milliseconds spent on a single RTC.

Column Heading	Notes
`MaxResolvedTime`	The maximum time in milliseconds spent to resolve a single RTC, include condition evaluation and action execution, but exclude operations related to object management (OM).
`MinResolvedTime`	The minimum time in milliseconds spent to resolve a single RTC, include condition evaluation and action execution, but exclude operations related to object management (OM).
Statistics Relating to Conditions	
`CONDITION_Stats_Type`	Type of rule condition. One of the following: • `CONDITION-Filter` • `Condition-Join`
`Timestamp`	The timestamp of the first time the RTC begins.
`RuleDescription`	Name of the rule containing the condition, or name of the state machine transition rule containing the condition.
`ConditionDescription`	Condition statement of a rule or a state machine transition rule for user-defined condition, or predefined condition name for internal conditions. When a user-defined rule condition has a commented-out line, the `ConditionDescription` of the next condition is `//… Only applies to CONDITION_Stats_Type`
`NumEvaluated`	The total number of times this condition is evaluated in the same RTC.
`NumEvalTrue`	The total number of times the `Join` condition

Column Heading	Notes
	evaluated is true. This value is the sum of `NumEvalTruePropagatedLeft` and `NumEvalTruePropagatedRight`.
`TotalTime`	Total time in milliseconds spent on the total number of condition evaluations.
`AvgTime`	`TotalTime / (NumLeftSearch + NumRightSearch)`
`MaxTime`	The maximum time in milliseconds spent on a single condition evaluation.
`MinTime`	The minimum time in milliseconds spent on a single condition evaluation.
`NumEvalPropagatedLeft`	Number of times the join condition evaluation is triggered by object assertion propagated from the left side of the condition.
`NumEvalTruePropagatedLeft`	Number of times the join condition evaluates to true and evaluation is triggered by object assertion propagated from the left side of the condition.
`AvgRateEvalTruePropagatedLeft`	The average rate that the condition evaluates to true and evaluation is triggered by object assertion propagated from the left side of the condition.
`MaxNumEvalTruePropagatedLeft`	The maximum number of times the join condition evaluates to true and evaluation is triggered by object assertion propagated from the left side of the condition.
`MinNumEvalTruePropagatedLeft`	The minimum number of times the join condition evaluate to true and evaluation is triggered by object assertion propagated from the left side of the condition.

Column Heading	Notes
`NumEvalPropagatedRight`	Number of times the join condition evaluation is triggered by object assertion propagated from the right side of the condition.
`NumEvalTruePropagatedRight`	Number of times the join condition evaluates to true and evaluation is triggered by object assertion propagated from the right side of the condition.
`AvgRateEvalTruePropagatedRight`	The average rate that the condition evaluates to true and evaluation is triggered by object assertion propagated from the right side of the condition.
`MaxNumEvalTruePropagatedRight`	The maximum number of times the join condition evaluates to true and evaluation is triggered by object assertion propagated from the right side of the condition.
`MinNumEvalTruePropagatedRight`	The minimum number of times the join condition evaluate to true and evaluation is triggered by object assertion propagated from the right side of the condition.
Statistics Relating to Actions	
`ACTION_Stats_Type`	Type of Actions. There are 10 RTC types and four action types. The four action types are: • `ACTION-Rule-Action` • `ACTION-Event-Expiry` • `ACTION-Invoke-Action` • `ACTION-Invoke-Function`
`Timestamp`	The timestamp of the first time the action execution begins.

Column Heading	Notes
`Description`	Information about the action corresponding to the current action type. For example, the description of type `ACTION-Rule-Action` is the name of the rule.
`NumExecuted`	The total number of times the same action has been run. A complete action has two phases, action execution and operation.
`TotalActionTime`	Total time in milliseconds spent on the total number of actions. `TotalActionTime = TotalExecutionTime + TotalOperationTime`
`AvgActionTime`	`TotalActionTime` / `NumExecuted`.
`MaxActionTime`	The maximum time in milliseconds spent on a single action.
`MinActionTime`	The minimum time in milliseconds spent on a single action.
`TotalExecutionTime`	Total time in milliseconds spent on the total number of action execution phases. Action execution time is the time the Rete network spends on running the action, for example, time spent in creating new objects, deleting existing objects, and so on.
`AvgExecutionTime`	`TotalExecutionTime` / `NumExecuted`.
`MaxExecutionTime`	The maximum time in milliseconds spent on a single action execution.
`MinExecutionTime`	The minimum time in milliseconds spent on a single

Column Heading	Notes
	action execution.
`TotalOperationTime`	Total time in milliseconds spent on the total number of action operation phases. Action operation time is the time TIBCO BusinessEvents spends on applying changes to the Rete network, for example, time spent asserting newly created objects, or retracting deleted objects.
`AvgOperationTime`	`TotalOperationTime` / `NumExecuted`.
`MaxOperationTime`	The maximum time in milliseconds spent on a single action operation.
`MinOperationTime`	The minimum time in milliseconds spent on a single action operation.
`MaxAgenda`	The maximum size of the rule agenda as a result of all action operations.
`MinAgenda`	The minimum size of the rule agenda as a result of all action operations.

Testing and Debugging Projects

This chapter explains how to test and debug projects within TIBCO BusinessEvents Studio.

The sections on debugger assume some familiarity with Eclipse Java debugger, as well as TIBCO BusinessEvents.

Note: Viewing the TIBCO BusinessEvents Studio Error Log

Errors in TIBCO BusinessEvents Studio functionality are reported in the error log. You can view the error log file in either of these ways:

- In TIBCO BusinessEvents Studio, navigate to **Help > About TIBCO BusinessEvents Studio > Installation Details > Configuration > View Error Log**
- In the file system, navigate to `Your_Workspace` > `.metadata` > `.log`

You can create test data and set breakpoints before you run debugger, or using views within the Debug perspective.

Preparing to Run (Test) or Debug a Project

You can do these tasks in any order. For information on setting up see Tester Preferences.

Build an EAR File

Procedure

1. In TIBCO BusinessEvents Studio, select the project in the BusinessEvents Studio Explorer panel, then select **Project > Build Enterprise Archive.**
2. If you want to build an EAR with debug information, select the **Generate Debug > Info** checkbox (selected by default).

3. In the File Location field provide the path to the EAR file and specify the EAR file name, for example, c:\myprojects\myproject.ear.
4. Generate the EAR file.

Create Test Data

You can create test data for use across multiple sessions, or you can provide rule input data while the engine is running. See Test Data.

For Remote Debugging Only Configure Java Debug Interface (JDI)

To configure for remote debugging, you configure the `be-engine.tra` file on the remote engine so that the engine uses the Java Debug Interface (JDI) for remote debugging.

You then configure a debug configuration for remote debugging, as explained in Adding and Working with Launch (Debug or Run) Configurations. You must specify the same JDI port number in the TRA file and in the debug configuration.

Configure Java Debug Interface (JDI)

For each TIBCO BusinessEvents engine you want to enable for remote debugging, perform the following actions.

Procedure

1. Open the BE_HOME`/bin/be-engine.tra` file for editing.
2. Specify the port on which you want the engine to listen, using the environment variable `tibco.env.JDI_PORT`, for example:

   ```
   tibco.env.JDI_PORT 5192
   ```

   Where `5192` is the default value. If multiple engines run on the same machine, ensure that each has a unique port.

3. Uncomment the following line:

```
-Xrunjdwp:transport=dt_socket,address=%JDI_
PORT%,suspend=na,server=y
```

4. Start or restart the engine with the `-d` (or `-debug`) command.

Adding and Working with Launch (Debug or Run) Configurations

Launch configurations are of two types: *run configurations* and *debug configurations*. All launch configurations have the same basic fields, but debug configurations have extra settings for remote debugging.

Before you begin, ensure you know the EAR file location, the CDD file location, and the name of the processing unit (configured in the CDD file) that you want to use. (The processing unit runs as an engine.)

Add and Work with Launch Configurations

A reference to the settings is provided in Launch Configurations Reference.

Note: For remote debugging, you first have to set properties in the remote engine TRA file. See For Remote Debugging Only Configure Java Debug Interface (JDI).

Procedure

1. Perform one of the following actions.
    - To create a debug configuration, click the down-arrow to the right of the debugger button () on the toolbar. From the dropdown list, and select **Debug Configurations**. Or, select **Run > Debug Configurations**.
    - To create a run configuration, click the down-arrow to the right of the Run button () on the toolbar. From the dropdown list, select **Run Configurations**. Or, select **Run > Run Configurations**.

      You see the Debug Configurations or Run Configurations dialog.
2. Select an option from the list on the left:
    - For testing or local debugging, select **TIBCO BusinessEvents Application.**
    - For remote debugging, select **Remote TIBCO BusinessEvents Application**.
3. Depending on your requirements, perform one of the following actions.
    - To edit a configuration, expand TIBCO BusinessEvents Application or Remote TIBCO BusinessEvents, and select an existing debug configuration.
    - To add a configuration, click the **New Configuration** () button.
    - To duplicate a configuration, select the configuration, and then click the **Duplicate** () button. Modify, then save as a new configuration.
    - To delete a configuration, select the configuration and then click the **Delete** () button.

      When you add, edit or duplicate a configuration, Configuration fields appear in the right panel.

4. If you have selected TIBCO BusinessEvents Application in Step 2, perform the following actions.
   - Select the **Main** tab and configure the values as explained in Launch Configurations Reference.
   - Select the **Classpath** tab and configure the classpath for external libraries or custom functions as needed, for example if the project uses JMS channels. See Working with External Library and Custom Function Paths for details.
   - Select the **Environment** tab and configure environment variables as needed, to run or debug the project in TIBCO BusinessEvents Studio. You can add new variables. You can select and then edit existing variables. You can append your edited variable to the existing environment variable, or you can replace the existing environment variable with it. For example, if a custom function depends on a native library, you can add the path to that library using the `PATH`, `LD_LIBRARY_PATH`, `SHLIB_PATH`, or `LIBPATH` variable, as appropriate for your operating system.

     Note: On Linux platforms, when Legacy ActiveSpaces is used, you must set `LD_LIBRARY_PATH` to *`AS_HOME`*`/lib` in the **Environment** tab.
   - If you need to display characters that are not present in the system's default charset on the console, set the encoding to UTF-8.
   - Select the **Common** tab and set the **Encoding** to Other. Then select **UTF-8** from the dropdown menu.

     This procedure is documented in **Plug-in Development Environment Guide > Reference > Launchers > JUnit Plug-in Test Launcher > Common Tab**.
   - For information on standard Eclipse features incorporated into this area, see Eclipse help.
5. If you have selected Remote TIBCO BusinessEvents Application in Step 2, you must connect to the remote engine. Click the **Remote** tab and specify the host name or IP address and port the engine is running on. Use the same JDI port number in the TRA file and in the debug configuration (see For Remote Debugging Only Configure Java Debug Interface (JDI)).
6. To save configuration settings, click **Apply**.

7. Perform one of the following actions.
    - Click **Close** and save the configuration you worked on.
    - If you have done the setup as explained in this chapter and are ready to run or debug, click **Debug** to save the configuration and open the debugger, or click **Run** to save the configuration and run the engine.

      You can then assert test data as desired to observe the effect of the data on the engine. See Asserting Rule Input Data.

Launch Configurations Reference

See Adding and Working with Launch (Debug or Run) Configurations for the related procedure.

Note: The Source and Common tabs are standard Eclipse dialogs. See Eclipse help for details about the use of those tabs. If the project uses third-party JAR files, you must also reference them in the **Classpath** tab, and update the Environment as needed.

For Testing and Local Debugging

Field	Notes
Name	A descriptive name. It appears in the dropdown list of configurations.
Main Tab	
Project	Browse to select the name of the TIBCO BusinessEvents project to use for this configuration. You can select from projects in the workspace. The project currently selected in BusinessEvents Studio Explorer appears by default.
VM Arguments	Optional. Provide options and parameters using -V, -D, -X, and so on. For example, to set global variables use: `-V`*`Variable`*`=`*`value`*.

Field	Notes
Startup Options	Optional. Provide the start up property file (`--propFile`), custom property file (`-p`) or engine name (`-n`) to run a configuration with specific startup values and other parameters.
CDD File Location	Browse to select the CDD file to be used for this launch configuration.
Processing Unit Name	Select the name of the processing unit (PU) whose values are used for this launch configuration. The dropdown list displays PUs available in the CDD specified in the CDD File Location setting.
Working Directory	The location of the working directory for the TIBCO BusinessEvents engine. It is used to store temporary files and logs. Browse to and select an existing directory. Path names that do not start with the root directory are assumed by the operating system to start from the working directory.
EAR File	Browse to select the EAR file to be used for this launch configuration. The EAR file must be generated with the Generate Debug Info option checked.

For Remote Debugging

You can debug a running TIBCO BusinessEvents engine on the current machine or another machine

Field	Notes
Name	A descriptive name. It appears in the drop-down list of configurations.
Main Tab	
Project	Browse to select the name of the TIBCO BusinessEvents project to use for this configuration. You can select from projects in the workspace. The name of the TIBCO BusinessEvents project in the workspace. It must be the same as

Field	Notes
	the project that is running remotely.
Remote Tab	
Remote Connection	Enter VM arguments for running the remote VM. Alternatively, add these to the `java.extended.properties` property in the remote application's runtime properties (TRA) file. Default value is: `-Xdebug -Xrunjdwp:transport=dt_socket,address=25192,suspend=n,server=y`
Host	The host name or IP address of the remote computer where you are running the TIBCO BusinessEvents engine. Default is `localhost`.
Port	Communication port for the debugger, on the remote machine. Default is 25192

Test Data

You can send data to the engine through channels in the normal way. You can also create data within TIBCO BusinessEvents Studio for assertion into the running engine's Rete network during testing and debugging. Doing so means you don't have to have the external resources in place in order to test or debug the runtime.

You can create data while you are working in the debug perspective. You can also create and save test data ahead of time and save it in your project for later use.

The internal data is used at the following *bottom* tabs, within the **Rule Input** tab:

Tester Data

This is concept and event instance data. You create it in the TIBCO BusinessEvents Development perspective, right-click an entity and select Create Test Data. You use it in the Debug perspective: select the **Rule Input View > Tester Data** tab.

Rule Data

This is data directly provided as rule input. It is created and edited in the **Rule Input View > Rule Data** tab (in the Debug perspective).

How you use test data depends on what aspect of a project you want to test or debug.

Note: If the test data gets asserted in Cache Agent then the studio shows a message "Test data cannot be asserted" along with a "Cannot assert Object when activeMode is false" message on console as cache agents are not actively participating in processing inputs. Only Inference Agents processes inputs and the test data should be asserted into Inference Agent.

Working with Concept and Event Test Data

You can enter data in the Test Data editor for event payloads, and for concept, event and scorecard properties, including properties that are primitive types, array types, contained concepts, and reference concepts. If the concept, event or scorecard properties are associated with a domain model, then the test data gets populated with the values in the domain model. You can use global variables.

Creating Concept and Event Instance Test Data

Procedure

1. To create test data in BusinessEvents Studio Explorer (or in the Debug perspective BusinessEvents Studio Explorer view), you can perform either of the following steps:
    - Right-click on an event or concept and click **New**, then **Test Data or Project**
    - Right-click **Project > Test Data**.
2. Right-click an event or concept and click **Create Test Data**.

   The Test Data editor appears showing the event or concept properties as column headers.

3. In the Test Data editor, click **Add** to add rows for new instances. You can add your own unique **extId** values to the test data input, as needed. You can also use global variables.

   You can also remove existing rows by selecting one or more rows and clicking **Remove**.

4. (Optional) To add new rows with random test data, in the Test Data editor, click **Generate Random Test Data**, select the applicable options, and click **OK**.

   For more information about the options available, see Generate Random Test Data Reference.

5. (Optional) Select the **Invoke preprocessor while asserting Events** checkbox and select the preprocessor that you want to invoke.

6. Specify the assertion rate for the test data.

   - **As fast as possible** (default) - Select this option to assert the test data as fast as possible.
   - **Events (or Concepts) per second** - Select this option to slow the pace of assertion of test data to track a specific output. You can slow the assertion rate to one concept or event per second.

7. Click **Save**. The entity's test data is saved to an XML file stored within the `TestData` folder in the project root.

   The `/TestData` folder is the default location (see Tester Preferences).

Generate Random Test Data Reference

Setup the random test data generator by defining the amount of the test data to be generated, method of generation, and value bounds (maximum and minimum values) for the generated test data.

Generate Random Test Data Reference

Field	**Description**
Owner Entity	Displays the entity for which the test data is created.
Property List	List of properties of the entity. Select the properties for which you want to generate the test data.

Field	Description
Number of Rows to generate	Number of rows to be added in the test data.
Generation Methods	Select the method to be used for generating the random test data. The methods are: • Random • Constant • Prefixed The methods are available based on the data type of the selected property. Based on the data type and generation method, you can provide the values for the test data generator. See the Generate Random Test Data Reference table.
Allow Empty Values	Select the checkbox to include empty values in the randomly generated test data.
Probability of generating an empty value	Specify the probability (0 to 100%) of generating an empty value.

Test Data Generation Methods

Property Type	Generation Methods	Value Bounds	Example Value
String	random, constant, prefixed, enumerate (domain models only)	Minimum and maximum length of string	`mXd8`
int	random, constant, incremental, enumerate (domain models only)	Minimum and maximum integer value. The values can be negative.	`4453`
long	random, constant, incremental, enumerate	Minimum and maximum long value. The values	`6302`

Property Type	Generation Methods	Value Bounds	Example Value
	(domain models only)	can be negative.	
double	random, constant, incremental, enumerate (domain models only)	Minimum and maximum double value. The values can be negative.	`256.3929935664355`
boolean	random, constant, enumerate (domain models only)	**Accepted values:** true or false	`true`
DateTime	random, constant, incremental, enumerate (domain models only)	Minimum and maximum DateTime value. The dates can be in the past.	`2019-10-17T04:12:37`

Edit Test Data for Concepts and Events in BusinessEvents Studio Explorer

To add more test data or edit test data you created earlier, perform the following actions.

Procedure

1. In BusinessEvents Studio Explorer, expand the `TestData` folder in the root of the project.

   The `/TestData` folder is the default location.

2. Drill down to the test data you want to edit. The folder structure matches the project's event and concept folder structure.

3. Double-click the name of the test data file you want to edit.

   The test data editor opens.

4. Add, remove, and edit rows of test data as required. Follow the Step 3 to Step 7 of Creating Concept and Event Instance Test Data to edit the test data.

Edit Concept and Instance Test Data in the Rule Input View

Procedure

1. Open the Rule Input view, if it is not already shown. To make the Rule Input View visible, perform one of the following actions.
    - Select the Debug perspective as follows: select **Window > Open Perspective**, or click the Open Perspective () button). Then select **Other > Debug**. The views associated with the Debug perspective open.
    - In the **Window** menu, click **Show View** > **Other**. Expand **TIBCO BusinessEvents**, and select **Rule Input**. Click **OK**.
2. Click the **Rule Input** tab and then select the **Tester Data** bottom tab.

    For each concept or event for which you created test data, you see one row showing the project path to that concept or event.
3. Double-click the row for the concept or event whose test data you want to edit.

    The Test Data editor appears, showing the rows of test data already created.
4. Edit as desired.

Working with Rule Data

To create and save rule Data:

Procedure

1. Open the Rule Input view, if it is not already shown. To make the Rule Input view visible, perform either of the following steps, as needed:
    - Select the Debug perspective: Select **Window > Open Perspective**, or click the **Open Perspective** () button). Then select **Other > Debug**. The views associated with the Debug perspective open.
    - In the **Window** menu, click **Show View** > **Other**. Expand **TIBCO BusinessEvents**, and select **Rule Input**. Click **OK**.

2. Provide input from the mapper as explained in Using the Function Argument Mapping Wizard.
3. Specify the Launch Target. This is generally the locally running engine. The other fields become active.
4. Specify an event or concept in the Entity URI field. If you specify an event, then specify the Destination URI. Specify the Rule Session (agent) to use.
5. Perform either of the following steps:
   - Click **Save** to save the input values.

     You can reuse these saved values for repetitive tests.
   - Click **Load** to load the input values from an existing XML file.
   - Click **Assert** to assert the data to a running engine.

Setting Breakpoints in Rules and Rule Functions

Setting breakpoints is an Eclipse feature. This section provides only basic information. You can also use advanced features such as importing and exporting breakpoints, and using class prepare breakpoints. See Eclipse help for more details about all breakpoint functionality. You can set or change breakpoints during a debug session also.

Note: You cannot step through custom Java code.

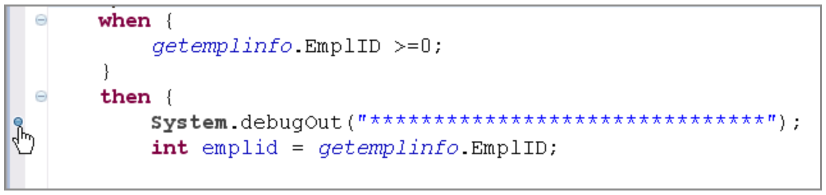
Procedure

1. In TIBCO BusinessEvents Studio, open the source editor for a rule or rule function.

   You can work with breakpoints in the debugger perspective as well as in the TIBCO BusinessEvents Studio development perspective.

2. To add a breakpoint put your cursor in the left margin (gray area) next to a row where you want to add a breakpoint. Perform one of the following steps.
   - Right-click and select Toggle Breakpoint.
   - Double-click in the left margin.

     A breakpoint icon appears in the left margin:



```
when {
    getemplinfo.EmplID >=0;
}
then {
    System.debugOut("******************************");
    int emplid = getemplinfo.EmplID;
```

3. Set and adjust breakpoints as needed. Select a breakpoint, right-click, and perform any of the following steps.
   - To disable a breakpoint, select Disable.
   - To remove a breakpoint, select Toggle Breakpoint. Or you can double-click the breakpoint.
   - To edit a breakpoint's properties, select Breakpoint properties. A dialog displays (with mostly runtime options). For example, you can use a class prepare breakpoint (so the running program is suspended when the specified class or interface is first loaded by the Java VM).

     A breakpoint may not be set exactly where you place it. This is because TIBCO BusinessEvents ensures that breakpoints fall on an executable statement, and moves any that do not to the nearest executable statement.

     Note: Executing step code line does not match rule editor code line

     This situation happens when there is a mismatch between the debug line information stored in the EAR, and the information in the open rule editor. To resolve the problem, recompile the EAR file.

Running Debugger

Make sure you have completed the setup as needed. Then run the debugger as explained below.

Note: For projects containing an RV channel, add the path to the location of the RV libraries by adding *RV_HOME*/`lib` to the existing path.

As needed, switch to Debug perspective. Select **Window > Open Perspective**, or click the **Open Perspective** () button). Then select **Other > Debug**.

Alternatively, wait until TIBCO BusinessEvents prompts you to change to debug perspective. This happens when the debugger reaches the first breakpoint.

Click the down-arrow to the right of the debugger () button. You see a dropdown list. Perform one of the following steps.

- Select a debug configuration from the list.

  Tip: To add configurations to the dropdown list, select **Organize Favorites** from the list.

- Select Debug Configurations. At the Debug Configurations dialog, select a debug configuration and click **Debug**.

- Click the **debugger** () button or select **Run > Debug**. (Only if you have already launched the debugger with a configuration.)

  The debugger starts a TIBCO BusinessEvents engine, using parameters provided in the launch configuration, if any were provided.

Procedure

1. Assert data as explained in Asserting Rule Input Data.
2. View the results as explained in Viewing the Results.
3. Use the standard Eclipse commands such as step into (F5), step over (F6), step return, step return, and so on, depending on what you want to examine.

   See the options in the **Run** menu and in the **Breakpoints** tab for more options, and use Eclipse help for details with these Eclipse features.

   When you are finished, click the **Terminate** () button to stop the engine.

Running Tester

Make sure you have completed setup as needed. You can create test data before you run tester, or using views within the Debug perspective. Other setup must be done before you begin.

Note: For projects containing an RV channel, add the path to the location of RV libs:

See the following sections for details:

First do the setup needed for your testing. See the following sections:

- Preparing to Run (Test) or Debug a Project
- Adding and Working with Launch (Debug or Run) Configurations
- Test Data

Then run tester as explained below.

Run the Tester

As needed, switch to Debug perspective. Select **Window > Open Perspective**, or click the **Open Perspective** () button). Then select **Other > Debug**.

Select **Run > Run Configurations** or click the down-arrow to the right of the **Run** () button and choose **Run Configurations**.

If you have configured favorites, you can click the down-arrow to the right of the **Run** () button and choose a favorite.

Procedure

1. At the Run Configurations dialog select a run configuration and click **Run**.

   Note: If you have already started an engine using a run configuration and want to start it again, click the Run ( ) button or Select **Run > Run**.

   A TIBCO BusinessEvents engine starts, using parameters provided in the run configuration, if any were provided.

2. Assert data as explained in Asserting Rule Input Data.

3. View the results as explained in Viewing the Results.

   When you are finished, click the **Terminate** () button to stop the engine.

Asserting Rule Input Data

You can assert either tester data, or rule data for use in Tester or Debugger.

Note: To perform any task in the **Rule Input** tab, you must keep the engine running.

You can also send messages to destinations, as you would at runtime.

See also Test Data for details about creating and saving data.

Note:

- Test data you created earlier for concepts, events, or scorecards appears in the Debugger perspective. By default the view appears in the middle on the left.
- For cache application using preprocessors to load a concept from cache, if an event does not have the default destination then that destination has to be assigned to event before starting test in Rule Input View.

Assert Tester Data

Procedure

1. Select the **Rule Input** tab and then the **Tester Data** tab.
2. One entity can have multiple rows of test data. To select which row or rows of test data to assert, double click the entity's URI in the Select Test Data view. It appears in the Test Data editor. Select one or more checkboxes in the Use column as desired. You can repeat this step for all the entities shown in the Select Test Data view, as needed.

   You can edit the test data at this time too and you can add test data for more entities. See Working with Concept and Event Test Data for details.
3. In the Input panel, **Launch Target** field, specify which engine to use.

   Multiple engines can run in tester at the same time, for example, a cache agent engine, and an engine running inference agents.
4. In the Input panel, **Rule Session** field, specify which agent to assert the data to.

   One engine (processing unit) can have multiple inference agents.
5. Click **Start Test**.

   You see console messages and results. See Viewing the Results for details about understanding the results of a test or debugger run.
6. As appropriate, select more test data to assert (as in Step 2), and again click **Start Test**. Once again, analyze the results of asserting that data using the other views.

   To Run Tester or Debugger with Rule Data
7. Open the Rule Input view, if it is not already shown. In the **Window** menu, click **Show View > Other**. Expand **TIBCO BusinessEvents**, and select **Rule Input**. Click **OK**.
8. Perform one of the following steps.
   - Provide input from the mapper as explained in Using the Function Argument Mapping Wizard. Optionally you can click **Save** and save the values to an XML file.
   - Click **Load** to load the input values from an existing XML file.
9. Specify the **Launch Target, Entity URI, Destination URI** and **Rule Session**.
10. Click **Assert**.

Result

See Viewing the Results for details about understanding the results of a test or debugger run.

Viewing the Results

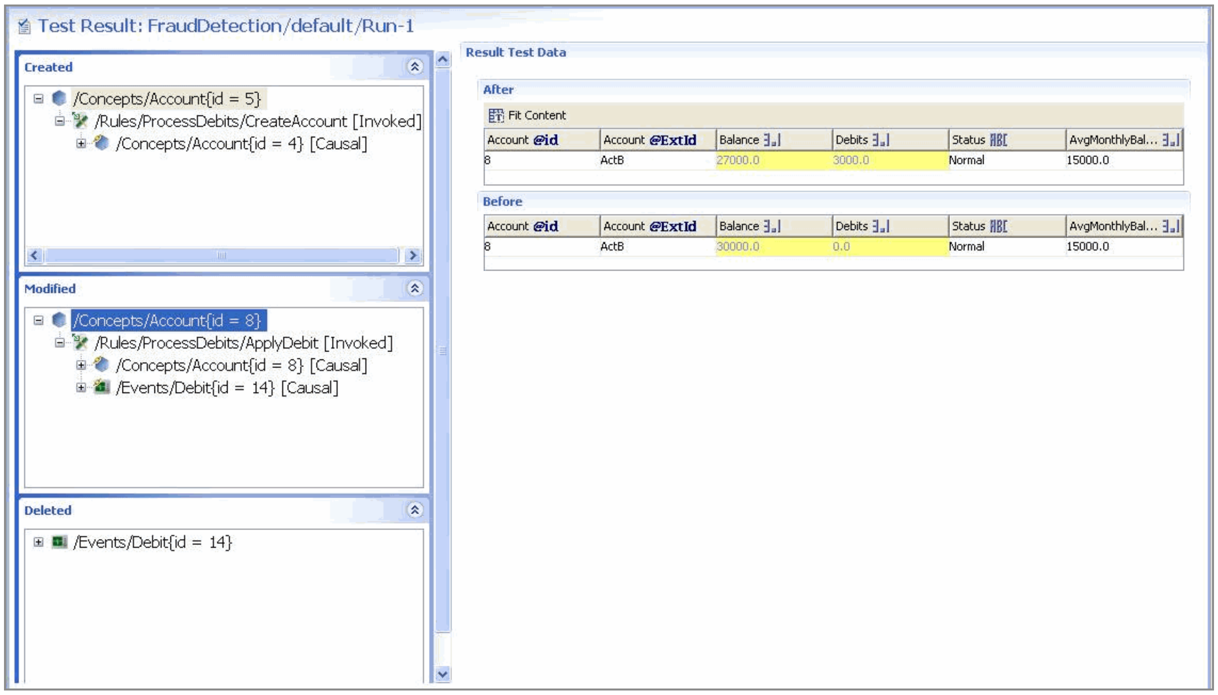
After data is asserted to the working memory (see Running Tester), a run to completion (RTC) cycle occurs and you see the following:

- The **Console** tab displays engine console messages.
- The result data editor appears to show the results of the run. The editor title displays the result data filename with the format `Run-`*`n`*`.resultdata`.
- The results of this test are stored in an XML format in a `.resultdata` file

By default the results are stored in the `/TestData/Project Name/Processing Unit` Name folder. See Tester Preferences for information on changing the location where test data is stored.

You can also open the result data in its editor by double-clicking the `.resultdata` file in BusinessEvents Studio Explorer.

Figure 18: Opening the .resultdata file in Result Data Editor



Test Result: FraudDetection/default/Run-1

Created

- /Concepts/Account{id = 5}
  - /Rules/ProcessDebits/CreateAccount [Invoked]
    - /Concepts/Account{id = 4} [Causal]

Modified

- /Concepts/Account{id = 8}
  - /Rules/ProcessDebits/ApplyDebit [Invoked]
    - /Concepts/Account{id = 8} [Causal]
    - /Events/Debit{id = 14} [Causal]

Deleted

- /Events/Debit{id = 14}

Result Test Data

After

Account @id	Account @ExtId	Balance	Debits	Status	AvgMonthlyBal...
8	ActB	27000.0	3000.0	Normal	15000.0

Before

Account @id	Account @ExtId	Balance	Debits	Status	AvgMonthlyBal...
8	ActB	30000.0	0.0	Normal	15000.0

Understanding Result Data

The Result Data view shows the results of a single RTC cycle. On the left of the view, you see the **Created**, **Modified**, and **Deleted** entities.

Expanding each entity shows the rule that affected the entity, and also the causal object that triggered the rule.

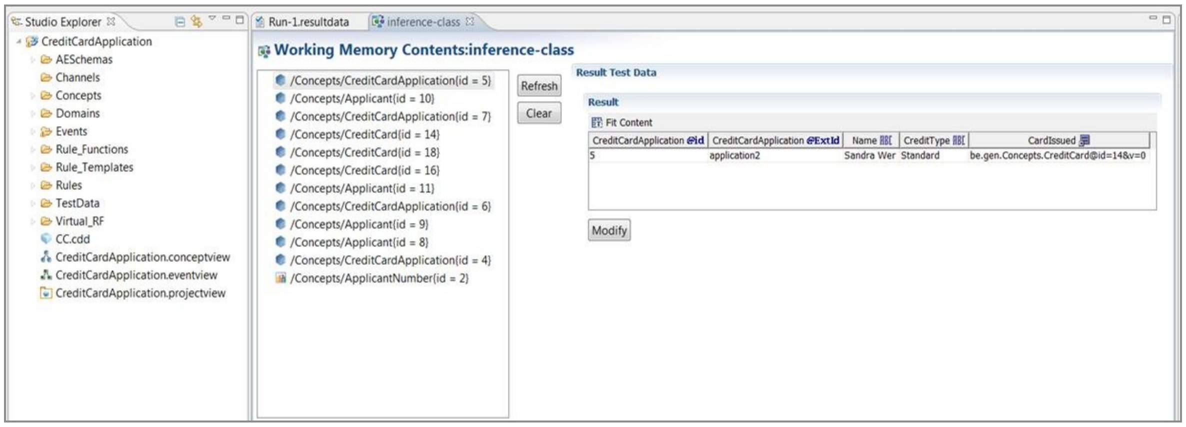
Click any entity to display its values in the **Result** panel.

Note: Test results are not shown for cache applications. They are shown only for in-memory applications.

Viewing and Understanding Working Memory Contents

The Working Memory concepts or scorecards can be manipulated in Working Memory View. Option to modify Working Memory Contents is available only in the Working Memory editor.
Change the property values in Working Memory View and click **Modify**.

Figure 19: Modifying Working Memory Contents in the Editor



/Concepts/CreditCardApplication{id = 5}
/Concepts/Applicant{id = 10}
/Concepts/CreditCardApplication{id = 7}
/Concepts/CreditCard{id = 14}
/Concepts/CreditCard{id = 18}
/Concepts/CreditCard{id = 16}
/Concepts/Applicant{id = 11}
/Concepts/CreditCardApplication{id = 6}
/Concepts/Applicant{id = 9}
/Concepts/Applicant{id = 8}
/Concepts/CreditCardApplication{id = 4}
/Concepts/ApplicantNumber{id = 2}

CreditCardApplication @id	CreditCardApplication @ExtId	Name	CreditType	CardIssued
5	application2	Sandra Wer	Standard	be.gen.Concepts.CreditCard@id=14&v=0

Procedure

1. Click the **Working Memory Contents** icon on the toolbar, or from the **Project** menu.
2. Select the appropriate rule session also known as the agent.

   The number of objects in the working memory contents is specified in the **Preferences**. For more details, see Tester Preferences.

Unit Testing Using JUnit

JUnit is a unit testing framework for the Java programming language. It is an instance of the xUnit architecture for unit testing frameworks. TIBCO BusinessEvents uses `BEUnit` (a JUnit based extension) to author unit tests for BusinessEvents applications.

JUnit Use Cases

In BusinessEvents, individual changes in business logic and data could also affect functioning of the BusinessEvents application. Using a unit testing framework, you can better understand the effect of individual changes to the BusinessEvents application.

For example, during migration to a new TIBCO BusinessEvents version, you can use the unit testing framework to test if your old application performs as expected in the new version of the TIBCO BusinessEvents software.

You can refer the BEUnitTest example located at *BE_HOME*`\examples\standard\BEUnitTest\BEUnitTest` for more information.

Note: If you are using a BEUnit project that is migrated from previous TIBCO BusinessEvents version, you must add the property `<pathelement path="../../../../lib/ext/tpcl/apache" />` to the `build.xml` file of the project. This helps to print appropriate logs when you run the BEUnit project using Apache Ant.

BEUnit (JUnit Extension) Implementation

The BEUnit test framework provides helper classes to author standard JUnit test suites, which you can invoke in the context of a running BusinessEvents engine.

The following classes are the major classes for the BEUnit test framework:

- `com.tibco.cep.runtime.service.tester.beunit.BETestEngine` - This class is used for setting up and configuring the BusinessEvents engine to run in a tester context. The BETestEngine class provides methods to assert data into the running test engine, invoke functions, reset the session between tests, and so on.

- `com.tibco.cep.runtime.service.tester.beunit.TestDataHelper` - This class provides helper methods to work with the test data. The existing test data files from previous versions of BusinessEvents can be used in BEUnit tests. You can also create individual concepts and events using the methods available in the TestDataHelper class.
- `com.tibco.cep.runtime.service.tester.beunit.Expecter` - This class provides a comprehensive set of methods to introspect the results of a test run.
- `com.tibco.cep.runtime.service.tester.beunit.ExpectationType` - This Enum class provides a set of fields that identifies the expected result of the test run. This class is used with the Expecter class in BEUnit test suites.

For more information on these classes, refer to *Java API Reference*.

Unit Test Suite Flow

The general flow of a particular unit test suite is as follows:

1. The BETestEngine is configured and the test engine is started. This is performed during the setup phase of the test suite.
2. One or more individual unit tests are run against a particular BusinessEvents application as specified in the setup.
3. For each individual unit test, the test creates and assert some concepts/events.
4. Next, use an Expecter class method to specify a particular outcome. This can include the case such as, determining that a particular rule fired, determining that a concept has been modified, or testing whether an event has been asserted. The Expecter class is used to test the specific behavior of the BusinessEvents application. If the expectation fails (for instance, RuleB fired before RuleA for certain input) the unit test also fails.

You can reset the tester session at any time; however, it is a good practice to do it at the start of an individual test. Resetting the session clears the working memory, and resets any active timers. So, any objects that were asserted in previous tests do not have an effect on the current test.

Adding Unit Test Suite for the BusinessEvents Project

TIBCO BusinessEvents Studio provides a wizard to create a unit test suite for the project using new wizard dialog.

Procedure

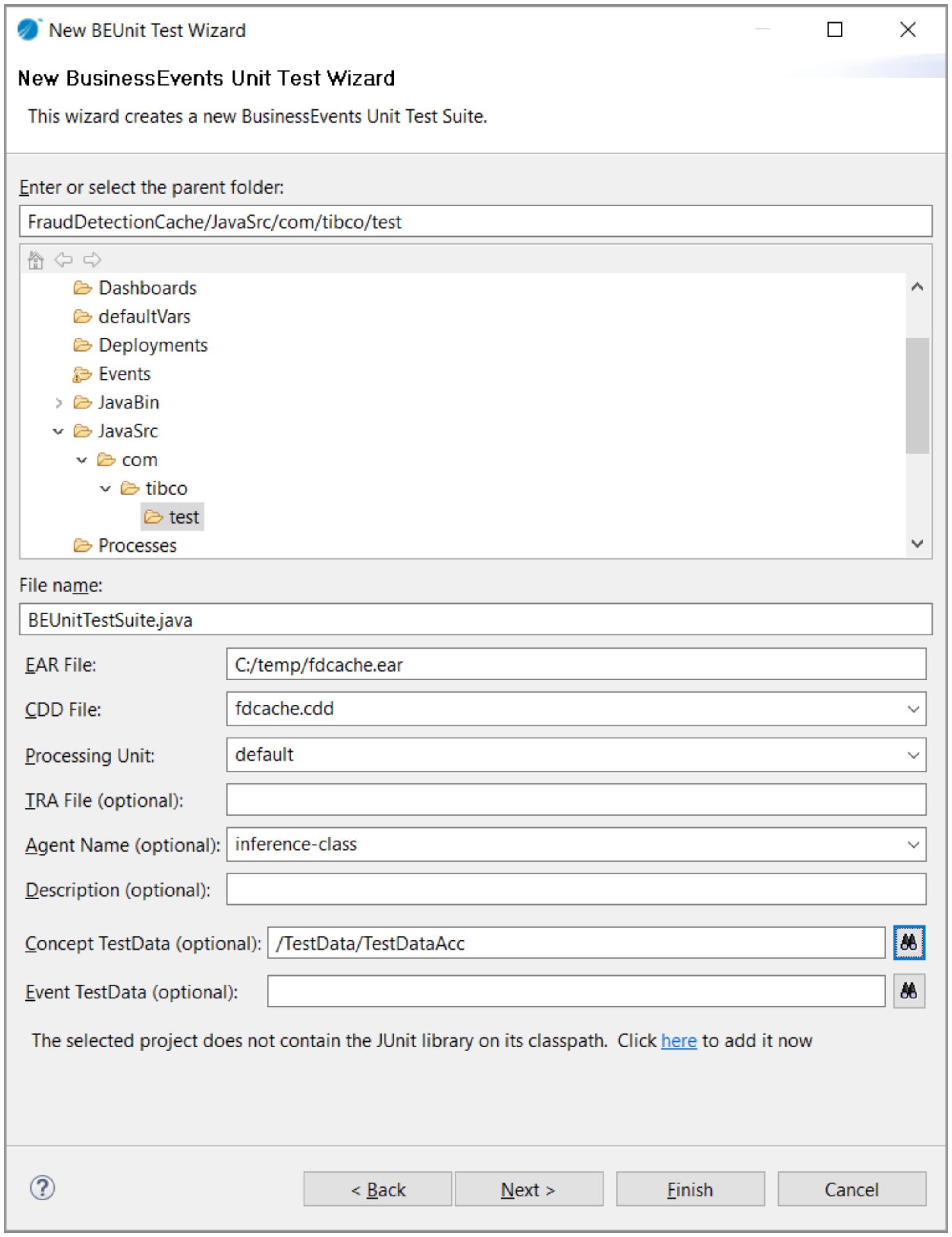
1. In BusinessEvents Studio explorer, select the BusinessEvents project for which you want to create the BEUnit test suite and in the menu bar, select **File > New > Other**.
2. In the wizard, select **TIBCO BusinessEvents > BEUnit Test Suite** and click **Next**.

   The New BusinessEvents Unit Test Wizard is displayed.

3. In the New BusinessEvents Unit Test Wizard, enter the values of the following fields and click **Next**.

Field	Description
Parent Folder	Select the Java source folder of the project as the parent folder in the project. **Note:** The test suite should be placed in an existing Java source folder. If the parent folder is not a Java source folder then after test suite creation, add the folder as Java source folder.
File Name	Name of the unit test suite Java file.
EAR File	The EAR archive for the selected project. The path can be a relative path as well.
CDD File	Select the CDD file of the project. This populates the **Processing Unit** and **Agent Name** options. The path can be a relative path as well.
Processing Unit	Processing unit for the project.
TRA File	*Optional* Filepath to the TRA file of the project. The path can be a relative path as well.
Agent Name	*Optional* Agent name for the selected CDD. If there are multiple agents, then the recommendation is to provide agent names separated by commas.
Description	*Optional* Short description for the test suite.
Concept TestData	*Optional* Select the concept test data for the project.
Event TestData	*Optional* Select the event test data for the project.

If the project does not have JUnit library in its classpath, the wizard displays a note in the bottom. Click the "here" link in the note to add the JUnit library automatically to the project. See the following figure for a sample screen of the wizard.



New BEUnit Test Wizard

New BusinessEvents Unit Test Wizard

This wizard creates a new BusinessEvents Unit Test Suite.

Enter or select the parent folder:

FraudDetectionCache/JavaSrc/com/tibco/test

- Dashboards
- defaultVars
- Deployments
- Events
- JavaBin
- JavaSrc
  - com
    - tibco
      - test
- Processes

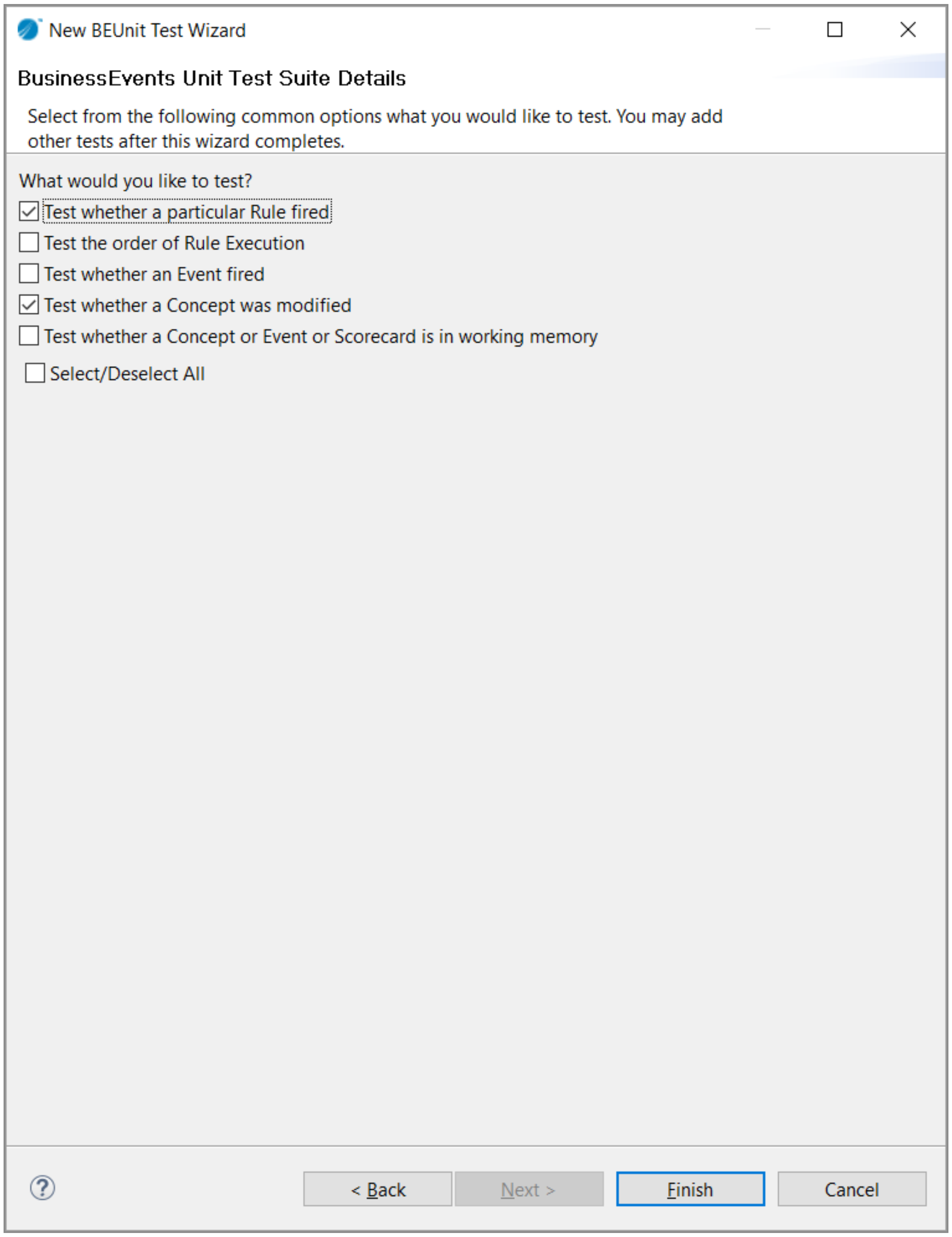
File name:

BEUnitTestSuite.java

EAR File:	C:/temp/fdcache.ear
CDD File:	fdcache.cdd
Processing Unit:	default
TRA File (optional):	
Agent Name (optional):	inference-class
Description (optional):	
Concept TestData (optional):	/TestData/TestDataAcc
Event TestData (optional):	

The selected project does not contain the JUnit library on its classpath. Click here to add it now

4. In the next page of the wizard, select the test that you want the unit test suite to perform and click **Finish**.



New BEUnit Test Wizard

BusinessEvents Unit Test Suite Details

Select from the following common options what you would like to test. You may add other tests after this wizard completes.

What would you like to test?

- [x] Test whether a particular Rule fired
- [ ] Test the order of Rule Execution
- [ ] Test whether an Event fired
- [x] Test whether a Concept was modified
- [ ] Test whether a Concept or Event or Scorecard is in working memory

- [ ] Select/Deselect All

The test suite Java file with name specified in **File Name** is created with code required for the test suite. Edit the file according to your requirement.

5. If you are using Legacy ActiveSpaces, then you have to enter the environment variable in the JUnit test **Run Configurations** in BusinessEvents Studio to identify the class definitions based on your operating system.

    a. In BusinessEvents Studio, click **Run > Run Configurations**.

    b. In the Run Configurations window, right-click **JUnit** and select **New**.

    c. Select the **Environment** tab, and click **New**.

    d. Now, based on your platform, enter the name and value of the environment variable and click **OK**.

        - For Windows, add `PATH` environment variable.
        - For Linux, add `LD_LIBRARY_PATH` environment variable.
        - For Mac OSX, add `DYLD_LIBRARY_PATH` environment variable.

        You can also copy the environment variable value from the launch configuration of the BusinessEvents application. See Adding and Working with Launch (Debug or Run) Configurations for more details.

        If you use Apache Maven for build and deployment of the BusinessEvents application with Legacy ActiveSpaces, perform similar steps for setting environment variables in Maven build **Run Configuration** for JUnit test suite to work in Maven.

Sample Unit Test Suite File

The following is a sample unit test suite `.java` file generated for the CreditCardApplication project to test if a particular event has been asserted.

```
package;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import java.util.List;
import com.tibco.cep.runtime.model.event.SimpleEvent;
import com.tibco.cep.runtime.service.tester.beunit.BETestEngine;
import com.tibco.cep.runtime.service.tester.beunit.Expecter;
import com.tibco.cep.runtime.service.tester.beunit.TestDataHelper;
import com.tibco.cep.runtime.session.RuleServiceProvider;

/**
 * @description
 */
public class BEUnitTestSuite2 {
       private static BETestEngine engine;
       private static TestDataHelper helper;
       private static Expecter expecter;

       @BeforeClass
       public static void setUpBeforeClass() throws Exception {
               engine = new BETestEngine
("C:/tibco/BE54v95/be/5.4/examples/standard/WebStudio/CreditCardApplicat
ion.ear", "C:/tibco/BE54v95/be/5.4/bin/be-engine.tra",
"Deployments/CreditCardApplication.cdd", "default", "inference-class",
true);

               // Start the test engine
               engine.start();

               // Create a helper to work with test data
               helper = new TestDataHelper(engine);

               // Create an Expecter object to test rule execution,
modifications, assertions, etc.
               expecter = new Expecter(engine);
       }

       @AfterClass
       public static void tearDownAfterClass() throws Exception {
               try {
                       engine.shutdown();
               } catch (Exception localException) {
               }
       }

       @Before
       public void setUp() throws Exception {
       }

       @After
       public void tearDown() throws Exception {
       }
       /**
       * Test whether a particular Event was asserted by the engine during
rule execution
       */
       @Test
       public void testEventAsserted() throws Exception {
               engine.resetSession(); // (optional) reset the rule session,
which will clear working memory, restart timers, and clear the data from
any previous tests

               // TODO : Change test data path here to
create events to be asserted from a test data file
               List<SimpleEvent> events = helper.createEventsFromTestData
("/TestData/<test data file name>");
               if (events.size() > 0) {
                       engine.assertEvent(events.get(0), false);
               }
               engine.executeRules();
               expecter.expectEventAsserted("/Events/<event name>");
       }

}
```

What to do next

Right-click the new BEUnit test suite `.java` file and select **Run as > JUnit Test** to execute the test suite.

Shared Resources

A set of resources called Shared Resources can be used for various purposes in your projects.

Note: A RuleServiceProvider Configuration shared resource is available In TIBCO BusinessEvents Studio. It can be used during migration to display a migrated version 3.x resource for informational purposes. For integration with TIBCO ActiveMatrix BusinessWorks projects, use the RuleServiceProvider Configuration activity, available in the TIBCO Designer palette of TIBCO BusinessEvents Activities. See TIBCO ActiveMatrix BusinessWorks 6 Integration.

Adding a Shared Resource

Tip: It is a good idea to use global variables in shared resources so that projects can be quickly adapted to run in different environments.

Procedure

1. In BusinessEvents Studio Explorer, right-click the folder where you want to store the shared resource and select **New > Other**.
2. In the Select a Wizard dialog, expand TIBCO Shared Resources and select the resource type you want to add (see list in *Step 3*) and click **Next**.

3. In the new resource wizard enter a name in the **File Name** field and click **Finish**.

   You see the editor for the shared resource you selected. See the following sections for guidelines on completing the fields:

   - Legacy ActiveSpaces Connection Reference
   - Hawk Connection Reference
   - HTTP Connection Reference
   - Identity Resource Reference
   - JDBC Connection Reference, and Enabling the Test Connection Feature
   - JMS Application Properties
   - JMS Connection Reference, and Enabling the Test Connection Feature
   - JNDI Configuration Reference
   - FTL Connection Reference
   - ActiveSpaces Connection Reference

4. (Optional) View the **Source** tab. The source tab shows the XML source format for the shared resource. You can edit the XML source directly, but this is not recommended because of the risk of error. The resource editor provides validation checks.

Enabling the Test Connection Feature

To make the Test Connection feature work for JMS Connection and JDBC Shared Connection shared resources, copy the relevant JAR files to the following directory:

```
BE_HOME/lib/ext/tpcl
```

(This directory is referenced in the extended classpath in the file BE_HOME`/studio/eclipse/configuration/studio.tra`.)

Note: If TIBCO BusinessEvents Studio is running when you copy the file or files, you must restart it for the test connection feature to work.

For TIBCO Enterprise Message Service copy the `jms-2.0.jar` and `tibjms.jar` to the above location.

For WebSphere MQ, copy the MQ JAR files and the binding file to the above location.

For DBMS products, copy the supported driver file you are using for the DBMS product.

Legacy ActiveSpaces Connection Reference

The Legacy ActiveSpaces Connection resource describes the connection to a Legacy ActiveSpaces metaspace. This section provides a reference to the fields. For procedures see Adding a Shared Resource.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers shared resource identifiers can have spaces in the name. The name then appears in the title of the resource.
Configuration		
Description	No	Short description of the resource.
Metaspace Name	Yes	Specifies a Legacy ActiveSpaces metaspace. A metaspace is an instance of a cluster of application processes deployed using Legacy ActiveSpaces. The application processes are typically deployed on multiple hosts that are interconnected by a network. The default value for this field is ’ms’.

Field	Global Var?	Description
MemberName	Yes	Name the AS channel will take when connecting to Legacy ActiveSpaces metaspace
Discovery URL	Yes	Specifies how a metaspace instance discovers the current metaspace members. Multicast discovery can use PGM - Pragmatic General Multicast protocol. If using PGM protocol, the multicast URL is expressed in the following format: `tibpgm://`*`destination port`*`/`*`interface`*`;`*`discovery group address`*`/`*`optional transport arguments`*, where • *`destination port`* specifies the destination port used by the PGM transport. If not specified, the default value of 7888 is used. • *`interface`*`;`*`discovery group address`* specifies the address of the interface to be used for sending discovery packets, and the discovery group address to be used. If not specified, it defaults to the default interface and discovery address, 0.0.0.0;239.8.8.8. • *`optional transport arguments`* specifies a semicolon-separated list of optional PGM transport arguments. By default, the PGM transport is tuned to provide the best performance according to the most common deployment architectures. The values of those optional arguments should be changed only when necessary and with care, since any inappropriate values could easily result in degraded performance of the product.
ListenUrl	Yes	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. This address and port are specified by the member's Listen URL. If not specified, the discovery process uses the default IP

Field	Global Var?	Description
		address and the first free TCP port that can be acquired from the operating system (starting 5000 and above).
RemoteListenUrl	Yes	This field is used to configure Legacy ActiveSpaces channel as a remote-discovery proxy. In this case, any remote client can connect to an Legacy ActiveSpaces metaspace via the Legacy ActiveSpaces Channel node.
EnableSecurity	Yes	Enables security for the Legacy ActiveSpaces connection resource when selected. **Note:** Some fields are activated only for the specific security role or authorization policy.
SecurityRole	Yes	Security role of a node for the secure Legacy ActiveSpaces channel in the metaspace. The values are: • Controller is dedicated to enforcing a security domain's defined security behavior for a metaspace associated with the security domain. Security domain controllers are the only discovery nodes in a metaspace. • Requester just requires access to the data in the data grid, such as a seeder or a leech, and which need to be authorized by a controller. Requesters can never be used as discovery nodes. The controller nodes are configured with a security policy file. The requester nodes 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. If the Controller option is selected, then the following fields become active: • **Identity Password** • **PolicyFile**

Field	Global Var?	Description
		If the Requester option is selected, then the following fields become active: • **Identity Password** • **TokenFile** • **Credential**
Identity Password	Yes	The password for the identity key in the security policy file.
PolicyFile	Yes	Absolute path to the policy file which contains the security settings that the controller node enforces. It is generated using the as-admin utility.
TokenFile	Yes	Absolute path to the token file which is used by requester to connect to a metaspace whose security is defined in the policy file.
Credential	Yes	Authentication policy to be used for authentication as specified in the policy file. The values are: • USERPWD - A username and password based authentication is used. It activates the following fields: ◦ **Domain** ◦ **Username** ◦ **Password** • X509V3 - The authentication source is an LDAP configured with certificate based authentication. It activates the following fields: ◦ **KeyFile** ◦ **PrivateKey**
Domain	Yes	Domain name for system based user authentication.

Field	Global Var?	Description
Username	Yes	Username for LDAP and system based authentication.
Password	Yes	Password for LDAP and system based authentication.
KeyFile	Yes	The absolute path for a file containing the key to use for LDAP with the certificate based authentication.
PrivateKey	Yes	The password for the identity key in the LDAP identity file specified in **KeyFile**.

Hawk Connection Reference

The Hawk Connection resource describes the connection to a TIBCO Hawk domain through a specific transport. This section provides a reference to the fields. For procedures see Adding a Shared Resource.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.

Field	Global Var?	Description
		The name then appears in the title of the resource.
Configuration		
Description	No	Short description of the resource.
Transport	Yes	Type of transport used by the Hawk domain: Rendezvous or EMS.
Domain	Yes	Name of the Hawk domain.
Transport: Rendezvous		
RV_Service	Yes	Specifies the User Datagram Protocol (UDP) service group used by the TIBCO Rendezvous daemon for session communications. A service can be specified either by its name or its port number. The default configuration uses the service port number 7474.
RV_Network	Yes	Specifies the network to be used for outbound session communications when a computer is connected to more than one network. A network can be specified by its name or by its IP address.
RV_Daemon	Yes	Specifies the TIBCO Rendezvous daemon that will handle communication for the session. A local daemon is specified by the communications type (always tcp) and a socket number (e.g., 7474). The default configuration uses the local daemon with the TCP socket number 7474. Specify a remote daemon by inserting its host name or IP address between the tcp entry and the port number of the daemon parameter, for example: tcp:*remote_computer*:7800
Transport: EMS		
EMS_	Yes	Specifies the EMS server to connect to. The server URL is

Field	Global Var?	Description
ServerURL		typically provided in the following format: `protocol://hostname:port-number` For example, `tcp://myhost:7222`
EMS_ UserName	Yes	Specifies the username used to connect to the EMS server.
EMS_ Password	Yes	Specifies the password for the username used to connect to the EMS server.

HTTP Connection Reference

The HTTP Connection resource describes the characteristics of the connection used to receive incoming HTTP requests. This section provides a reference to the fields. For procedures see Adding a Shared Resource.

The HTTP Connection resource can specify that the HTTPS (secure sockets layer or SSL) protocol must be used by clients. If this is enabled, you can configure the SSL parameters for the HTTP server using the Configure SSL Button. See SSL Configuration for more information.

Warning: If you have multiple HTTP Connection resources specified by multiple HTTP Receiver process starters, the HTTP servers require that all of the connections must be valid to initialize all HTTP Receivers. Therefore, make certain that all HTTP Connection resources have valid configurations before testing or deploying the project.

The HTTP connection editor have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name. This field appears on the New HTTP Connection Wizard. The name then appears in the title of the resource.
Configuration		
Description	Yes	Short description of the resource.
Host	Yes	Specifies the name of the host that accepts the incoming requests. For machines that have only one network card, the default value `localhost` specifies the current machine. For machines that have more than one network card, this field specifies the host name of the card that are used to accept incoming HTTP requests. **Note:** Only one HTTP server can be started on each port. Therefore, if you have a machine with multiple network cards, make certain that all HTTP Connection resources that use the same host name specify different port numbers. **Note:** If there is more than one network card on the machine, and you specify `localhost` in this field, then all network cards on the machine listen for incoming HTTP requests on the specified port.
Port	Yes	Port number on which to listen for incoming HTTP requests.
Use SSL	No	Specifies whether incoming requests must use the HTTPS (secure

Field	Global Var?	Description
		socket layer, or SSL) protocol. This protocol authenticates the server to the client, and optionally authenticates the client to the server. By enabling this field you can specify SSL parameters with the Configure SSL button (see SSL Configuration).

SSL Configuration

Using the SSL Configuration for HTTPS Connections dialog (accessed by configuring the Configure SSL button) you can specify the SSL parameters for the HTTP connection.

The following are the fields in the SSL Configuration for HTTPS Connections dialog:

Field	Description
Requires Client Authentication	Selecting this field requires clients to present their digital certificate before connecting to the HTTP server. When this field is selected, the Trusted Certificates Folder becomes enabled so that you can specify a location containing the list of trusted certificate authorities.
Trusted Certificates Folder	This field is only applicable when the Requires Client Authentication field is selected. This field specifies a folder in the project containing one or more certificates from trusted certificate authorities. This folder is selected when a client connects to ensure that the client is trusted. This prevents connections from rogue clients.
Identity	This is an Identity resource that contains the HTTP server’s digital certificate and private key. See Identity Resource Reference for more information.
Trust Store	Specifies the password for the truststore.

Field	Description
Password	**Note:** There are no restrictions on the password that you use. Do not keep this field empty.
Strong Cipher Suites Only	When selected, this field specifies that the minimum strength of the cipher suites used can be specified with the `bw.plugin.security.strongcipher.minstrength` custom engine property. See *TIBCO ActiveMatrix BusinessWorks Administration* documentation for more information about this property. The default value of the property disables cipher suites with an effective key length below 128 bits. When this field is not selected, only cipher suites with an effective key length of up to 128 bits can be used.

Identity Resource Reference

The Identity resource encapsulates information that may be used to authorize a user, connection, and so forth. The information you supply changes depending on the type of Identity resource you want to use. This section provides a reference to the fields. For procedures see Adding a Shared Resource.

The identity certificate location, its type, and password can be specified as global variables.

Wizard and Configuration Tab

The New Identity Resource Wizard and the Configuration tab of the Identity Resource have the following fields.

Field	Global Var?	Description
Wizard		

Field	Global Var?	Description
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers shared resource identifiers can have spaces in the name. This field appears in the New Identity Resource Wizard. The name then appears in the title of the resource.
General		
Description	Yes	Short description of the resource.
Type	No	The type of identity resource: Identify File, Certificate/Private Key, or Username/Password (the default). See sections below for details.
Identity File Use this option if the certificate includes the private key information in the same file.		
URL	Yes	Location of the certificate (which includes the private key).
File Type	No	Choose the certificate file type from the drop-down list: Entrust JCEKS JKS PEM PKCS12
Password	Yes	Password for the certificate.
Certificate/Private Key Identity		

Field	Global Var?	Description
Use this option if the private key and the certificate are in two separate files.		
Certificate URL	Yes	Location of the certificate. Click the browse icon or type in a URL.
Key URL	Yes	Location of the private key file associated with the certificate.
Key Password	Yes	Password used for private key.
Username/Password Use this option if you want to use a username and password for authentication and do not want to use a certificate.		
Username	Yes	Name of the user for this identity.
Password	Yes	Password for the user for this identity.

JDBC Connection Reference

The JDBC Connection resource describes a JDBC connection. JDBC connections are used with backing stores and with database concepts.

For procedures to add a JDBC shared resource, see Adding a Shared Resource.

For using Secure Sockets Layer protocol for JDBC connection between BusinessEvents and a database server, see Configuring SSL for JDBC Connection.

JDBC Connection Wizard and Configuration Tab

Field	Global Var?	Description
Wizard		

Field	Global Var?	Description
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.
Configuration		
Description	Yes	Short description of the resource.
Connection Type	Yes	Specifies the kind of JDBC connection you wish to create. The connection type can be one of the following: • JDBC • JNDI The type of connection determines the other configuration fields that appear.
JDBC Connection Type Configuration Fields **Tip:** Using global variables makes the project more portable.		
JDBC Driver	Yes	The name of the JDBC driver class. You can select from a list of drivers or enter a driver manually. Listed drivers are as follows: • `oracle.jdbc.OracleDriver (thin)` • `com.ibm.db2.jcc.DB2Driver` (supported for database concepts only) • `com.microsoft.sqlserver.jdbc.SQLServerDriver` • `com.mysql.jdbc.Driver` When you select a driver, the Database URL field is populated

Field	Global Var?	Description
		with a template for the URL for the driver.
Database URL	Yes	The URL to use to connect to the database. A template of the URL is supplied for the selected JDBC driver. You must supply the portions of the URL that are in angle brackets, for example, the host, port number, and database instance name. You can configure Single Client Access Name (SCAN) for Oracle database in a cluster with numerous nodes. For more information on configuring SCAN, see *TIBCO BusinessEvents Configuration Guide*.
Maximum Connections	Yes	The maximum number of database connections to allocate. The default maximum is 10. The minimum value that can be specified is 1. See Connection Pooling for more details, including related settings that override this setting.
User Name	Yes	User name to use when connecting to the database.
Password	Yes	Password to use when connecting to the database.
Login Timeout	Yes	Time (in seconds) to wait for a successful database connection. Only JDBC drivers that support connection timeouts can use this configuration field. If the JDBC driver does not support connection timeouts, the value of this field is ignored. Most JDBC drivers support connection timeouts.
JNDI Connection Type Configuration Fields		
JNDI DataSource Name	Yes	The JNDI name specified for the DataSource.
Use Shared JNDI	No	When this field is selected, the **JNDI Configuration** field is displayed, allowing you to choose a shared JNDI Configuration

Field	Global Var?	Description
Configuration		resource. When this checkbox is not selected, the following JNDI fields are displayed: • **JNDI Context Factory** • **JNDI Context URL** • **JNDI User Name** • **JNDI Password**
JNDI Configuration	No	This field only appears when the **Use Shared JNDI Configuration** field is selected. This field allows you to choose a JNDI Configuration shared resource that specifies the JNDI connection information.
JNDI Context Factory	No	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The initial context factory class for accessing JNDI. (`javax.naming.Context.INITIAL_CONTEXT_ FACTORY`). You can choose from the drop down list of supported classes, or you can type in a different InitialContextFactory class name.
JNDI Context URL	Yes	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The URL to the JNDI service provider (`javax.naming.Context.PROVIDER_URL`). An example URL is provided when one of the supported JNDI context factory classes is selected. See your JNDI provider documentation for the syntax of the URL.
JNDI User Name	Yes	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The user name to use when logging into the JNDI server

Field	Global Var?	Description
		(`javax.naming.Context.SECURITY_ PRINCIPAL`). If the JNDI provider does not require access control, this field can be empty.
JNDI Password	Yes	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The password for logging into the JNDI server (`javax.naming.Context.SECURITY_ CREDENTIALS`). If the JNDI provider does not require access control, this field can be empty.
Use SSL	Yes	Specifies whether you want to use secure sockets layer (SSL) protocol for connection to the database server. This protocol authenticates the server to the client, and optionally authenticates the client to the server. Select this field to specify SSL parameters using the **Configure SSL** button (see JDBC SSL Configuration Reference).

Connection Pooling

TIBCO BusinessEvents creates a pool of JDBC connections for every JDBC Connection shared resource that uses the JDBC connection type. The maximum size of this pool is specified by the **Maximum Connections** configuration field.

Resources such as backing stores and database concepts that use this JDBC Connection resource are given a connection from the pool. Once the maximum number of connections is reached, resources requesting a connection cannot proceed. Once a connection is freed by an activity, the connection is returned to the pool. Connections that are left open eventually times out and closed. These connections can be reopened at a later time, until the maximum number of connections specified in this field is reached.

For backing store connections, you can use additional connection pool properties, which override equivalent settings in the JDBC Connection resource.

Test Connection Button

Using the **Test Connection** button you can test the connection specified in the configuration of this resource.

See Enabling the Test Connection Feature for a step you must take to enable the connection to work.

JDBC Connection Wizard and Configuration Tab

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.
Configuration		
Description	Yes	Short description of the resource.
Connection Type	Yes	Specifies the kind of JDBC connection you wish to create. The connection type can be one of the following: • JDBC • JNDI The type of connection determines the other configuration fields that appear.
JDBC Connection Type Configuration Fields		

Field	Global Var?	Description
Tip: Using global variables makes the project more portable.		
JDBC Driver	Yes	The name of the JDBC driver class. You can select from a list of drivers or enter a driver manually. Listed drivers are as follows: • `oracle.jdbc.OracleDriver (thin)` • `com.ibm.db2.jcc.DB2Driver` (supported for database concepts only) • `com.microsoft.sqlserver.jdbc.SQLServerDriver` • `com.mysql.jdbc.Driver` When you select a driver, the Database URL field is populated with a template for the URL for the driver.
Database URL	Yes	The URL to use to connect to the database. A template of the URL is supplied for the selected JDBC driver. You must supply the portions of the URL that are in angle brackets, for example, the host, port number, and database instance name. You can configure Single Client Access Name (SCAN) for Oracle database in a cluster with numerous nodes. For more information on configuring SCAN, see *TIBCO BusinessEvents Configuration Guide*.
Maximum Connections	Yes	The maximum number of database connections to allocate. The default maximum is 10. The minimum value that can be specified is 1. See Connection Pooling for more details, including related settings that override this setting.
User Name	Yes	Username to use when connecting to the database.
Password	Yes	Password to use when connecting to the database.

Field	Global Var?	Description
Login Timeout	Yes	Time (in seconds) to wait for a successful database connection. Only JDBC drivers that support connection timeouts can use this configuration field. If the JDBC driver does not support connection timeouts, the value of this field is ignored. Most JDBC drivers support connection timeouts.
JNDI Connection Type Configuration Fields		
JNDI DataSource Name	Yes	The JNDI name specified for the DataSource.
Use Shared JNDI Configuration	No	When this field is selected, the **JNDI Configuration** field is displayed, allowing you to choose a shared JNDI Configuration resource. When this checkbox is not selected, the following JNDI fields are displayed: • **JNDI Context Factory** • **JNDI Context URL** • **JNDI User Name** • **JNDI Password**
JNDI Configuration	No	This field only appears when the **Use Shared JNDI Configuration** field is selected. This field allows you to choose a JNDI Configuration shared resource that specifies the JNDI connection information.
JNDI Context Factory	No	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The initial context factory class for accessing JNDI. (`javax.naming.Context.INITIAL_CONTEXT_ FACTORY`). You can choose from the drop down list of supported classes, or you can type in a different InitialContextFactory class name.

Field	Global Var?	Description
JNDI Context URL	Yes	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The URL to the JNDI service provider (`javax.naming.Context.PROVIDER_URL`). An example URL is provided when one of the supported JNDI context factory classes is selected. See your JNDI provider documentation for the syntax of the URL.
JNDI User Name	Yes	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The username to use when logging into the JNDI server (`javax.naming.Context.SECURITY_ PRINCIPAL`). If the JNDI provider does not require access control, this field can be empty.
JNDI Password	Yes	This field only appears when the **Use Shared JNDI Configuration** field is not selected. The password for logging into the JNDI server (`javax.naming.Context.SECURITY_ CREDENTIALS`). If the JNDI provider does not require access control, this field can be empty.
Use SSL	Yes	Specifies whether you want to use secure sockets layer (SSL) protocol for connection to the database server. This protocol authenticates the server to the client, and optionally authenticates the client to the server. Select this field to specify SSL parameters using the **Configure SSL** button (see JDBC SSL Configuration Reference).

JDBC SSL Configuration Reference

Enter SSL parameters for a secure JDBC connection. Click **Configure SSL** in JDBC Configuration to display the SSL Configuration for JDBC page.

SSL Configuration for JDBC

Field	Global Variable?	Description
Requires Client Authentication	Yes	Two-way SSL requires client side authentication. Selecting this field requires clients to present their digital certificate before connecting to the database server. The **Identity** field becomes active when this field is selected, so that you can specify the identity file.
Trusted Certificates Folder	Yes	Path to the trusted certificates folder on the client machine. The trusted certificates are a collection of certificates from servers with whom you establish connections. If the server you wish to establish a connection with, presents a certificate that does not match one of your trusted certificates, the connection is refused. This prevents connections to unauthorized servers. You can store the trusted certificates outside your project using a global variable. For details, see Storing Trusted Certificates Outside of Your Project.
Identity	No	If you want two-way SSL, select the **Required Client Authentication** field to activate this field. Input the location of the identity shared resource file. Two-way SSL requires an identity file that contains the information to authenticate BusinessEvents client identity. See Identity Resource Reference for more information.
Trust Store Password	Yes	Specifies the password for the truststore. **Note:** There are no restrictions on the password that you use. Do not keep this field empty.

Field	Global Variable?	Description
Verify Host Name	Yes	Specifies whether to verify the host name
Expected Host Name	Yes	Select the **Verify Host Name** field to activate this field. Specifies the host name for verification.

Configuring SSL for JDBC Connection

Using the Secure Sockets Layer (SSL) protocol, you can establish a secure communication between the JDBC client and DBMS servers. You can use TIBCO BusinessEvents Studio to configure SSL for a JDBC connection.

Refer the *TIBCO BusinessEvents Configuration Guide* for more information on JDBC backing store and database connections.

Before you begin

- Configure a database server for SSL. Refer to the respective DBMS documentation for server-side configuration steps.
- Copy the appropriate JDBC driver file to `BE_HOME/lib/ext/tpcl`.

Procedure

1. In TIBCO BusinessEvents Studio, open the JDBC shared resource for editing.

   See JDBC Connection Wizard and Configuration Tab for more details about JDBC fields.

2. Select the **Use SSL** checkbox for activating the SSL protocol for the JDBC connection.

   The **Configure SSL** button is activated.

3. Click **Configure SSL**.

   The SSL Configuration for JDBC window is displayed.

4. Enter the values for the SSL parameters and click **OK**.

   See JDBC SSL Configuration Reference for more details about these parameters.

5. Save the JDBC shared resource.

   The JDBC connection is now configured to use the SSL protocol.

   Oracle Database Configuration

6. *(Oracle database only)* When using the Oracle wallet or `.p12` file as keystore or truststore type, copy the following JAR files from the `jlib` directory in the Oracle installation to *BE_HOME*`/lib/ext/tpcl`:
   - `oraclepki.jar`
   - `osdt_cert.jar`
   - `osdt_core.jar`

Note: If you get the following exceptions when using JKS as a keystore or truststore type:

java.security.cert.CertPathValidatorException: Algorithm constraints check failed: MD5withRSA
then perform the following steps:

Oracle 11g

- Remove `MD5` from the following property value in the `java.security` file:

```
 jdk.certpath.disabledAlgorithms=MD2,MD5,RSA
keySize < 1024
```

- Remove `MD5withRSA` from the following property value in the `java.security` file:

```
  jdk.tls.disabledAlgorithms=SSLv3, RC4,
MD5withRSA, DH keySize < 768
```

- Store the root certificate in the truststore folder.

Oracle 12c

Create wallets and certificates with SHA-256 or other than MD5withRSA as signing algorithm. Now create the JKS keystore and truststore using these wallets.

See SSL Connection to Oracle DB for more information.

Connection Pooling

TIBCO BusinessEvents creates a pool of JDBC connections for every JDBC Connection shared resource that uses the JDBC connection type. The maximum size of this pool is specified by the **Maximum Connections** configuration field.

Resources such as backing stores and database concepts that use this JDBC Connection resource are given a connection from the pool. Once the maximum number of connections is reached, resources requesting a connection cannot proceed. Once a connection is freed by an activity, the connection is returned to the pool. Connections that are left open get eventually time out and be closed. These connections can be reopened at a later time, until the maximum number of connections specified in this field is reached.

For backing store connections, you can use additional connection pool properties, which override equivalent settings in the JDBC Connection resource.

TIBCO BusinessEvents version 6.x onwards the default database pool for DBConcepts is JDBC. To set this to Oracle, set the property `be.dbconcepts.use.oracle.pool` to `true` in the CDD file for your project.

Test Connection Button

Using the **Test Connection** button you can test the connection specified in the configuration of this resource. See Enabling the Test Connection Feature for a step you must take to enable the connection to work.

JMS Application Properties

The JMS Application Properties resource describes any JMS message properties that a JMS application expects. These properties can then be added.

For procedures see Adding a Shared Resource.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.
Description	Yes	Short description of the resource.
Configuration		
Properties Table A table listing any application-specific properties. Use the + and x buttons to the right of the table to add and delete properties. Use the up and down arrow buttons to move selected properties to the desired location in the table.		
Property Name	Yes	Name of the column.
Type	Yes	The data type of the property. Double-click the cell to cause a drop down list of valid JMS datatypes to appear, and choose a value.
Cardinality	No	Specifies whether the property is optional or required. Double-click the cell to cause a drop down list of two values to appear and select optional or required.

JMS Connection Reference

JMS Connection resource describes a JMS connection. This section provides a reference to the fields. For procedures, see Adding a Shared Resource.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers shared resource identifiers can have spaces in the name. This field appears on the New JMS Connection Wizard. The name then appears in the title of the resource.
Configuration Tab		
Description	Yes	Short description of the resource.
User Name	Yes	Username to use when logging into the JMS server. If the JMS provider does not require access control, this field can be empty. Not all JMS servers require user names and passwords. Refer to your JMS provider documentation and consult your system administrator to determine if your JMS server requires a user name and password.
Password	Yes	Password to use when logging into the JMS server. If the JMS provider does not require access control, this field can be empty.
Auto-Generate Client ID	No	Selecting this field specifies you wish to automatically generate the client ID if no client ID is specified or if the specified ID is already in use. When this field is selected, if a value is specified

Field	Global Var?	Description
		in the Client ID field, an attempt is made to use the specified value. However, if the specified value is already in use, a new client ID is generated. If this field is not selected, then the value specified in the Client ID field is used. If no value is specified in the Client ID field, then no client ID is set.
Client ID	Yes	Client ID for the connection. Typically JMS providers have a provider-specific format for client IDs. See your JMS provider's documentation for more information about client IDs. Each connection must use a unique Client ID. You cannot use the same JMS Connection resource for accessing both topics and queues. You should create separate JMS Connection resources if you wish to access both topic and queue destinations.
Use SSL	No	Specifies whether you wish to use SSL for the connection to the JMS server. SSL is used when the Use SSL checkbox is checked. Click the **Configure SSL** button to configure the SSL connection parameters. **Note:** SSL is supported only when using TIBCO Enterprise Message Service. See Configure SSL for more information.
Use JNDI for Connection Factory	Yes	Specifies whether JNDI should be used to look up the ConnectionFactory object. If this field is not selected, the Provider URL and Use XA Connection Factory fields appear. If this field is selected, JNDI configuration fields appear.
Provider URL	Yes	This field is only available when the Use JNDI for Connection Factory field is not selected. This is the URL to use to connect to the JMS server.

Field	Global Var?	Description
Connection Factory SSL Password	Yes	This field is only available when the Use SSL checkbox is selected, and the User Shared JNDI Configuration checkbox is not selected. The SSL configuration is specified in the ConnectionFactory object, except for the client SSL password. You can specify your client SSL password in this field, or you can leave this field empty. If your password is not specified, the private key password is used.
Use Shared JNDI Configuration	Yes	When this field is selected, the JNDI Configuration field appears. Using this you can choose a JNDI Configuration Reference . When this field is not selected, JNDI configuration fields appear.
JNDI Configuration	No	This field only appears when the Use Shared JNDI Configuration field is selected. This field allows you to choose a JNDI Configuration Reference that specifies the JNDI connection information.
JNDI Context Factory	Yes	This field only appears when the Use Shared JNDI Configuration field is not selected. Required. The initial context factory class for accessing JNDI. (`javax.naming.Context.INITIAL_CONTEXT_ FACTORY`). You can choose from the drop down list of supported classes or you can type in a different InitialContextFactory class name. **Note:** TIBCO BusinessEvents attempts to find the class. However, you may need to add the Java file supplied by your JNDI service provider to the CLASSPATH environment variable to use JNDI.
JNDI Context URL	Yes	This field only appears when the Use Shared JNDI Configuration field is not selected. Required. This is the URL to

Field	Global Var?	Description
		the JNDI service provider (`javax.naming.Context.PROVIDER_URL`). An example URL is provided when one of the supported JNDI context factory classes is selected. See your JNDI provider documentation for the syntax of the URL.
JNDI User Name	Yes	This field only appears when the Use Shared JNDI Configuration field is not selected. Username to use when logging into the JNDI server (`javax.naming.Context.SECURITY_ PRINCIPAL`). If the JNDI provider does not require access control, this field can be empty.
JNDI Password	Yes	This field only appears when the Use Shared JNDI Configuration field is not selected. Password to use when logging into the JNDI server (`javax.naming.Context.SECURITY_ CREDENTIALS`). If the JNDI provider does not require access control, this field can be empty.

Test Connection Button

Using the **Test Connection** button you can test the connection specified in the configuration of this resource. See Enabling the Test Connection Feature for a step you must take to enable the connection to work.

When JNDI is used (that is, when the Use JNDI for Connection Factory checkbox is selected), the Test Connection button works only when the JNDI-related fields on the **Configuration** and **Advanced** tab are correctly specified.

Advanced Tab

The **Advanced** tab has the following fields.

Field	Global Var?	Description
Topic Connection Factory	Yes	This field is only available when the Use JNDI for Connection Factory field on the **Configuration** tab is selected. The `TopicConnectionFactory` object stored in JNDI. This object is used to create a topic connection with a JMS application. See your JNDI provider documentation for more information about creating and storing `TopicConnectionFactory` objects. Specify the Topic Connection Factory name in this field. Default: `TopicConnectionFactory`
Queue Connection Factory	Yes	This field is only available when the Use JNDI for Connection Factory field on the **Configuration** tab is selected The `QueueConnectionFactory` object stored in JNDI. This object is used to create a queue connection with a JMS application. See your JNDI provider documentation for more information about creating and storing `QueueConnectionFactory` objects. Specify the Queue Connection Factory name in this field. Default: `QueueConnectionFactory`
Optional JNDI Properties	No	Any additional properties to supply for the connection. You specify a name, datatype, and value for each property. These properties are typically vendor-specific. See your JNDI provider documentation for more information about the available properties.

Configure SSL

The SSL Configuration button allows you to configure the SSL connection parameters.

Note: When using JNDI to lookup the JMS Connection factory, the parameters `ssl_identity` and `ssl_verify_host` must be specified in the `factories.conf` file of the Enterprise Message Service server.

The following table describes the SSL Configuration dialog.

Field	Description
Trusted Certificates Folder	Location of the trusted certificates on this machine. The trusted certificates are a collection of certificates from servers to whom you establish connections. If the server you wish to establish a connection presents a certificate that does not match one of your trusted certificates, the connection is refused. This prevents connections to unauthorized servers. Trusted certificates must be imported into a folder, and then you can select the folder in this field.
Identity	The location of the client certificate. You only need to specify the client certificate when the JMS server requires client authentication. See Identity Resource Reference for more information.
Trust Store Password	Specifies the password for the truststore. **Note:** There are no restrictions on the password that you use. Do not keep this field empty.
Trace	Specifies whether SSL tracing should be enabled during the connection. If selected, the SSL connection messages are logged and sent to the console.
Debug Trace	Specifies whether SSL debug tracing should be enabled during the connection. Debug tracing provides more detailed messages than standard tracing.
Verify Host Name	Specifies whether you wish to verify that the host you are connecting to is the

Field	Description
	expected host. The host name in the host’s digital certificate is compared against the value you supply in the Expected Host Name field. If the host name does not match the expected host name, the connection is refused. **Note:** The default context factories for TIBCO Enterprise Message Service automatically determine if host name verification is necessary. If you are using a custom implementation of the context factories, your custom implementation must explicitly set the verify host property to the correct value. For example: `com.tibco.tibjms.TibjmsSSL.setVerifyHost(false)`
Expected Host Name	Specifies the name of the host you are expecting to connect to. This field is only relevant if the Verify Host Name field is also selected If the name of the host in the host’s digital certificate does not match the value specified in this field, the connection is refused. This prevents hosts from attempting to impersonate the host you are expecting to connect to.
Strong Cipher Suites Only	When selected, this field specifies that the minimum strength of the cipher suites used can be specified with the `bw.plugin.security.strongcipher.minstrength` custom engine property. See TIBCO ActiveMatrix BusinessWorks Administration for more information about this property. The default value of the property disables cipher suites with an effective key length below 128 bits. When this field is not selected, only cipher suites with an effective key length of up to 128 bits can be used.

JNDI Configuration Reference

The JNDI Configuration shared resource provides a way to specify JNDI connection information that can be shared by other resources. This resource can be specified in any resource that permits JNDI connections. For example, JDBC Connection Reference can use JNDI connections. This section provides a reference to the fields. For procedures see

Adding a Shared Resource.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements. Unlike other resource identifiers shared resource identifiers can have spaces in the name. This field appears on the New JNDI Connection Wizard. The name then appears in the title of the resource.
Configuration		
Description	Yes	Short description of the resource.
JNDI Context Factory	Yes	The initial context factory class for accessing JNDI. (`javax.naming.Context.INITIAL_CONTEXT_ FACTORY`). You can choose from the drop down list of supported classes. **Note:** TIBCO BusinessEvents attempts to find the class. However, you may need to add the JAR file supplied by your JNDI service provider to the CLASSPATH environment variable to use JNDI.
JNDI Context URL	Yes	The URL to the JNDI service provider (`javax.naming.Context.PROVIDER_URL`). An example URL is provided when one of the supported JNDI context factory classes is selected.

Field	Global Var?	Description
		See your JNDI provider documentation for the syntax of the URL.
JNDI User Name	Yes	Username for logging into the JNDI server (`javax.naming.Context.SECURITY_PRINCIPAL`). If the JNDI provider does not require access control, this field can be empty.
JNDI Password	Yes	Password for logging into the JNDI server (`javax.naming.Context.SECURITY_CREDENTIALS`). If the JNDI provider does not require access control, this field can be empty.

Advanced Section

The Advanced section has the following fields.

Field	Global Var?	Description
Validate JNDI Security Context	No	Some application servers store the security context on the thread used to establish the JNDI connection (at the time of this release, only the WebLogic application server does this). In that case, the first activity to use this resource establishes the security context, then subsequent activities use the same security context, unless this field is selected. Selecting this field ensures that each activity that uses this resource examines the security context to determine if the activity uses the same security context as the security context established on the thread. If they are different, the activity's configured security context is used. Selecting this field causes additional overhead for activities that use this resource. Therefore, only select this field when necessary.
Optional JNDI Properties	No	The table in this field contains optional properties to pass to the JNDI server. Use the +, X, and arrow keys to add, delete, and move properties in the list. Each property requires the property name, the datatype for the property, and the value for the property. See

Field	Global Var?	Description
		the documentation for your JNDI provider for more information about properties that can be passed to the JNDI server.

FTL Connection Reference

The TIBCO FTL and TIBCO BusinessEvents connection details are configured in the shared resource.

For the procedure to add an FTL connection, see Adding a Shared Resource.

Field	Global Var?	Description
Description	No	Short description of the resource.
RealmServer	Yes	The URL at which BusinessEvents can connect to the FTL realm server. The default value is: http://localhost:8080
UserName	Yes	A valid username for the FTL realm server. **Note:** This field is required only when the basic authentication is enabled at the realm server.
Password	Yes	The password assigned to the user name specified in **UserName**, for accessing the FTL realm server. **Note:** This field is required only when the basic authentication is enabled at the realm server.
Use SSL	Yes	Specifies whether secured channel is used for communication. After selecting the **Use SSL** checkbox, click **Configure SSL** to

Field	Global Var?	Description
		specify SSL configuration for the FTL channel. Select one of the following option in SSL configuration: • **Trust File** - Browse and select the identity resource that contains the information to authenticate BusinessEvents client identity. If not already present, first create an identity resource with the following values and select that identity resource: ◦ **Type** - `Identity File` ◦ **URL** - Location of the FTL trust file (`.pem`) ◦ **File Type** - `PEM` For more information on how to create shared identity resource, see Adding a Shared Resource. • **Trust String** - Enter the content for the trust file in the same format as applicable for the trust file. • **Trust Everyone** - No trust file is required.

ActiveSpaces Connection Reference

ActiveSpaces connection resource describes an ActiveSpaces connection. This section provides a reference to the fields in an ActiveSpaces connection. For the procedure to add an ActiveSpaces connection shared resource see Adding a Shared Resource

Configuration Tab for ActiveSpaces Channel

The configuration tab has the following fields:

ActiveSpaces Channel Properties

Field	Global Var?	Description
Description	No	Short description of the resource
Realm Server URL	Yes	The URL at which the TIBCO BusinessEvents can connect to the TIBCO FTL realm server. **Default URL:** http://localhost:8080
Grid Name	Yes	Name of the data grid to which the ActiveSpaces channel connects
User Name	Yes	A valid user name for the TIBCO FTL realm server
Password	Yes	The password assigned to the user name specified in **UserName**, for accessing the TIBCO FTL realm server
Use SSL	No	Check this box if you want to use secure socket layer (SSL) protocol while connecting to the TIBCO FTL realm server. If checked, the **Trust Type** and **Identity** fields are activated
Trust Type	Yes	Specifies whether you want to use a trust file for SSL authentication
Identity	No	The **Identity** field is activated if you have selected the **Trust File** option in the **Trust Type** dropdown. Browse to select the trust file from your system.

Persistence Store Setup

A persistence store enables persistent backup of the objects generated and modified at runtime. Use of a backing store enables recovery in the event of a system-wide failure.

The backing store feature requires use of Cache or Store object management. Before you add a backing store, ensure that your project ontology is completely configured.

Note:

- Use a separate schema and schema owner for each project, even if different projects use the same ontology (otherwise ontology conflicts can occur).
- If your project ontology changes after the backing store is in place, you must update the backing store schema. See Updating Existing Backing Store Schema.

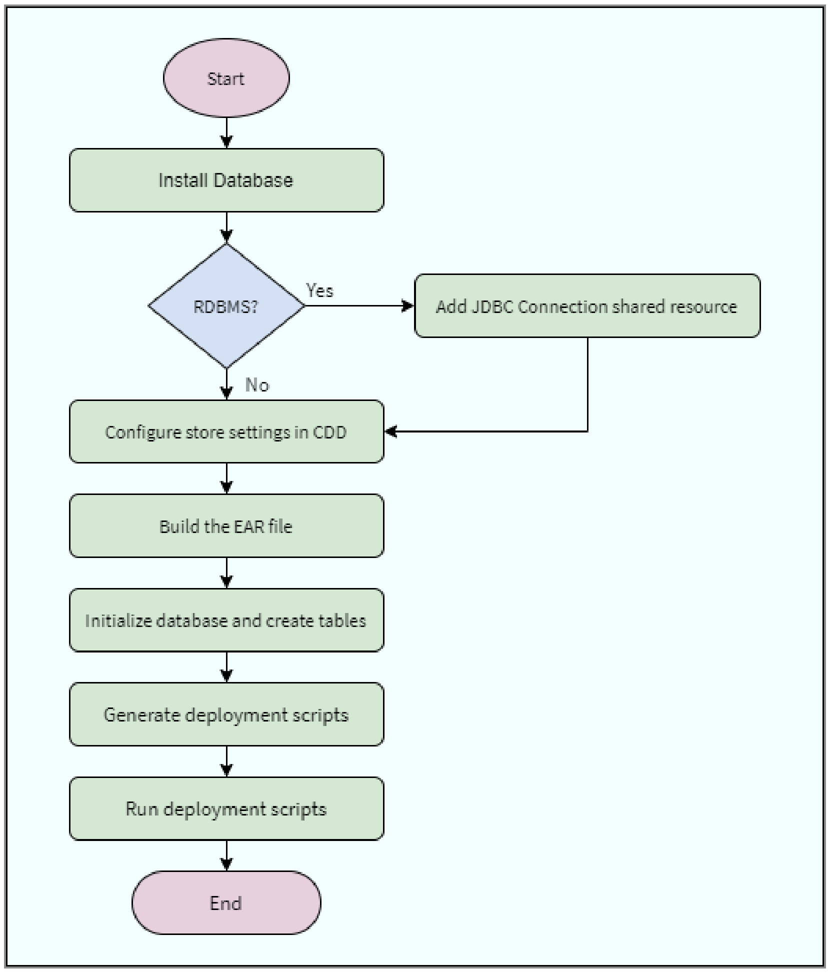
Setup refers to using the provided scripts to create the backing store schema for your project. Also, see Resources Required for Setting Up the Database for the DBMS-related requirements.

See the following flowcharts to understand the store setup procedure:

- Store Setup (Simple)
- Store Setup (Complex)

Store Setup (Simple)

Figure 20: Simple task flow


Start
Install Database
RDBMS?
Yes
Add JDBC Connection shared resource
No
Configure store settings in CDD
Build the EAR file
Initialize database and create tables
Generate deployment scripts
Run deployment scripts
End

Procedure

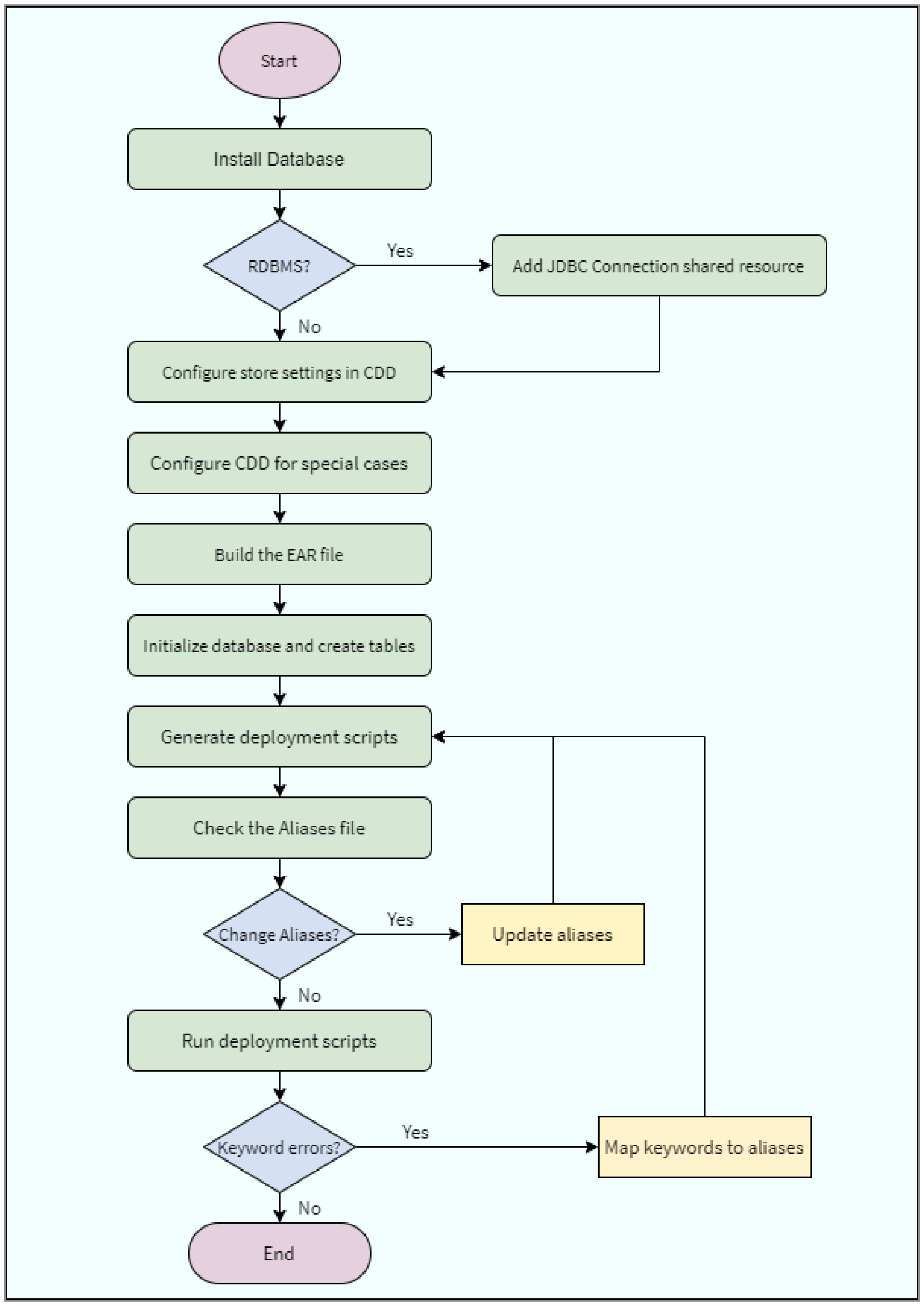
1. *(JDBC backing store only)* Add a JDBC Connection shared resource, see JDBC Connection Reference.
2. Configure backing store settings and properties in the CDD file, see *TIBCO BusinessEvents Configuration Guide*.
3. Prepare and build the EAR file, see *TIBCO BusinessEvents Administration*.
4. Initialize the backing store database and run SQL scripts to create tables, see Initializing the Database and Generate Non-Project Tables.

5. Generate the backing store deployment scripts for your project, see Generating Deployment Scripts for a Store.
6. Run the generated deployment scripts, see Run the Project Schema Script (as BE_USER).

Note: Task D is not available for DB2 since DB2 uses the OS runtime authentication system. Therefore, this step (run the initialize database script) does not apply to DB2.

Store Setup (Complex)

Figure 21: Complete task flow



Procedure

1. Install the prerequisites for the database software. See Install Prerequisites for Backing Stores.
2. *(JDBC backing store only)* Add a JDBC connection shared resource. See JDBC Connection Reference.
3. Configure backing store settings and properties in the CDD file. See *TIBCO BusinessEvents Configuration Guide*.
4. Configure the CDD file for special cases. See Cases That Need Additional Setup.
5. Prepare and build the EAR file. See *TIBCO BusinessEvents Administration*.
6. Initialize the backing store database and run SQL scripts to create tables. See Initializing the Database and Generate Non-Project Tables.
7. Generate the backing store deployment scripts for your project. See Generating Deployment Scripts for a Store.
8. Check and modify the alias file. See Check the Aliases File and Modify Aliases as Desired.
9. Run the generated deployment scripts. See Run the Project Schema Script (as BE_USER).
10. Map keywords to aliases. See If Needed - Map Key (Reserved) Words to Aliases.

Note: Task F is not available for DB2 since DB2 uses the OS runtime authentication system. Therefore, this step (run the initialize database script) does not apply to DB2.

Resources Required for Setting Up the Database

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

The file extension of the resource identifies the store type for which the resource is used:

- `.sql` - JDBC backing stores
- `.tibdg` - TIBCO ActiveSpaces

- `.cql` - Apache Cassandra

Provided Configuration Resources

The following table lists the resources available in TIBCO BusinessEvents at *BE_HOME*/`bin` for setting up stores.

Resources Required for JDBC Backing Store Implementation

Resource	Default Location and Notes
`base_types.xml`	The `base_types.xml` file is used by the deployment utility. Do not edit this file.
`be-storecdeploy.exe` `be-storedeploy.tra`	Only used for manual SQL script generation. Generally not needed. You can use a TIBCO BusinessEvents Studio option instead. See Generated SQL Scripts .

`create_tables_*.sql`
`create_tables_as.tibdg`
`create_tables_cassandra.cql`
	Use the appropriate SQL (DDL) script for your DBMS. This script creates the tables that are used to maintain the metadata.
	The script drops any existing tables and recreates them.

`dbkeywordmap.xml`	This file contains mappings to handle words used in the TIBCO BusinessEvents project that are database reserved words. See Ontology Identifiers that Use Database Key Words for details.

`initialize_database_<`*`database`*`>.sql`
`initialize_database_cassandra.cql`
	Use the appropriate script for your DBMS.
	By default the user is called `BE_USER` with the password `BE_USER` and the user has DBA rights. Edit the script if you want the user to have a different name or different rights.
	For SQL Server, this script also creates the default database, with the name `BE_USER` and makes it the default database for the user `BE_USER`.

Resource	Default Location and Notes
	Note: Use a different user (and schema) for every TIBCO BusinessEvents project that needs a backing store. This script drops the user (and therefore all the tables) and adds the user again.

Generated SQL Scripts

These scripts are generated when you use the **File > Export > TIBCO BusinessEvents > Backingstore Deployment > JDBC Deployment wizard**. The value for *yourname* is specified in the Output Script Name Prefix setting. You specify the location of the scripts in the wizard.

You can manually execute the `be-storedeploy` executable. You specify script name prefix at the command line. Scripts are generated in the same directory where you run `be-storedeploy`.

Generated SQL scripts

Script Filename	Description
yourname`.sql` *yourname*`.tibdg` *yourname*`.cql`	This SQL (DDL) script creates schema tables and types.
yourname`.aliases`	This file has entries if the database table identifiers are longer than the DBMS maximum character limit. See and Configuring Aliases File and Project Schema Script .
yourname`_alter.sql` *yourname*`_alter.tibdg` *yourname*`_alter.cql`	The *yourname*`_alter.sql` script is for use in schema migration. Generated only after updates are made to the `be-storedeploy.tra` file. See Updating Existing Backing Store Schema.
yourname`_delete.sql`	For use as needed. This script deletes the entities that have been marked as deleted in the object table.

Script Filename	Description
	Restriction: Not applicable for TIBCO ActiveSpaces and Apache Cassandra.
yourname`_cleanup.sql` *yourname*`_cleanup.cql`	For use as needed. This script truncates the tables. **Restriction:** Not applicable for TIBCO ActiveSpaces.
yourname`_remove.sql` *yourname*`_remove.tibdg` *yourname*`_remove.cql`	For use as needed. This script removes the database schema. You can use it to reset the project.
yourname`_migration.sql`	This script is generated to migrate existing database data from legacy ID format to new ID format for a project with shared all or store persistence configurations (with the new ID lookup implementations). When run on the existing database, the migration script copies the data to the new ID format by setting the extId as the primary key for all entities in database tables. The script adds a unique value wherever extId value is null for the rows in the existing database tables and also maintains all the reference and containment relationships of the data. **Note:** The former legacy ID tables are renamed with the suffix as `<TableName>_old`. You can delete these tables manually.

Install Prerequisites for Backing Stores

You must install prerequisites for the store (database) provider before you begin to configure the backing store.

Procedure

1. Install and start a supported store.

   See the product readme file for a list of supported products.

   See the store provider documentation for installation instructions.

2. (JDBC backing stores only) Copy the appropriate JDBC drivers file to `*BE_HOME*/lib/ext/tpcl`.

3. Restart TIBCO BusinessEvents Studio after copying the drivers file.

   This step is required before you can use the design-time Test Connection feature. It is also required for runtime.

4. Now or later: If you use the debugger or tester features, add your store libraries to the TIBCO BusinessEvents Studio classpath.

   The remainder of this section provides a few tips for each supported DBMS.

Microsoft SQL Server

Here are a few helpful points about SQL Server:

SQL Server authentication for non-production purposes

It is convenient to use SQL Server authentication so you can create database users as needed. Select this option when you install Microsoft SQL Server. With Windows Authentication, on the other hand, you may have difficulties creating users without help from others in your enterprise.

Availability Group

An availability group must be ready and dedicated to TIBCO BusinessEvents so that the a TIBCO BusinessEvents database or TIBCO BusinessEvents databases can be added to that group.

Note: It is required to create a full backup of the database before adding it the availability group.

SQL Server AlwaysOn Availability Groups

The AlwaysOn Availability Groups feature in SQL Server 2012 Enterprise Edition is a high-availability and disaster-recovery solution used by TIBCO BusinessEvents. To implement this feature, make sure that your SQL Server setup is up and running properly as per the Microsoft documentation.

Connection Properties to the SQL Server Cluster

Use the availability group listener defined at the SQL Server cluster level in the TIBCO BusinessEvents JDBC resource. The availability group listener enables clients to connect to a SQL Server replica without knowing the name of the physical instance of the SQL Server, for example:

```
jdbc:sqlserver://sqlserver-group-listener:1433;databaseName=be_user
```

To allow faster TCP connection retries, use the `property multiSubnetFailover=true` even if your cluster is on the same subnet as recommended by Microsoft,

Set this property in the BusinessEvents JDBC resource in the URL field, for example:

```
jdbc:sqlserver://sqlserver-group-listener:1433;databaseName=be_
user;multiSubnetFailover=true
To support Windows authentication, use property integratedSecurity=true
```

See SQL Server authentication vs. Windows authentication for details.

Configuring Your Machine for Windows Authentication

This authentication is only supported on Microsoft Windows operating systems. You must configure TIBCO BusinessEvents to support Windows authentication when accessing SQL Server database.

For more details SQL server and Microsoft Windows, see their documentation.

Procedure

1. Download and install the Microsoft JDBC 4 Driver distribution on the machines TIBCO BusinessEvents runs on.

   The distribution contains the `sqljdbc_auth.dll` used by the client to support Windows authentication.

2. Edit the `be-engine.tra` file and update the *tibco.env.PATH* variable so that it points to the folder where `sqljdbc_auth.dll` resides. For instance:

   ```
   tibco.env.PATH C:/sqljdbc_4.0/enu/auth/x64%PSP%%BE_
   HOME%/hotfix/bin%PSP%.
   ```

   Note: JDBC Driver distribution has an x64 or x86 version of that DLL file. Pick the same version as your TIBCO BusinessEvents installation.

3. Edit the `studio.ini` file and add the *java.library.path* variable so that it points to the folder where `sqljdbc_auth.dll` resides. For instance:

   ```
   -Djava.library.path=C:/sqljdbc_4.0/enu/auth/x64
   ```

   Note: JDBC Driver distribution has an x64 or x86 version of that dll. Pick the same version as your BusinessEvents installation.

4. Use the property `integratedSecurity=true` in the BusinessEvents JDBC resource. For instance:

   ```
   jdbc:sqlserver://sqlserver-group-listener:1433;databaseName=be_
   user;integratedSecurity=true
   ```

5. Replace the script in `<tibco_be_home>/bin/initialize_database_sqlserver.sql` with the following script that creates a user associated to Windows login:

```
use master
go
drop database be_user
go
create database be_user
go
drop login [domain\user]
go
create login [domain\user] from windows with default_database = be_
user
go
use be_user
create user [domain\user] for login [domain\user]
go
grant control, alter, connect to [domain\user]
go
alter role [db_owner] add member [domain\user]
go
```

SQL Server authentication vs. Windows authentication

As mentioned in the Microsoft documentation, you should use Windows domain logins to access databases that are members of availability groups.

Database users that are associated to domain login added to a database in a primary replica are propagated to secondary databases, and then continue to be associated with the specified domain login.

In the other hand, database users associated with an SQL Server login will be propagated to the secondary databases without a login. The user will not be able to access data from any secondary database.

To overcome this behavior, give the SQL Server login a higher permission at the server level such as `sysadmin` on all the SQL Server replicas, or use Windows login.

For Windows login refer to SQL Server AlwaysOn Availability Groups.

Datatype and Driver Information

The datetime datatype in SQL Server 2005 has the following range: 1/1/1753 to 12/31/9999. Microsoft SQL Server 2008 has added a new data type, `datetime2`, which has a date range of 0001/01/01 through 9999/12/31. Therefore, if you are using Microsoft SQL Server 2008, then you can manually change the generated SQL script (DDL) for your backing store, and replace any affected columns' data type from `datetime` to `datetime2`.

Use the SQL Server JDBC driver, `sqljdbc4.jar`. You can download this driver from Microsoft site. For details, see Microsoft SQL documentation.

Configuring Your Machine for Oracle Database

For configuring your machine for Oracle database, you need to install OCI drivers.

In production environments, you might have to ask a database administrator to create a database user for you. You should then be able to run the other SQL scripts yourself, logged on as the user created by the administrator.

Minimum User Permissions

By default the TIBCO BusinessEvents user permissions are set to DBA privileges. At a minimum, the user must be able to create tables and views. For example for an Oracle database you could use the following:

```
DROP USER BE_USER CASCADE;
CREATE USER BE_USER IDENTIFIED BY BE_USER;
GRANT CONNECT TO BE_USER;
GRANT RESOURCE TO BE_USER;
GRANT CREATE ANY VIEW TO BE_USER;
GRANT CREATE ANY TABLE TO BE_USER;
```

Installing an OCI Driver - OCI Driver Support

This procedure installs an OCI driver for the Oracle database.
Instructions assume you are working with a local database for testing purposes. Adapt the instructions if you are working with a remote database.

Procedure

1. Install an Oracle client: preferred as Admin/custom installation.
2. Admin/Custom installation creates the file `tnsnames.ora`.

   Configuration while creating the file `tnsnames.ora` should match the database server configuration (database server is running on a remote machine).
3. Install TIBCO BusinessEvents and add the file `ojdbc7.jar` from the client installation to the folder *BE_Home*`/lib/ext/tpcl`.
4. The connect URL used in the TIBCO BusinessEvents is `jdbc:oracle:oci:@DBServerHostName:1521:DBInstance.`

   This URL can be found in the file `tnsnames.ora`.
5. If the installation was performed properly, the studio JDBC test connection should show success.

Oracle Real Application Cluster (RAC)

Oracle Real Application Cluster (RAC) supports both thin and OCI drivers.

The sample connection string for a thin driver is:

```
jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=on)
(ADDRESS=(PROTOCOL=TCP)(HOST=10.107.146.70) (PORT=1521))
(ADDRESS=(PROTOCOL=TCP)(HOST=10.107.146.71) (PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=TIBQADB)))
```

Failover mode is supported with the basic and preconnect methods.

The sample connection string for an OCI driver is:

```
jdbc:oracle:oci:@net_service_name
```

When using an OCI driver, Transparent Application Failover (TAF) is supported.

You can configure a Single Client Access Name (SCAN) for an Oracle database in a cluster with numerous nodes. SCAN is a feature of Oracle Real Applications Clusters (RAC) 11g, which provides a single name for clients to access Oracle Databases running in a cluster. You can configure SCAN during the installation of Oracle Grid Infrastructure. Once configured, application tier connection descriptors specify the SCAN name instead of all the virtual hosts in the cluster.

You can specify the SCAN name as follows: `VISION = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=C-SCAN)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=VISION)))`

For more information, see the Oracle documentation.

For a thin driver, the sample SCAN connection string is as follows:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=engrac-scan)
(PORT=1521))(CONNECT_DATA
=(SERVER = DEDICATED)(SERVICE_NAME = TIBQADB)))
```

For an OCI driver, the sample SCAN connection string is as follows:

```
jdbc:oracle:oci:@TIBQADB
```

The sample `TNSNames.ora` file for the above URL is as follows:

```
TIBQADB =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = engrac-scan)(PORT = 1521))
)
(CONNECT_DATA =
(SERVICE_NAME = TIBQADB)
)
)
```

Note: See JDBC Connection Reference for details of the database URL of a shared JDBC connection.

Other Information about Oracle Database and Drivers

A few helpful points about Oracle database and drivers.

- Use `ojdbc7.jar` drivers file. You can download this file from Oracle.com.
- Maximum length for an Oracle table name or column name is 30 characters.

- When OCI client is configured with 32-bit client, Oracle might display "No data found" error. To resolve the error, reduce the recovery batch size using the `be.engine.cluster.recovery.batchsize` property.

  For example, if you are getting "No data found" error when the batch size is 10000, a batch size of 5000 might resolve the error, that is,

  ```
  <property name="be.engine.cluster.recovery.batchsize"
  value="5000"/>
  ```

Cases That Need Additional Setup

The following sections identify some cases where additional setup might be required.

Ontology Identifiers that Exceed the DBMS Maximum Column Length

Entity names and entity property names are used by backing store scripts to generate database table and column identifiers.

DBMSs put different limits on the length of a database identifier name. For example, in Oracle the maximum length is 30 characters, and in SQL Server the limit is 128 characters. Such restrictions might be applicable for ActiveSpaces 4.x, Cassandra, and custom backing stores too.

Generated database identifiers are longer than the TIBCO BusinessEvents identifiers because they contain characters in addition to the TIBCO BusinessEvents identifier. You can handle long identifier issues in either of the following ways (or a combination of these ways).

Letting the Utility Generate Short Aliases for Table Names

You can allow the `be-storedeploy` utility to generate short aliases for long names. You can also edit those names and rerun the utility. For details see Configuring Aliases File and Project Schema Script.

Note that alias file entries are also generated for another reason. See Ontology Identifiers that Use Database Key Words.

Specifying Short Table Names in the CDD File

You can avoid the problem of long entity type names before you begin to configure the backing store by specifying short database identifiers using the CDD Table Name setting.

The advantage of this method is that you can choose meaningful names before running the `be-storedeploy` utility. The disadvantages are that you may not know ahead of time which entities require short names, and you must also ensure that the table names you specify are unique across all entities in the ontology.

If you do not specify table names, and entity names are repeated. On the other hand, the generated table names are appended with dollar (`$`) characters as necessary, for example, `D_ORDER`, `D_ORDER$`, `D_ORDER$$` and so on.

See Configuring CDD for Special Cases (As Needed) for details.

Ontology Identifiers that Use Database Key Words

As well as database names that are too long, ontology terms that are key (reserved) words in your DBMS product must also be mapped to an alias. If errors occur when you run the SQL scripts due to key word clashes, examine the errors and add the appropriate words to the keyword mapping file.

A provided file (*BE_HOME*`/bin/dbkeywordmap.xml`) includes some basic mappings: `start`, `end`, `schema`, `mode`, and `index`. You can use it as a model.

Unlike the Aliases file, the keyword mapping file is not a project-specific file. It is intended to be generally useful across different projects. However, keyword mappings are also added to the aliases file when you run the SQL scripts, so you can also provide project-specific aliases for the generic mappings, if you want to.

The procedures are explained in the task sections within the section Initializing the Database and Generate Non-Project Tables.

String Properties that Exceed the DBMS Maximum Column Length

The default column size for String type attributes is 255 characters. If you expect the data length of an entity property to exceed that value, then in the CDD file set the Max Length field for each entity's properties. The utility changes the data type of string attributes with long lengths to CLOB, as appropriate.

See Configuring CDD for Special Cases (As Needed) for details.

Excluding Entities from the Backing Store

You do not have to use the backing store for all entities. In the CDD file, you can specify entities for which you do not want to use the backing store by using the Has Backing Store domain object override setting.

Note: If later you want to include any excluded entities, you must change the setting and update the backing store setup as explained in Updating Existing Backing Store Schema.

See Configuring CDD for Special Cases (As Needed) for details.

Configuring CDD for Special Cases (As Needed)

This section summarizes CDD configuration that may be required to handle special cases.

See Cases That Need Additional Setup.

The CDD file is only required during backing store setup if configured for these special cases. Other aspects of CDD configuration are ignored. Do this aspect of CDD configuration before you use the `be-storedeploy` utility.

For more details on CDD file configurations, see *TIBCO BusinessEvents Configuration Guide*.

Procedure

1. In TIBCO BusinessEvents Studio, open the project's CDD file and select Persistence Options from the list on the left.
2. In the tree on the left, click **Overrides** and click **Add**.
3. Select the entity or entities you want to customize and click **OK**.

4. In the tree on the left, select the **/uri** entry for each selected entity in turn, and configure the settings on the right:
   - To exclude an entity from the backing store, uncheck the Has Backing Store checkbox.

     Note: In Memory Only mode, if you configure any entity override Mode setting as Memory Only, then backing store is disabled for that entity.

   - To specify a short table name for entities whose table names would exceed the database product's maximum length, check the Has Backing Store check box, and enter the name in the Table Name field. See Ontology Identifiers that Exceed the DBMS Maximum Column Length for details, and for an alternative way to handle long names.

     Tip: It is recommended that you specify table names that start with `"D_"` to match the standard naming convention.

   - 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), check the Has Backing Store checkbox, and in the Properties Metadata section, Max Length setting, specify the expected maximum length for all such properties. See String Properties that Exceed the DBMS Maximum Column Length for details.
5. Save the CDD.

Initializing the Database and Generate Non-Project Tables

Ensure that you have done all earlier tasks that may pertain to your case. See Cases That Need Additional Setup.

Run the Initialize Database Script as the DBA or System User

This script creates the TIBCO BusinessEvents user and initializes the database.

Warning: Running the script `initialize_database_YourDBMS.sql` script deletes the user before creating it again. Running the script `create_tables_YourDBMS.sql` drops all database tables before creating them again. This means you can run these scripts again during the test phases of your project development, without having to take extra cleanup steps.

The first time you run the `create_tables_YourDBMS.sql` script, you see harmless error or warning messages because there is nothing to delete.

If you are updating the schema for an existing backing store, see Updating Existing Backing Store Schema.

Procedure

1. As desired, change the default TIBCO BusinessEvents user credentials: Open the `initialize_database_YourDBMS.sql` script for editing and change the default username and password. The documentation uses the default username (`BE_USER`) and password (`BE_USER`)

2. Open a command window in the *BE_HOME*/bin directory (the default location of the scripts), and the appropriate command for your DBMS at the prompt:
   - For Oracle:

     ```
     sqlplus sys_user/sys_user_password@SID @initialize_database_oracle.sql
     ```

     Type `exit` to exit and commit.
   - For SQL Server:

     ```
     osql -S Your-Server -U sys_user -P sys_user_password -n -i initialize_
     database_sqlserver.sql
     ```

   - For MySQL:

     ```
     mysql -u username -p database_name < initialize_database_mysql.sql
     ```

   - For PostgreSQL:

     ```
     psql -U username -d database_name -f initialize_database_
     postgres.sql
     ```

   - For Cassandra:

     ```
     cqlsh -f $BE_HOME/bin/initialize_database_cassandra.cql
     ```

   This script creates the TIBCO BusinessEvents database user. This user must be used to run the other scripts. You see messages like the following:

   ```
   User dropped.User created.Grant succeeded.
   ```

   Tip: Using your database product, you can configure additional users to access the database, in addition to this user.

Run the Create Tables Scripts as the TIBCO BusinessEvents User

Log in as the TIBCO BusinessEvents user, `BE_USER` by default and run a script to create non-project specific tables.

Procedure

1. Open the command prompt and navigate to the *BE_HOME*/bin directory (the default location of the scripts).
2. On the command prompt, type the appropriate command for your DBMS:
   - For Oracle:

     ```
     sqlplus BE_USER/BE_USER@SID @create_tables_oracle.sql
     ```

     Type `exit` to exit and commit.
   - For SQL Server:

     ```
     osql -S Your-Server -d Your-DB -U BE_USER -P BE_USER -n -i @create_
     tables_sqlserver.sql
     ```

   - For MySQL:

     ```
     mysql -u BE_USER -p database_name < create_tables_mysql.sql
     ```

   - For PostgreSQL:

     ```
     psql -U username -d database_name -f create_tables_postgres.sql
     ```

   - For ActiveSpaces:

     ```
     tibdg -g <grid_name> -r <FTL_realm_URL> -s <BE_
     HOME>/bin/create_tables_as.tibdg
     ```

   - For Cassandra:

     ```
     cqlsh -k <keyspaceName> -u <username> -p <pwd> -f create_
     tables_cassandra.cql
     ```

   Use the credentials that are defined in the SQL files. By default those are: username `BE_USER`, with password `BE_USER`.

Result

You see various harmless error messages the first time that you run the script.

Note: TIBCO BusinessEvents Studio provides a wizard to configure the required properties and generate the project-schema-specific SQL scripts.

Generating Deployment Scripts for a Store

You can generate store deployment scripts for your projects that use a persistence store. For projects with shared all or store persistence configurations, you can generate migration scripts to migrate an existing database data from a legacy ID format to a new ID format.

See TIBCO BusinessEvents Migration Guide, for details about the data migration process.

To generate project-schema-specific and migration SQL scripts using TIBCO BusinessEvents Studio or the CLI, use one of the following procedures.

Before you begin

(*For ActiveSpaces store only*) For generating deployment scripts for the ActiveSpaces store, configure the *tibco.env.ACTIVESPACES_HOME* variable in the `be-storedeploy.tra` file.

Note:

- Ensure that you copied your JDBC drivers file to *BE_HOME*`/lib/ext/tpcl` (or other location in your class path).
- Ensure that the database server is started.

Generating Deployment Scripts Using the TIBCO BusinessEvents Studio

Procedure

1. In the TIBCO BusinessEvents Studio, right-click the project name and select **Studio Explorer > Export**.
2. In the Export window, select **TIBCO BusinessEvents > Backingstore Deployment** and click **Next**.

3. In the Deploy wizard, from the **Backingstore Type** dropdown, select the required store option and click **Next**.

   Store Types

Store Type	Description
Relational Database (JDBC)	Select this option if your project has a JDBC backing store setup, such as MySQL, Oracle, SQL Server, and DB2.
ActiveSpaces	Select this option if your project has ActiveSpaces as a persistence store.
Apache Cassandra	Select this option if your project has Apache Cassandra as a persistence store.

4. Provide the values for the displayed fields. See Project Schema Specific SQL Script Fields for the list of fields.
5. You can click either of these buttons:
   - **Finish** - To exit the wizard and generate the project-schema specific SQL scripts.
   - **Next** - To go to migration scripts generation page (*Step 6*).
6. On the Generate Migration SQL Scripts page, select the Generate Migration Scripts checkbox to enable the rest of the fields.

   Provide the values for the store connection fields. See Migration SQL Scripts Fields for a list of fields.

   To generate migration scripts for migrating existing data in a project from legacy ID format to new ID format, connect to an existing legacy ID database in the project by using store connection fields.
7. *(Optional)* Click **Test Connection** to test if connection to the store is successful.
8. Click **Finish** to generate the migration SQL scripts.

Generating Deployment Scripts Using the CLI

You can use the `be-storedeploy` utility (available at `BE_HOME/bin/`) to generate project-schema and migration SQL scripts from the CLI.

Before you begin

If you are using the relational database, then before you run `be-storedeploy`, open the file *BE_HOME*`/bin/be-storedeploy.tra` for editing. Specify the JDBC database type by using the `java.property.jdbcdeploy.database.type` property:

```
java.property.jdbcdeploy.database.type [oracle | db2 | sqlserver | mysql
| postgres]
```

For ActiveSpaces, you can set the rows expiration scan period (`be.as.store.expiration.scan.period`) and default TTL value (`be.as.store.default.ttl`) in `be-storedeploy.tra`. For more details on the expiration scan period and default TTL value in ActiveSpaces, see TIBCO ActiveSpaces documentation.

Note: The expiration scan period value and default TTL values in the `be-storedeploy.tra` file is applicable for the generated schemas only. For an object table, update the value in the `create_tables_as.tibdg`.

Procedure

1. Open the CLI and navigate to the `bin` folder in *BE_HOME*.

2. Use the following command to generate backing store deployment scripts for the project:

   ```
   be-storedeploy [-p <property file>] [-c <cdd file>] [-o <Schema
   output file>] [-op <schemagen | migration>] [-s <JDBC |
   ACTIVESPACES | CASSANDRA>] [-a <true|false>] [-u <PU Name>] [-h]
   [EAR file path]
   ```

   See Store Deployment CLI Utility Options for details on the parameters.

Result

The scripts are generated in the location that you specified. Information about script generation is also printed to the Console tab. See Generated SQL Scripts for a list of the generated scripts.

Deployment SQL Scripts Wizard Fields

The following tables lists the settings present in the wizard for generating project-schema and migration scripts.

For details about the procedure, see Generating Deployment Scripts for a Store.

Project Schema Specific SQL Script Fields

[Project Schema Specific SQL Scripts Options]

Field	Description
Common Fields	
Cluster Deployment Descriptor	Specifies the name of the cluster configuration file.
Output Directory	Specifies the schema output file directory.
Output File Name	Specifies the schema output file name.
Relational Database (JDBC) Backing Store Fields	
Database Type	Select the type of relational database setup in your project. The values are: • `oracle` • `db2` • `mysql` • `sqlserver` • `postgresql` The default value is `oracle`

[Project Schema Specific SQL Scripts Options](Continued)

Field	Description
	Note: The supported databases to migrate existing database data in a project from legacy ID format to new ID format are Oracle, MS SQL, MySQL and PostgreSQL.
Generate ANSI Scripts	Select this checkbox to use ANSI compatible SQL types during script generation. Default setting is checked.
Generate Optimized Scripts	Select this checkbox to eliminate script generation for in-memory events.
Use Extended Max String Size	This is active only for the `oracle` database type. Sets the maximum string size value to extended, that is to 32767 bytes.
Cluster Deployment Descriptor	Browse to and select the CDD file you want to use for the project.
Output Directory	Browse to and select a directory where the scripts are to be generated, for example, *BE_HOME*`/bin` (This directory is used if you generate files manually.)
Output Script Name Prefix	Enter a prefix for the output script filenames. For example, if you enter `acme` as prefix, the following scripts are generated: • `acme.sql` • `acme.aliases` • `acme_alter.sql` • `acme_cleanup.sql` • `acme_delete.sql` • `acme_remove.sql`

Migration SQL Scripts Fields

[Relational Database (JDBC) Migrations Scripts Fields]

Field	Description
Connection Configuration	Browse to and select the JDBC Connection shared resource used to connect to the existing backing store.
Database URL	Enter the database URL that points to the existing backing store.
Database Username	Enter the database username that was used when setting up the existing backing store.
Database Password	Enter the database password that was used when setting up the existing backing store.
Database Schema Owner	Enter the database schema owner for the existing backing store.

[ActiveSpaces Migrations Scripts Fields]

Field	Description
ActiveSpaces Connection	Browse to and select the ActiveSpaces Connection shared resource used to connect to the existing backing store. All other fields in the page are automatically populated based on the ActiveSpaces Connection selected.
Realm Server	Enter the URL of the TIBCO FTL Realm server.
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.
Username	Enter the database username that was used when setting up the existing persistence store.
Password	Enter the database password that was used when setting up the existing persistence store.
Use SSL	Select the checkbox to enable secure authentication.

[ActiveSpaces Migrations Scripts Fields](Continued)

Field	Description
	When you select the checkbox, the **Trust All** and Identity fields are enabled.
Trust All	Select the checkbox if the **Trust Type** in the store configuration is set to `Trust Everyone`. Clear the checkbox if the trust type is `Trust File`. When you select the checkbox, the **Identity** field is disabled.
Identity	If the Trust All checkbox is clear, provide path of the identity file.

[Cassandra Migrations Scripts Fields]

Field	Description
Database Node (Hostname)	Enter the hostname for the Cassandra server contact point.
Database Port	Enter the port number for the Cassandra server contact point.
Database Username	Enter the database username that was used when setting up the existing persistence store.
Database Password	Enter the database password that was used when setting up the existing persistence store.
Database KeySpace	Name of the keyspace to connect with on the Cassandra server. If not specified, the cluster name is used for the keyspace name.
Use SSL	Select the checkbox to enable secure authentication. When you select the checkbox, the following fields are enabled: • **TrustStore Location** • **TrustStore Password** • **Requires Client Authentication** **Note:** Generate JKS certificates for Cassandra SSL scenarios using Java 8 or Java 11.

[Cassandra Migrations Scripts Fields](Continued)

Field	Description
TrustStore Location	While using SSL authentication, provide the path to the base folder containing the certificate files.
TrustStore Password	Password for the trust store while using SSL authentication.
Requires Client Authentication	Select this checkbox if you enable two-way SSL authentication. When you select the checkbox, the **ID File** field is enabled:
ID File	Path to identity file while using two-way SSL.

Store Deployment CLI Utility Options

Run the script `be-storedeploy.exe` to build the schema in the database with the provided options.

Option	Description

`-h`
 	
`/h`
 	
`-help`
 	
`/help`
	Optional Displays this help.

`--propFile`
 	
`/--propFile`
 	
`-system:propFile`
	Optional When you execute `be-storedeploy`, it searches for a property file of the same name in the working directory. This property file provides startup values and other parameters to the executable. You can specify the path and filename of a startup property file explicitly using the `--propFile` parameter.
	
	For example, if you start the engine from a directory other than *BE_HOME*`/bin`, then you would generally use `--propFile` to specify *BE_HOME*`/bin/be-storedeploy.tra`.

Option	Description

`-p`
`/p`
`-property`
`/property`
	Optional Location of the custom property file for the project. Values in custom property file override the values in the startup property file.

`-o`
`/o`
`-out`
`/out`
	Specify the schema output path and file name.
	
	If you do not specify the path, the output file is created in the directory from where the script is being executed.
	
	If you specify a directory path, the backing store scripts are generated in the specified directory and the last element of the path is taken as the schema output filename.

`-c`
`/c`
`-cdd`
`/cdd`
	Specify the absolute path of the CDD file that contains the configuration details of backing store.
	
	For example: `C:\workspace\FraudDetection\fdstore.cdd`

`-s`
`-store`
`/s`
`/store`
	Optional Specify the store type for schema generation.
	
	The values are:
	• `JDBC`
	• `ACTIVESPACES`
	• `CASSANDRA`
	
	The default type is `JDBC`.

`-op`
`-operation`
`/op`
`/operation`
	Optional Specify the operation type for deployment.
	
	The values are:

Option	Description
	• `schemagen` - Select this, if you are creating a new backing store. This operation generates schema files. When database details are provided in the TRA file, it also generates ID migration script [<name>_migration.sql]. • `migration` - Select this, if you are migrating an existing backing store. **Note:** To generate migration scripts for migrating data in a project from legacy ID format to new ID format, connect to an existing legacy ID database in the project. The default value is `schemagen`.

`-u`
`-unit`
`/u`
`/unit`
	*(Only for the migration operation)*Specify the processing unit name for the migration.

`-n`	*(Only for the migration operation)*Specify the TIBCO BusinessEvents engine name for the migration.
Relational Database (JDBC) Backing Store Fields	

`-a`
`/a`
`-ansi`
`/ansi`
	Optional If set to true, ANSI compatible SQL types are used during script generation.
	Allowable values are true and false. For ANSI compatible databases, it is set to true by default.

`-optimize`	*Optional* Use this option to eliminate schema generation for in-memory events.
`-maxstringsize`	Sets the maximum string size value to extended, that is to 32767 bytes.

Option	Description
`-histidnotnull`	Disallow null values in history arrays.
`-childidnotnull`	Disallow null values in child arrays.

Migrating Data from Old TIBCO BusinessEvents Projects to Version 6.x

You can use the migration utility `be-storedeploy` to migrate data from old projects to the version 6.x. You can perform the following types of data migration:

Migrating Data from TIBCO ActiveSpaces 2.x based Clusters

To migrate data from TIBCO ActiveSpaces 2.x (Legacy ActiveSpaces) clusters to newly supported store and cluster providers in TIBCO BusinessEvents, you can use the migration utility. TIBCO BusinessEvents still supports the old store and cluster providers and you do not need to migrate data to continue using them. The minimum supported versions for data migration to new stores are ActiveSpaces version 2.3 and above, and TIBCO BusinessEvents version 5.5.0 and above.

The following table shows the supported data migration paths:

TIBCO BusinessEvents version 5.x configuration	TIBCO BusinessEvents version 6.x configuration
Legacy ActiveSpaces - Shared Nothing	Apache Ignite - Shared Nothing
Legacy ActiveSpaces - Shared Nothing	TIBCO ActiveSpaces store
Legacy ActiveSpaces - Shared Nothing	Apache Cassandra store

Migrating Primary Keys

When you migrate a project, the data is migrated to the new ID implementation by default.

The new ID implementation sets the existing extID as the primary key in the migrated data. You can migrate the project with extId with Legacy ActiveSpaces - Shared Nothing persistence configuration to the new ID while still keeping Legacy ActiveSpaces - Shared Nothing configuration.

You can optionally continue to use Legacy Id settings while migrating a project.

For details, see "Enabling the Legacy Lookup Strategy" in *TIBCO BusinessEvents Administration*.

Before you begin

Download and install TIBCO ActiveSpaces, Apache Cassandra, and TIBCO FTL as required by the project configuration. For details about the installation and configurations, see their respective documentation.

Procedure

1. Set the following environment variables in the `be-storedeploy.tra` file located at *`BE_HOME`*`/bin` as applicable:
    - `ACTIVESPACES_HOME` based on the new project configuration
    - `FTL_HOME` based on your project requirement
2. Import the old project to the TIBCO BusinessEvents Studio. For more information, see Importing Projects in TIBCO BusinessEvents Studio.
3. Configure the backing store or the Shared Nothing persistence as required. For more information, refer one of the topics below:

Persistence Option	**Reference**
TIBCO ActiveSpaces as backing store	Configuring ActiveSpaces as a Backing Store
Apache Cassandra as backing store	Configuring Apache Cassandra as a Store Provider
Apache Ignite for Shared Nothing persistence	Configuring Apache Ignite as a Cache Provider

4. Build an EAR file for the project. For more information, see Build an EAR File.

5. Start the cache agents for your old TIBCO BusinessEvents project. For details on starting the agents, see the *TIBCO BusinessEvents Administration* guide.
6. Run the `be-storedeploy` data migration utility.

   For a detailed process, see the Generating Deployment Scripts for a Store topic in *TIBCO BusinessEvents Developer Guide*.

   For projects with no cache no cluster configurations, run the migration utility with default processing unit. For details about the `be-storedeploy` utility options, see Store Deployment CLI Utility Options.

Aliases File and Project Schema Script

For every entity, property, or state machine whose database identifier name exceeds the database maximum length. A table name entry is created in the generated *`yourname`*`.aliases` file (for example, `acme.aliases`).

See Ontology Identifiers that Exceed the DBMS Maximum Column Length for more information, and for an alternative way to specify short table names.

It is a good idea to check the aliases file for entries, even if the TIBCO BusinessEvents names are not very long. The length of the generated database table names is not easy to predict.

Optionally, you can edit the file to provide more meaningful names.

Note: It is recommended that you keep the aliases file for future reference. If the project ontology changes after the backing store has data in it, you must also update the database schema to match the new schema (as explained in Updating Existing Backing Store Schema). If you modified the generated aliases, you must use the same aliases again when you update the schema, to preserve those columns and their data.

Keyword mapping file

Add entries in the keyword mapping file to the aliases file so that you can replace the keyword aliases with project-specific ones, as desired. It is generally done in a second pass. See If Needed - Map Key (Reserved) Words to Aliases for details.

Configuring Aliases File and Project Schema Script

To configure aliases files and project schema scripts, following are the steps:

1. Check the Aliases File and Modify Aliases as Desired
2. Run the Project Schema Script (as BE_USER)
3. If Needed - Map Key (Reserved) Words to Aliases
4. Project Configuration (As Needed)

Check the Aliases File and Modify Aliases as Desired

Procedure

1. Open the `*yourname*.aliases` file for editing.
2. Replace any aliases as desired with more meaningful short names.

   Make sure that each name is unique. It is a good idea to leave any system generated prefixes or suffixes in place for consistency of names across the database.
3. Rerun the `be-storedeploy` tool, using the same parameters as before.

   For details see Generating Deployment Scripts for a Store . This time, the aliases you created are used.

Run the Project Schema Script (as BE_USER)

In this step, you log in as the user you created and run a script to create the project related part of the database schema.

Procedure

1. Open the command prompt and navigate to the *BE_HOME*/`bin` directory (the default location of the scripts).
2. Run the *yourname*`.sql` script. (For example, `@acme.sql`). You can log in with the username `BE_USER`, password `BE_USER` or whatever username and password you have set up).

3. On the command prompt, type the appropriate command for running project schema scripts for your backing store. For complete details on these commands, such as to enable SSL authentication or enable client-side and server-side encryption, see the documentation of your backing store.

   - For Oracle:

     ```
     sqlplus BE_USER/BE_USER @ yourname.sql
     ```

   - For SQL Server:

     ```
     osql -S Your-Server -d Your-DB -U BE_USER -P BE_USER -n -i
     @yourname.sql
     ```

   - For MySQL:

     ```
     mysql -u BE_USER -p database_name < yourname.sql
     ```

   - For PostgreSQL:

     ```
     psql -U username -d database_name -f yourname.sql
     ```

   - For ActiveSpaces:

     ```
     tibdg -g <grid_name> -r <FTL_realm_URL> -s <yourname>.tibdg
     ```

   - For Cassandra:

     ```
     cqlsh -k <keyspaceName> -u <username> -p <pwd> -f
     <yourname>.cql
     ```

Result

If there are no errors, then your database tables are now configured for use. But, if there are errors, then you may need to add some mappings to the key word mapping file.

If Needed - Map Key (Reserved) Words to Aliases

This task is optional and performed in case there are errors in the previous steps.

Complete this task only if you saw errors after completing Run the Project Schema Script (as BE_USER). Such errors are caused when your project ontology uses terms that are key words (reserved terms) in the DBMS you are using. You must map these terms to an alias in the keyword mapping file.

Procedure

1. Edit the keyword map file in either of the following ways:
    - Edit the *BE_HOME*/`dbkeywordmap.xml` file to add entries.
    - If you use a custom backing store, you can create your own `.xml` mapping file.
    - For Cassandra backing store, edit the *BE_HOME*/`bin/cassandrakeywordmap.xml` file at the location *BE_HOME*/`bin`.

    Below is the format for entries in the .xml mapping file:

    ```
    <keyword name="“dbKeyWord" mapname="“nonDbKeyWord"/>
       <keyword name="start" mapname="start_"/>
    ```

2. *(Optional)* If you create a custom `.xml` keyword file or use the `cassandrakeywordmap.xml` file, edit the `java.property.jdbcdeploy.bootstrap.keyword.file` path in the `be-engine.tra` and `be-storedeploy.tra` files as follows:

    ```
    java.property.jdbcdeploy.bootstrap.keyword.file  path_to_your_file/your_
    file.xml
    ```

3. Repeat Generating Deployment Scripts for a Store, and tasks following as needed.
4. If desired, create project-specific aliases for the key word mappings as explained in Providing Project-Specific Keyword Aliases.

Providing Project-Specific Keyword Aliases

When you repeat, the new key words are added to the *yourname*`.aliases` file. You can create project-specific aliases for the key word mappings as desired. Then repeat and continue.

Note: You must generate the SQL scripts a minimum of three times if you add keyword mappings to the aliases file-it might be more because you may not catch all errors at once. For example, if there are multiple keyword clashes in one table, only the first are reported. Perform this loop until no more errors occur.

Procedure

1. Generate the SQL scripts and run them (as explained in the procedures).

   Errors occur due to key word clashes.

2. Add the appropriate key word mapping entries to the key word mapping file.

3. Generate the SQL script again.

4. To use project-specific aliases for the keyword mappings (Optional):

   a. Edit the aliases file entries for the key word mappings.

   b. Generate the SQL script again.

5. Run the SQL script to create the backing store.

Project Configuration (As Needed)

Complete the TIBCO BusinessEvents Studio project configuration tasks if you have not already done so. These tasks can be done before or after database setup. See the following sections:

- For details on adding a JDBC Connection resource, see *TIBCO BusinessEvents Developer Guide*Adding a Shared Resource
- For details on backing store configurations in CDD, see *TIBCO BusinessEvents Configuration Guide*Adding a Shared Resource

  Note: Update your schema if your ontology changes, or if you want to include or exclude different entities in the backing store. See Updating Existing Backing Store Schema for details.

Updating Existing Backing Store Schema

If you change the project ontology, that is, if you create, alter or delete a concept or an event, you must update the backing store schema so it matches the updated ontology. In the case of changes in project ontology, you must update the backing store schema before you deploy the updated project.

You may also wish to change which entities are excluded from the backing store using CDD settings (see Excluding Entities from the Backing Store for more information. This change does not require project redeployment. It requires that the updated CDD file is copied to all runtime machines.

Examine the alter script before you run it. The section What the Schema Update Utility Can Handle Automatically provides more information.

To run the Schema Update Utility:

Procedure

1. Gracefully shut down the deployed application (all agents including cache agents).
2. Back up your existing database.
3. Generate the updated EAR file for the modified project.
4. If you modified aliases when you created the schema, locate the `*yourname*.aliases` file you used. It will help you to modify those aliases in the newly generated file, so they match.

5. Open the `be-storedeploy.tra` file for editing and set the following properties:

   Relational database (JDBC)

   ```
   be.jdbc.schemamigration.url DbURL
   be.jdbc.schemamigration.user username
   be.jdbc.schemamigration.pswd password
   ```

   TIBCO ActiveSpaces

   ```
   #be.jdbc.schemamigration.url jdbc:tibco:tibdg:_
   default;realmurl=http://localhost:8080

   #be.jdbc.schemamigration.url jdbc:tibco:tibdg:_
   default;realmurl=https://localhost:8080;username=<user-
   name>;userpassword=<password>

   #be.jdbc.schemamigration.trustfile
   /home/user/activespacesdata/realm_data/ftl-trust.pem
   ```

   Cassandra

   ```
   be.cas.schemamigration.url hostname
   be.cas.schemamigration.port port_number
   be.cas.schemamigration.user username
   be.cas.schemamigration.pswd BE_USER
   be.cas.schemamigration.useSsl true_or_false
   be.cas.schemamigration.keyspace keyspace_name
   ```

6. Use one of the following:

   a. Database URL that points to the existing peristence store. For details on adding a JDBC or ActiveSpaces Connection resource, see Adding a Shared Resource.

   b. Same username and password you used when setting up the persistence store. See Initializing the Database and Generate Non-Project Tables

   These properties enable the program to compare the schema of the existing database with the ontology in the project EAR file, and generate the alter script.

7. Log on as the user name you specified in Initializing the Database and Generate Non-Project Tables.

8. Run the `be-storedeploy` utility as explained in Generating Deployment Scripts for a Store .
9. If any of the new or changed definitions result in entries in the `*yourname*.aliases` file, and you want to change the provided aliases, follow instructions in Configuring Aliases File and Project Schema Script. If you modify aliases, remember to generate the scripts again so the modified aliases are used.

   Note: You must use the same aliases that you used before. If any were modified when the schema was created, you must modify them the same way when updating the schema. It can be useful to refer to the original aliases file.

10. Examine the generated ***yourname*_alter.sql** script and modify as needed so you only run statements for changes you want to make. See What the Schema Update Utility Can Handle Automatically for details.
11. Run the ***yourname*_alter.sql** script.

Result

Your database tables are now configured for use.

What the Schema Update Utility Can Handle Automatically

You must examine the altered script before you run it. Decide what changes to make manually and what changes to make using the script, considering the kind of data in the tables. Entries that could result in data loss are commented. Remove or comment entries for changes you do manually.

Restriction: You cannot change the type of the field for ActiveSpaces and Apache Cassandra.

Adds

The schema migration utility handles the addition of entity types and attributes. New entity types and attributes are added to the database schema.

Changes (Drop and Add)-Assess individually

The utility handles changes to attributes (entity properties) as `DROP` and `ADD` operations. However, `DROP` operations are commented in the script to avoid data loss.

If a column is empty, or you do not want to keep the data they contain, you can enable the `DROP` operation and let the utility handle the change.

If the column contains data that you want to keep, then make the change manually by using an appropriate database tool. For example, you can change the data type of a column from string to double without loss of data, as long as all the column values are numeric values.

Entity Deletions

If an entity is deleted from the TIBCO BusinessEvents Studio project, the corresponding tables are not dropped from the database schema. Existing data is not lost. Deleted entities are not mentioned in the alter script. Manually monitor and delete such tables as needed.

Attribute Deletions

The schema update utility does handle deletion of entity attributes. SQL statements for deleted attributes are generated but they are commented. Examine the altered script and enable these commands if you want to run them. Note that existing data is lost when you drop an attribute.

Example Alter Script

Below is an example `*yourname*_alter.sql` script.

Property type change

```
--  ##### WARNING :  Non-alterable Ontology changes found. Please see
following errors. Manual schema-migration is required.
--* For Concept Concept1 field PROPERTY_1 type changed from VARCHAR2
to LONG
-- ALTER TABLE D_Concept1 DROP ( Property_1 );
```

```
ALTER TABLE D_Concept1 ADD ( Property_1 numeric(19) );
New table
DROP TABLE D_Book_rrf;
CREATE TABLE D_Book_rrf (pid numeric(19), propName char varying(255),
id$ numeric(19) not null);
New property
-- ALTER TABLE D_MyConcept DROP ( FOLDER_1 );
ALTER TABLE D_MyConcept ADD ( Folder_0 char varying(255) );
```

Persistence Store Table Reference

The backing store uses relational tables and SQL data types for ease of maintenance. The SQL (DDL) scripts use ANSI SQL type definitions (where supported by the target DBMS product).

Each ontology type in the backing store has its own primary table and zero or more second-level tables. There are only two levels of tables, which makes the database easier to manage and easier to understand. Because the backing store adheres to SQL standards and a straight-forward structure, standard database tools can be used to view backing store data.

Primary Tables

Primary tables contain only primitive properties such as the following:

JDBC Store Property	Cassandra Property	ActiveSpaces Property	Note
`cacheId`	`cacheId`	`cacheId`	Entity version number (starts with 1)
`payload__p`	`payload__p`	`payload__p`	The Event payload content.
`time_acknowledged$`	`time_ack_`	`time_acknowledged$`	Time when the Event was acknowledged.
`time_sent$`	`time_sent$`	`time_sent$`	Time when the Event was sent.

JDBC Store Property	Cassandra Property	ActiveSpaces Property	Note
`time_created$`	`time_ created_`	`time_created$`	Time when the entity was created
`time_last_ modified$`	`updated_`	`time_last_ modified$`	Time when the entity was last modified
`parent$_id$`	`rrfs_parent`	`parentid$`	Id of the parent for contained concepts
`id$`	`id_`	`ID`	Unique Id of the entity (must be unique across all entities)
`extId$`	`extid_`	`extid`	Unique (or null) extId assigned
`state$`	`state_`	`state$`	Always set to ‘C’ meaning ‘Created’ (reserved for future use)

Secondary Tables (JDBC Only)

Secondary tables are used for complex properties, that is, arrays, properties with history, and concept relationship properties. Each array and history-enabled property has a separate table. Only primitive properties are stored in the primary table.

Note: Secondary tables are not used for ActiveSpaces and Apache Cassandra stores.

Secondary table structure

Property Type	Column	Description
Array	`pid$`	Parent ID
	`valPid$`	Array index
	`val`	Item’s value

Property Type	Column	Description
History	`pid$`	Parent ID
	`howMany`	Number of history items
	`timeIdx`	Item’s time stamp
	`val`	Item’s value
Array with History	`pid&`	Parent ID
	`valPid$`	Array index
	`howMany`	Number of history items
	`timeIdx`	Item’s time stamp
	`val`	Item’s value

Reverse Reference Tables

Each concept also has a reverse reference table. This table’s name contains the concept name and ends with the characters `_rrf$`. It has these columns:

Column	Description
`pid$`	Parent ID from the main concept table
`propertyName$`	Property name (field) from the referencing concept.
`id$`	Identifier (`id$`) of the referencing concept.

Class-to-Table Mapping

This table contains the mapping between class names and table names, and the mapping between complex property field names and secondary table names. For example:

```
'be.gen.Ontology.DeleteVerifyEvent', 'D_DeleteVerifyEvent‘
'be.gen.Ontology.Treatment', 'D_Treatment‘
‘be.gen.Ontology.Treatment', 'rrf$', 'D_Treatment_rrf$‘
'be.gen.Ontology.BaseAlert', 'treatments', 'D_BaseAlert_treatments'
```

Domain Models

Domain model resources enable you to control user input for decision tables and test data. You can add and import domain models and associate them with entity properties by using TIBCO BusinessEvents.

You can add domain models for concept, event, and scorecard properties. A domain model specifies the values that you may find useful for defining ontology item properties. For example, instead of typing text for a certain concept property, you can pick a value from a list, or enter a value within a predefined range.

A domain model can extend another domain model. When defining a domain model you can specify which domain model it inherits from.

Domain model entries are case sensitive. For example, m and M are recognized as different entries.

Domain models can be used in decision tables and in test data for Tester. For details about working with decision tables, see *TIBCO BusinessEvents Decision Manager User Guide*.

Note: You cannot add domain models for concept properties of the `ContainedConcept` or `ConceptReference` type.

Setting Up a Domain Model

Procedure

1. Add domain models. For example, create a folder and add all the models you need for a project in this folder.

   For details, see Add a Domain Model.

   You can also import and export domain models.

   For details, see Importing and Exporting Domain Models.

2. Associate domain models with properties.

   For details, see Associating Domain Models to Properties.

Result

When creating a decision table, domain models control the values you can use for a given property.

Domain Model Value Descriptions for User-Friendly Presentation

All domain model values can have optional descriptions that appear in the domain model editor. A preference determines whether domain model values or their descriptions appear in decision table cells. For some applications, displaying descriptions can make the table easier for users to understand. For example, suppose the value is a code such as 23, and the description is North West. Users find it is easier to work with the description than the code. As another example, for a Boolean data type, the description can provide words such as Accepted and Rejected for the values True or False.

Adding a Domain Model

You can store domain models as desired. For example, in a folder called `DomainModels`. For each domain model, you create a set of domain entries. Each entry represents a valid value for the entity property that uses the domain model.

After you add a domain model, associate it with a property. See Associating Domain Models with Properties

Add a Domain Model

You can store domain models as desired, for example, in a folder called `DomainModels`. For each domain model, you create a set of domain entries, where each entry represents a valid value for the entity property that uses the domain model.

Procedure

1. Right-click the folder where you want to store the domain model, and select **New > Domain Model.** You see the New Domain Model Wizard.

   Alternatively, right-click a resource, and select **New > Other**. In the New dialog, select **Domain Model** under TIBCO BusinessEvents.

2. In the **Domain Model Name** field, type a name for the domain model. In the **Description** field, type a description.

   Note: You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename**. See Element Refactoring Operations for more details.

3. Click **Finish**.

   The new Domain Model is opened in the Domain Model editor.

4. In the **Domain Type** field, select the data type for this domain model. See Supported Data Types for supported types.

5. As needed, complete the **Inherits From** field. If this domain model extends another domain model, browse to and select that domain model.

6. In the Domain Entries section, click **Add.**

   An empty row appears in the table of entries.

   You can also select rows and click **Duplicate** to duplicate (and then modify) selected rows.

   You can select a row and click **Remove** to remove individual rows that are not needed.

7. For each row you add, enter a description (optional) and in the Details section, define the domain model entry.

   The Details section presents appropriate fields for defining the type of domain model entry you selected in *Step 4*.

8. When you have created the entries for the domain model, save the domain model resource.

What to do next

After you add a domain model, associate it with a property, see Associating Domain Models to Properties.

Adding Domain Entries

When you add a domain model, first select its data type. The Domain Model editor then displays an appropriate user interface for defining domain model entries of that data type.

The Domain Entries section is the same for all types. The Details section changes depending on what data type you select for the domain model.

Supported Data Types

Domain Models support the following data types:

- String
- Integer
- Long
- Double
- Boolean
- DateTime

Sections below show the user interface for each type.

String

String entries are simple text strings.

Note: Numeric values in a String domain type

When you use a numeric value (Integer, Double, or Long) in a domain model of type String, TIBCO BusinessEvents adds double quotes around the value. (These are visible after you save and reopen the domain model.)

Integer, Double, Long

In a single domain model, you can enter single values, range values, or a mixture of both. Acceptable values for integer, long, and double domain entries are the same as for the equivalent Java data types.

Note: Text values in a numeric data type

When you enter text in a domain model of type Integer, Double or Long, TIBCO BusinessEvents marks it in red color, and indicates you to correct it.

The user interface is similar for all numeric data types.

You define whether each end of the range is inclusive. For complete coverage, ensure that there is no gap and no overlap between ranges through consistent use of the included checkbox.

Boolean

Boolean entry values are always `true` or `false`. The description can give the meaning of the pair of choices, such as male or female, supported or unsupported, eligible or ineligible and so on.

DateTime

In a single domain model, you can enter single values, range values, or a mixture of both. Using a date and time picker, you can specify a date and time.

If you do not want to specify a time of day, set the time to midnight (12:00 AM) for the start date, and to a minute before midnight (11:59:59 PM) for the end date of an inclusive range, or for a single date.

The calendar shows a 12-hour clock. PM numbers are converted to a 24-hour clock format in the value table.

Importing and Exporting Domain Models

You can import domain model information from a database, from a Microsoft Excel spreadsheet, and from the source of a database concept property.

Importing Domain Model from a Database Table

When you import domain model information from a database, the result set from the SQL query is transformed for use as domain model entries. For details, see Importing Domain Model Entries from a Database Table.

Importing Domain Model from a Database Concept

You can create a domain model by importing values from the database column that corresponds to a database concept property. For details, see *TIBCO BusinessEvents Data Modeling Developer's Guide*.

Exporting To and Importing Domain Model From Excel

You can export a domain model to a Microsoft Excel spreadsheet. The description for each domain model entry goes in column A, and the value goes in column B. When you import from an Excel spreadsheet, similarly, column A is used as the description, and column B is used for the value.

For details on exporting and importing from a Microsoft Excel, see the following topics:

- Export Domain Values to Excel
- Importing Domain Model Entries from Excel

After you import a domain model, associate it with a property. See Associating Domain Models to Properties

Exporting To and Importing From Excel

You can export a domain model to an Excel spreadsheet. The description for each domain model entry goes in column A, and the value goes in column B. When you import from an Excel spreadsheet, similarly, column A is used as the description, and column B is used for the value.

In TIBCO BusinessEvents Decision Manager, you can export decision tables to Excel and import them from Excel. When you do so, you also export and import domain models associated with properties used in a decision table column. See *TIBCO BusinessEvents Decision Manager User Guide* for details.

Importing Domain Model Entries from Excel

Procedure

1. In TIBCO BusinessEvents Studio Explorer, perform one of the following actions.
   - Right-click the folder where you want to create the domain model and select **Import > TIBCO BusinessEvents > Domain Model.**
   - In TIBCO BusinessEvents Studio Explorer, select any item in the project entity and select **File > Import > TIBCO BusinessEvents > Domain Model.**
2. Click **Next**. You see the Import Domain Model Wizard.

   If you invoked the import wizard by right-clicking a folder, that folder is selected as the parent folder. You can choose a different one as desired.
3. In the **Domain Import Source** field, select **EXCEL**.
4. In the **File Name** field, enter a name for the domain model resource. Optionally, enter a description.
5. In the **Data Type** field, select the appropriate data type for the domain model and click **Next**. You see the next wizard page.
6. In the **Select Excel File to Import** field, browse and select the Excel file that has the domain model information.
7. Click **Finish**.

   The `Domain Import Successful` message is displayed.
8. Click **OK**.

   You see the Domain Model editor. The column values appear as domain entries. You can add, edit, duplicate, and remove entries as appropriate.

Export Domain Values to Excel

Procedure

1. In TIBCO BusinessEvents Studio Explorer, perform one of the following actions.
   - Right-click the domain you want to export and select **Export > TIBCO BusinessEvents > Export Domain to Excel**.
   - In TIBCO BusinessEvents Studio Explorer, select any item in the project entity and select **File > Export > TIBCO BusinessEvents > Export Domain to Excel**.
2. Click **Next**. You see the Export to Microsoft Excel File Wizard.
3. In the **Select Excel File to Export to** field, browse and select the Excel file to which you want to export the domain model information.

   To create a new Excel file, specify a filename that does not yet exist. If you specify an existing Excel file, the file contents are replaced with the exported domain model information.
4. In the **Select Domain to Export** area, expand the display and select the domain model you want to export.

   If you invoked the export wizard by right-clicking a domain model, that domain model is selected. You can choose a different one as desired.
5. Click **Finish**.

   The `Export Domain Model to File Successful` message is displayed.
6. Click **OK**.

Importing Domain Model Entries from a Database Table

When you import domain model information from a database, the result set from the SQL query is transformed for use as domain model entries. This feature is supported with Oracle and MySQL Database.

If imported from a database concept, database table, or a database query, you can update the domain model by using the **Reload Domain Model** button available on the top bar of the Domain Model editor.

Procedure

1. Add a JDBC Connection resource and configure it to connect to the database from which you want to import the domain model.

   For details, see JDBC Connection Reference for details.

2. In TIBCO BusinessEvents Studio Explorer, perform one of the following actions.

   - Right-click the folder where you want to create the domain model and select **Import > TIBCO BusinessEvents > Domain Model.**
   - In TIBCO BusinessEvents Studio Explorer, select any item in the project entity and select **File > Import > TIBCO BusinessEvents > Domain Model.**

3. Click **Next**.

   The Import Domain Model Wizard opens.

   If you invoked the import wizard by right-clicking a folder, that folder is selected as the parent folder. You can choose a different one as desired.

4. In the **Domain Import Source** field, select **DATABASE_TABLE**.

5. In the **File Name** field, enter a name for the domain model resource. Optionally enter a description.

6. In the **Data Type** field, select the appropriate data type for the domain model and click **Next**. You see the next wizard page.

7. In the **JDBC Resource URI** field, browse and select the JDBC Connection resource that connects to the database you want to use.

   The connection information from the JDBC Connection resource displays. You can override it here as desired.

   Click **Next**. You see a list of tables in the database. Expand the list of tables to see the columns that match the datatype specified in *Step 6*.

8. Perform one of the following actions.

   - Select one or more columns, then click the **Create Domain for Selected Columns** button. Values of all columns are used for the domain model entries.
   - Click **Advanced** and enter an SQL query whose result set is used to create the domain entries. This option enables you to make use of joins, where clauses, and so on. Then click **Execute Query**.

9. Click **Finish**.

   The `Domain Import Successful` message is displayed.

10. Click **OK**.

    You see the Domain Model editor. The column values appear as domain entries. You can add, edit, duplicate, and remove entries as appropriate.

Associating Domain Models with Properties

You can associate domain models with concept, event, and scorecard properties.

If domain models have the same data type as that of the properties, you can associate multiple domain models with multiple properties.

For example, an event and a concept have an `OrderID` property defined the same way, and both the `OrderID` properties use an `OrderIDModel` domain model. In this case, the `OrderIDModel` domain model is associated with the `OrderID` property of the event, as well as of the concept. However, the concept or the event property is associated only with the `OrderIDModel` domain model.

Associating Domain Models to Properties

You can associate domain models with concept, event, and scorecard properties.

If domain models have the same data type as that of the properties, you can associate multiple domain models with multiple properties.

For example, an event and a concept have an `OrderID` property defined the same way, and both the `OrderID` properties use an `OrderIDModel` domain model. In this case, the `OrderIDModel` domain model is associated with the `OrderID` property of the event, as well as of the concept. However, the concept or the event property is associated only with the `OrderIDModel` domain model.

Procedure

1. Perform one of the following actions.
   - In TIBCO BusinessEvents Studio Explorer, expand the concept, event, or scorecard to display its properties. Right-click the desired property and select **Associate Domains**.
   - In TIBCO BusinessEvents Studio Explorer, open the desired concept, event, or scorecard editor. In the **Domain Model** field of the desired property, click the browse icon.

     The Associate Domains dialog appears.
2. Expand the project tree to display domain models, and select one or more as appropriate.

   The displayed domain models have the same data type as that of the property.
3. Save the domain model resource.

Validating Data in Domain Models

You can validate duplicate entries and a mismatch of upper and lower range values while defining domain models.

Duplicate Domain Values Not Allowed

Each domain value must be unique. If you accidentally enter duplicate values, the Problems view displays helpful information.

Mismatching Range Values Not Allowed

If you accidentally enter mismatching values in the Lower and Upper fields for range values, the Problems view displays helpful information.

Display Models

Display Models provides a mechanism to define the display text for elements in the TIBCO BusinessEvents WebStudio classic user interface, such as, properties. Instead of displaying the default element name, a more user readable name in any selected language could be displayed.

You can provide a name for the artifact and its properties in your preferred language using Display Model. Thus, you can see the artifact and its properties name in your language while using the WebStudio. See the *TIBCO BusinessEvents WebStudio User Guide* for how it would appear in the WebStudio.

Display Models are defined inside the TIBCO BusinessEvents Studio, similar to Concepts, Events, and Rules. A Display Model is a specialized type of property file, and follows the familiar name/value pair syntax for property files in Java. Display Models are the first class citizens inside the BusinessEvents Studio project, and their file extension is `.DISPLAY`.

Naming

TIBCO BusinessEvents assigns the Display Model name that matches with the associated BusinessEvents project artifact (Concept or Event). For instance, for the Concept named `Person.concept`, the corresponding Display Model artifact is named `Person.display`. This naming method simplifies the syntax and allows for a loose coupling of the artifacts. However, the Display Model follows the standard Java I18N naming conventions if language and country values are specified, that is the name of the file is `<`*`artifact_name`*`>_<`*`language_code`*`>_<`*`country_code`*`>.display`. For example, if English is selected as the language and the United States is selected as the country, for the Concept named `Person.display`, the Display Model file is saved with the file name `Person_en_US.display`.

The Display Model also supports code refactoring. If an artifact is renamed, then the Display Model file associated with the artifact is automatically renamed. If the properties of the artifact is renamed, then the properties are also updated in the Display Model.

Syntax

The syntax for the Display Model files follows the standard Java property file syntax, that is, specifying name and value pairs. There are three major parts of the file:

- For each artifact property, there is an optional text that is displayed in the WebStudio UI. The simple syntax is as follows:

  ```
  <property_name>=<display_text>
  ```

  For example,

  ```
  Name.displayText=User Name
  ```

  Note: In the case of Arabic characters, do not use accents, as they are not supported in TIBCO BusinessEvents.

- Additionally, there is an optional property that allows properties to be hidden entirely from the WebStudio UI, so that they are not accessible from the builder-based Business Rule editor or Decision Table editor.

  ```
  <property_name>.hidden=true | false
  ```

  For example,

  ```
  Name.hidden=true
  ```

- Finally, there is an optional property to specify the text to be displayed wherever the artifact itself is shown in the WebStudio UI, for instance inside of the Group Contents window. The syntax for this setting is:

  ```
  displayText=<display_text>
  ```

  For example, for an `Applicant` concept

  ```
  displayText=Credit Card Applicant
  ```

Overriding Behaviors

As Concepts and Events can inherit properties from the parent Concepts and Events, any visible property from the artifact's hierarchy is also visible to the Display Model. For example, if the Concept `Employee` inherits a property called `Name` from the Concept `Person`, it is valid to specify the display text for the `Name` property inside of the Display Model for the `Employee` Concept. The `Name` property can have different meanings for different concepts, and you can use display text to differentiate them.

You can also override a display text inside a Rule Template. This overriding behavior provides another level of meaningful, context-aware specification of properties. A new `display` section could be added to the Rule Template source, and the context-sensitive display text for each accessible property can be specified using the dot notation. For example,

```
display {
             account.AccountID = "Personal Checking Account ID";
       }
```

Creating a Display Model

A Display Model provides the more readable name to the artifact properties and you can create it using the TIBCO BusinessEvents Studio.

Procedure

1. Right-click on any folder in the project and select **New > Other**. In the New dialog select **TIBCO BusinessEvents > Display Model**.

   The New Display Model Wizard window is displayed.

2. Enter the values of the following fields in the wizard and click **Finish**.

Field	Description
Project	Name of the project
Target Entity	Project path of the artifact (concept or event) for which the display text is created. Click **Browse** to browse and select the artifact of the project.
Language	(Optional) Language for the display model. If none is selected, the default language is English.
Country	(Optional) Country for the display model. If none is selected, the default country is United States.

The Display Model editor is displayed for adding the display text for the artifact and its arguments. See Editing a Display Model for more details.

Result

If no language/country is chosen, a file with the default name of `<artifact_name>.display` is created. If language and country values are selected, the name of the file follows the standard Java I18N naming conventions, that is the name of the file is `<artifact_name>_<language_code>_<country_code>.display`. For example, if English is selected as language and United States is selected as country, for the Concept named `Applicant`, the Display Model file is saved with the file name `Applicant_en_US.display`.

Editing a Display Model

You can define display text for each property of the artifact and also can hide a property from displaying in WebStudio using the Display Model editor.
The following fields are not editable but can be set up only while creating Display Model.

- Target Entity
- Language
- Country
- Variant

Procedure

1. In TIBCO BusinessEvents Studio, double click the Display Model file to open it in the form editor.
2. Enter the display text for the artifact (Concept or Event) in the **Target Entity Display Text** field.
3. In the **Display Properties** section, configure the display text for the properties of the artifact.
   a. Click **Add**, select the properties for which you need to assign the display text, translate, or hide, and click **Finish**

      The selected properties are listed.

   b. Select a property and enter the required display text in the **Display Text** field.

      Note: In the case of Arabic characters, do not use accents, as they are not supported in TIBCO BusinessEvents.

   c. Select the **Hidden** checkbox, if you do not want the property to appear in the WebStudio UI.
4. After you have added the display text for the required properties, save the Display Model.

Handling Null Properties

There are several ways of handling null concept property values:

Enabling Use of the Nillable Attribute

The presence of the `xsd:nillable` attribute in an XSD element means that the corresponding element in the XML file permits null values.

Setting `tibco.be.schema.nil.attribs=true` in `studio.tra` causes the `xsd:nillable` attribute ("`xsd:nillable=true`") to be set on all elements in the TIBCO BusinessEvents concept XSD. When an element in the XML file generated using that XSD has a null value, the `xsi:nil="true"` attribute is set on that element.

When set to false, the `xsd:nillable` attribute is not added and the corresponding XML file does not treat empty elements as null values.

In the absence of the `xsd:nillable` attribute in the XSD element, a corresponding empty element in the XML file is assumed to have a value. Elements that have no value are treated as empty strings ("").

Note: Effect on schema generation tool

The setting for this property affects the concept XSD files generated using the Generate Schema utility.

Enabling Null Property Values to Appear When Serializing Concepts to XML or JSON

By default concept properties with null values are excluded when concept objects (instances) are serialized to XML or JSON. You can override this behavior.

Setting the following property to false in the `studio.tra` file causes properties with null values to be included in the XML representation of a concept:

```
tibco.be.schema.exclude.null.props=false
```

To enable null property value while serializing concepts to JSON, add the following property in the CDD editor or to the `be-engine.tra` file:

```
tibco.be.schema.exclude.null.props=false
```

Examples of Nillable Attribute and Null Properties Settings

These examples illustrate the effect of the following properties on concept serialization:

```
tibco.be.schema.nil.attribs
tibco.be.schema.exclude.null.props
```

If Null Properties are Excluded

```
tibco.be.schema.nil.attribs= true or false
tibco.be.schema.exclude.null.props=true
```

Suppose a Customer concept instance has no value for its CustomerName property. By default, the CustomerName property is excluded from the XML output. The output might look like the following:

```
<CustomerID>111</CustomerID>
<Country>Japan</Country>
<City>Tokyo</City>
```

If null properties are excluded when concepts are serialized, the `tibco.be.schema.nil.attribs` property has no effect on concept serialization.

If Null Properties are Included and the Nillable Attribute is Set

```
tibco.be.schema.nil.attribs=true
tibco.be.schema.exclude.null.props=false
```

The output for the Customer concept instance shown above would be as follows, where there is no value for the `CustomerName` element in the concept instance:

```
<CustomerID>111</CustomerID>
<Country>Japan</Country>
<City>Tokyo</City>
<CustomerName
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:nil="true"/>
```

If Null Properties are Included and the Nillable Attribute is not Set

```
tibco.be.schema.nil.attribs=false
tibco.be.schema.exclude.null.props=false
```

In this case, each null property is considered to be an empty string, and is represented, for example, as follows:

```
<CustomerName/>
```

Special Treatment of Numeric and Boolean Null Values

If you enable null values to be output to XML (see Enabling Null Property Values to Appear When Serializing Concepts to XML or JSON), then you may also want to configure additional properties for defining how to treat null values for numeric types and Booleans, as explained in this section.

TIBCO BusinessEvents does not implicitly support null values for numeric types and Booleans. This can lead to interoperability issues when working with external sources, such as databases, which do permit blank (null) values.

To address such issues, you can enable special treatment of numeric null values at design time. At runtime, TIBCO BusinessEvents then uses a special numeric value for each numeric datatype, and (by default) `Boolean:FALSE` for Booleans, to represent a null value. Default special values are provided and you can override the defaults at runtime (see Property Reference for Null Property Handling).

The special numeric values that indicate null are used in TIBCO BusinessEvents when serializing and deserializing a concept to and from its XML representation, and when performing various operations on database concepts and the database tables to which they are linked.

The special values that indicate null appear only in TIBCO BusinessEvents. The appropriate null value is used in the XML or database representation of the concept property.

Conversely, when deserializing or importing a concept instance, TIBCO BusinessEvents represents null values in the source using these special values that indicate null in the concept instance.

To enable special treatment of numeric null values, set the following property in `studio.tra`:

```
tibco.be.schema.treat.null.values=true
```

At runtime you can override the default values that indicate null. See Setting Runtime Properties for Special Treatment of Null Values.

Summary

Set the following properties in BE_HOME`/studio/eclipse/configuration/studio.tra` as desired:

- To enable null property values to appear when serializing concepts to XML, add the following property and set it to false:

  ```
  tibco.be.schema.exclude.null.props=false
  ```

- To enable use of the nillable attribute in the concept XSD, add the following property and set it to true:

  ```
  tibco.be.schema.nil.attribs=true
  ```

- To enable special handling of null properties, add the following property and set it to true:

  ```
  tibco.be.schema.treat.null.values=true
  ```

Setting Runtime Properties for Special Treatment of Null Values

If you have enabled special treatment of null numeric properties, you can override the default special values that indicate null values of various data types in TIBCO BusinessEvents (shown in Property Reference for Null Property Handling).

To override the default values perform the following steps.

Procedure

1. In TIBCO BusinessEvents Studio, open the CDD editor for the project and add the following properties at the cluster level properties sheet. Provide the special values as desired:

   `tibco.be.property.int.null.value=`*`value`*

   `tibco.be.property.long.null.value=`*`value`*

   `tibco.be.property.double.null.value=`*`value`*

   `tibco.be.property.boolean.null.value=`*`value`*

   Choose values that will not be misinterpreted as literal values.

2. Save the file.

3. Deploy the project with the updated CDD file.

Property Reference for Null Property Handling

Set the following properties in the `studio.tra` file as needed to configure the output for your needs before you generate the EAR file.

Properties for Null Property Handling

Property	Note
Properties Set in ***BE_HOME***/studio/eclipse/configuration/**studio.tra**	
`tibco.be.schema.nil.attribs`	Setting this property to true causes the `xsd:nillable` attribute (`"xsd:nillable=true"`) to be set on all elements in the TIBCO BusinessEvents concept XSD. See Enabling Use of the Nillable Attribute Possible values are true and false. Default is false.
`tibco.be.schema.exclude.null.props`	When this property is set to true, null-valued concept properties are not output when the concept is serialized to XML or JSON. When set to false, null-valued concept properties are output to XML or JSON. See Enabling Null Property Values to Appear When Serializing Concepts to XML or JSON . Possible values are true and false. Default is true.
`tibco.be.schema.treat.null.values`	Setting this property to true causes TIBCO BusinessEvents to use special numeric values that indicate null for numeric datatypes. The special numeric values are set using the properties listed next. Possible values are true and false.

Property	Note
	Default is false.
Properties Set in CDD at the cluster level	

`tibco.be.property.int.null.value`
`tibco.be.property.long.null.value`
`tibco.be.property.double.null.value`
`tibco.be.property.boolean.null.value`
	These properties define a special value that indicates null. Use a value that will not be confused with an actual numeric value.
	
	These properties are used only if `tibco.be.schema.treat.null.values` is set to true.
	
	Default values for each numeric datatype are the following Java constants:
	
	`int: Integer.MIN_VALUE`
	
	`long: Long.MIN_VALUE`
	
	`double: Double.MIN_VALUE`
	
	`boolean: Boolean.FALSE`
	
	For Integer and Long these constants represent the most negative value. For Double the constant represents smallest positive nonzero value (4.9e-324).

Diagrams

TIBCO BusinessEvents provides several kinds of diagrams, which are visualization tools that help you understand and analyze even very large and complex projects.

Note: The diagrams in TIBCO BusinessEvents are based on Unified Modeling Language (UML), but are **not** completely compliant to UML.

Using the diagrams you can show or hide details. You can open editors for any project component in a diagram. You can also create snapshot images of diagrams to share project information with other personnel.

The main types of diagrams, and the project elements that use them, are as follows:

Selected Entity Project diagrams

These diagrams display all the selected project resources, and how they interact with one another. External resources such as XSD and WSDL files are not shown. You can either create a project diagram for an entire project, or only for the selected resources of a project. You can also choose to run the project analyzer while creating a project diagram, which analyzes the project resources and gives a report in the Problems view.

Dependency diagrams

These diagrams show the relationships between the selected project resource and its dependent resources. They are available for channels, concepts, state models, scorecards, events (all types except advisory events), rule functions, and rules.

Sequence diagrams

These diagrams show how project resources are called into use at run time. They are available for events (all types except advisory events), rule functions, and rules.

Concept model diagrams

These diagrams show the concept model for a project. You can view a model diagram for all the concepts in a project.

State model diagrams

A state model allows you to model the life cycle of a concept instance.

Event model diagrams

These diagrams show the event model for a project. You can view a model diagram for all the events in a project, except for advisory events.

Types of Diagrams Available for Different Resource Types

Project Resource	Project	Dependency	Sequence	Concept Model	Event Model
Channel	Yes	Yes			
Concept	Yes	Yes		Yes	
Domain	Yes	Yes			
State Model	Yes	Yes			
Scorecard	Yes	Yes			
Event	Yes	Yes	Yes		Yes
Rule Function	Yes	Yes	Yes		
Rule	Yes	Yes	Yes		
Shared Resource	Yes	Yes			

Working with Diagrams

You can work with diagrams in a variety of ways for your own information, and you can use them to share aspects of your project with others in a visual way that is easy to understand. This section presents some common tasks and lists the various tools you can use to work with all types of diagrams.

- Configuring Diagram Preferences and Properties
    - Diagram Preferences

- Diagram Properties

  Use the following properties to handle diagrams that are too complex to display in a timely manner. You can adjust the values in the BE_HOME`/studio/eclipse/configuration/studio.tra` file to define upper limits:

  ```
  #Diagram properties
  be.studio.project.diagram.nrEdges=3000
  be.studio.project.diagram.nrMaxEdges=1500
  ```

  If the number of edges in the diagram exceeds the limit defined by `be.studio.project.diagram.nrEdges`, then the usage links in the diagram are hidden. Usage links are links from a rule or rule function to other rules or rule functions it uses.

  If the number of edges in the diagram exceeds the limit defined by `be.studio.project.diagram.nrMaxEdges`, then symmetric layout is run as default rather than hierarchical because symmetric layout is faster and less expensive.

- Different Ways to Create Diagrams

  You can create diagrams in various ways as shown in the table below.

Method of creating diagrams	Diagrams you can create
In BusinessEvents Studio Explorer, right-click a resource or a project, and select the appropriate menu option.	Selected Entity Project Diagram, Concept Model Diagram, Event Model Diagram
In Studio Explorer, select a resource and right-click.	Selected Entity Project Diagram, Concept Model Diagram, Event Model Diagram, Dependency diagram
In a resource editor, click the appropriate diagram button in the top right corner. Concept Model Diagram Event Model Diagram Dependency Diagram Sequence diagram	Concept Model Diagram, Event Model Diagram, Dependency Diagram, Sequence Diagram
In a Selected Entity Project diagram, right-click a resource.	Dependency Diagram
In a Selected Entity Project diagram, right-click anywhere on the canvas.	Selected Entity Project diagram

Common Tasks

You can perform the following common tasks on all types of diagrams:

- Hover the mouse over a resource to view a tooltip with details.
- Double-click any resource represented in the diagram to open its editor.
- Drag resources to any location in order to change the layout as desired.

- Drag any link to change the layout as desired.
- Click any arrowhead to move the arrow to any other element in the project. When you hover the mouse over an arrowhead, you see a small graphic on it( ).
- Double-click the black and white plus signs in a dependency or sequence diagram to unfold the next level of dependency.

Diagram Tools

You can work with diagrams with the help of the following tools available from the Diagram menu:

- Interactive Tools, such as Select, Pan, Magnify, Marquee Zoom, Interactive Zoom, Link Navigator
- Layout, such as Default Layout, Circular Layout, Orthogonal Layout, Symmetric Layout, Rectilinear Hierarchical Layout, and Oblique Hierarchical Layout
- Print tools: Print Setup, Print Preview, Print
- Export to Image
- Labeling
- Link Routing
- Incremental Layout
- Fit In Window

The above tools are also available on the toolbar when you create or display a diagram. The toolbar also has three additional tools listed below:

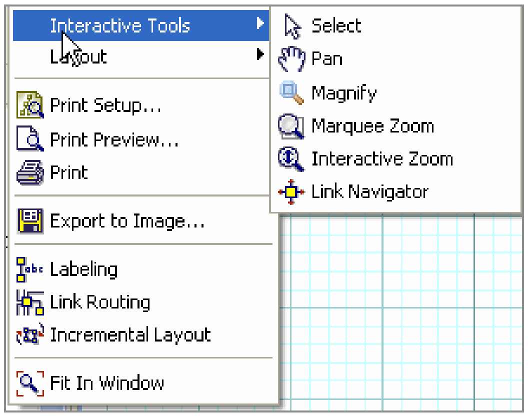
- Refresh Diagram
- Zoom Percentage
- Search Diagram Entities

Note: Diagram menu and the toolbar described in this section are available only when you create or open a diagram.

For more details about the diagram tools, see Diagram Options and Tools Reference.

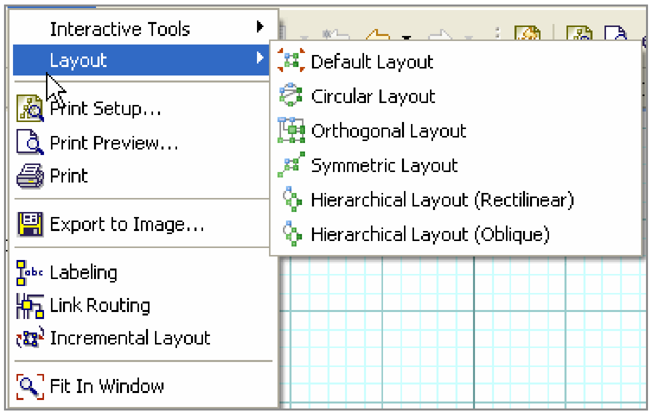
Interactive Tools

To access the Interactive Tools, select **Interactive Tools** from the **Diagram** menu. They are also available on the toolbar when you create or display a diagram. Interactive Tools let you select, pan, and magnify diagrams. They also include tools such as marquee zoom, interactive zoom, and link navigator. For more details about these tools, see Diagram Options and Tools Reference.



Layout

You can access the **Layout** menu from the **Diagram** menu. They are also available on the toolbar when you create or display a diagram. Layout options let you change the layout of a diagram to Circular, Orthogonal, Symmetric, or Hierarchical. For more details about these tools, see Diagram Options and Tools Reference.



Context Menu Diagram Tools

Diagram tools are available as context menus for diagram canvas and objects. The following table lists these context menu options.

Context Menu Options for Canvas and Objects

Context Menu Option	Available for
Selected Entity Project Diagram	Canvas-Selected Entity Project Diagram
Export to Image...	Canvas-All diagrams
Print Setup...	Canvas-All diagrams
Print Preview...	Canvas-All diagrams
Fit In Window	Canvas-All diagrams
Create Dependency Diagram	Object-Selected Entity Project Diagram
Fold One Level	Object-Selected Entity Project Diagram
Fold N Levels...	Object-Selected Entity Project Diagram
Fold All Levels	Object-Selected Entity Project Diagram
Edit	Object-All diagrams

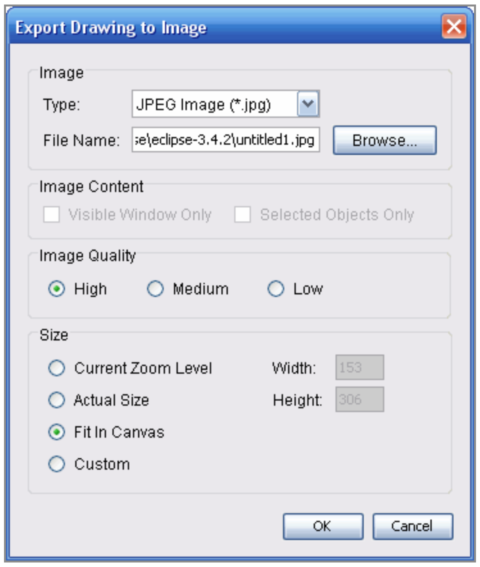
See Diagram Options and Tools Reference for more details about the diagram tools.

Exporting a Diagram to an Image

Using the Export to Image option you can save the diagrams locally for later use.

Procedure

1. From the top menu, select **Diagram > Export > to Image**. You see the Export Drawing to Image dialog.



Export Drawing to Image

Image
Type: JPEG Image (*.jpg)
File Name: :e\eclipse-3.4.2\untitled1.jpg [Browse...]

Image Content
☐ Visible Window Only ☐ Selected Objects Only

Image Quality
[✓ High] Medium Low

Size
Current Zoom Level
Actual Size
[✓ Fit In Canvas]
Custom
Width: 153
Height: 306

[OK] [Cancel]

2. Select the type of the image from the Type list.
3. In the **File Name** field, specify the file name along with the location.
4. Select the checkboxes **Visible Window Only** and **Selected Objects Only** appropriate to specify the content of the image.

   Note: Only if you select an object in the diagram, then **Selected Objects Only** is enabled.

5. Specify the Image Quality as appropriate.
6. Specify the size, and click OK.

Result

The diagram is exported to an image, and saved on the system.

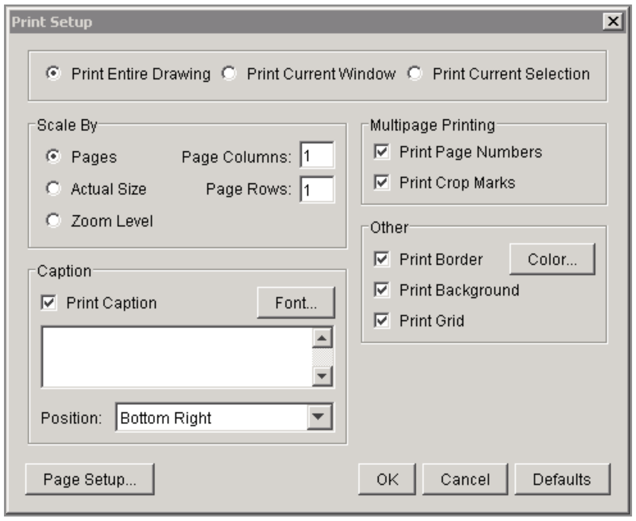
Printing a Diagram

You can set up preferences, preview the print job, and then print the diagram.

Using the Print Setup dialog you can specify the print setup for a diagram.

Procedure

1. Display the diagram, and from the top menu select **Diagram > Print Setup**.



Print Setup

[✓ Print Entire Drawing] Print Current Window Print Current Selection

Scale By

[✓ Pages] Page Columns: 1

Actual Size Page Rows: 1

Zoom Level

Multipage Printing

☑ Print Page Numbers

☑ Print Crop Marks

Other

☑ Print Border

☑ Print Background

☑ Print Grid

Caption

☑ Print Caption

Position: Bottom Right

2. Select an appropriate option to print the entire diagram, or to print only the current window, or to print the selection.
3. In the **Scale By** section, select an option to scale the diagram by pages, by actual size, or by zoom level.
4. If you want multiple prints, you can specify if you want the page numbers and crop marks to be printed by clicking to select the respective checkboxes.
5. If you want a caption for your diagram, you can type a caption in the text box that appears on enabling the **Print Caption** checkbox. You can also select a Font for this caption, and its position on the page.
6. If you want to have borders for your diagrams, click to select the **Print Border** checkbox. Click **Color...** to specify the color of the border.
7. If you want to print the background of the diagram, click to select the **Print Background** checkbox.
8. Use **Page Setup** to specify the size, orientation, and other page properties.
9. If you want to reset all the options to the defaults, click **Defaults**.
10. Click **OK**.
11. To preview and print the diagram, select **Diagram > Print Preview**. An image of the diagram to be printed is displayed.

12. From the top menu select **Diagram > Print**.

Setting Diagram Preferences

You can set the preferences for all types of diagrams.

Procedure

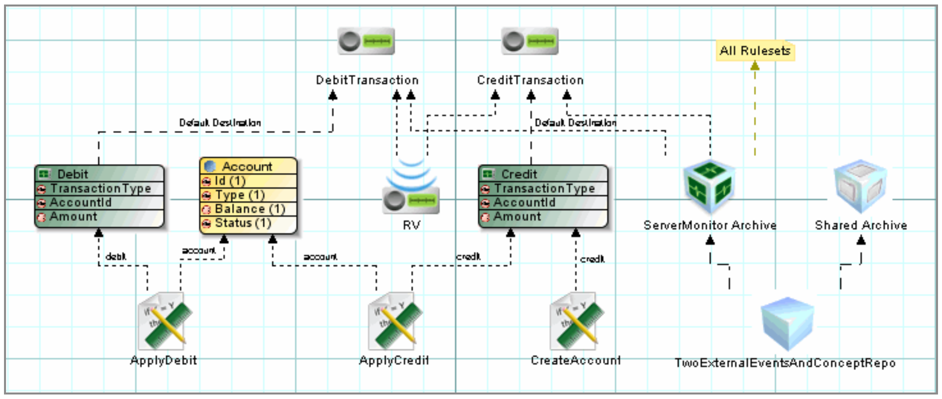
1. From the top menu, select **Window > Preferences**.
2. In the Preferences dialog, expand **TIBCO BusinessEvents**, and then expand **Diagram**.
3. Select any type of diagram.

   On the right side, you see preferences for the selected diagram type.
4. Edit the preferences, and click **Apply**.

   OR

   To set all preferences to their default values, click **Restore Defaults**.

Project Analyzer and Selected Entity Project Diagrams



Project Analyzer and the Selected Entity Project diagram work together to provide insights into your project. Project Analyzer is a document generated in the Problems view area. Selected Entity Project diagram provides a visualization of the whole project or of selected project elements, including dependencies. Selected Entity Project diagrams can display all the resources in a project, how they interact with one another, and how they use or update other resources at run time. You can see concepts, events, rules, rule functions, channels, destinations, scorecards, shared resources, and all other resources of a particular project in a Selected Entity Project diagram. It also shows all dependencies in a project.

You can set preferences to determine whether project analyzer runs when you create a Selected Entity Project diagram or not. You can use a menu option to create the project analyzer report without creating the Selected Entity Project diagram.

Types of Links

A Selected Entity Project diagram has three types of links:

- Links, represented by continuous lines, indicate inheritance or containment.
- Scope Links, represented by dashed lines, indicate resources in the scope.
- Usage Links, represented by dashed lines with dots, indicate resources that can be used.

Advantages of Project Analyzer and Selected Entity Project Diagrams

Using Project Analyzer and Selected Entity Project Diagrams together helps you understand your project in these ways:

- Helps you understand how all the selected project resources are connected and how they interact.
- Analyzes rules and how they modify ontology.
- Lets you search entities in a diagram.
- Lets you choose the entities to be displayed in a diagram by using Project Filters.
- Displays statistics of project resources (number of concepts, events, state machines, and so on) in the properties view.
- Enables you to share project design, logic, deployments with others in a format they can read without having TIBCO BusinessEvents Studio, as well as print the entire project according to page setup.

The Project Analyzer report helps you analyze the project by performing the following tasks:

- Finds rules that can never be fired
- Finds rule functions that are never used
- Finds events that are never used/may never be fired
- Finds destinations with no default events
- Finds domain models that are never used, or that are not associated with entity properties
- Finds state models that are orphaned, or that are not associated with any concepts

Project Analyzer and Selected Entity Project Diagrams

You can show the diagram for an entire project or for selected elements only. You can also filter the diagram to show only certain types of project element.

✓ **Tip:** You can choose whether to run Project Analyzer whenever you create a Selected Entity Project diagram. To do this, from the **Window** menu, select **Preferences > TIBCO BusinessEvents > Diagrams** > **Project**. Select the **Run Analysis When Creating View** checkbox, on the right.

You can also run **Project Analyzer** separately in either case. See Project Analyzer and Selected Entity Project Diagrams .

To Run Project Analyzer Without Selected Entity Project Diagram

Right-click the top-level project folder, and click **Analyze**. Project Analyzer displays the report in the Problems view.

Create a Selected Entity Project Diagram

Procedure

1. Do one of the following to create a Selected Entity Project diagram for the entire project:
   - In BusinessEvents Studio Explorer, right-click a project resource and select **Create Selected Entity Project Diagram**.
   - Double-click the `*.projectview` file in the Studio Explorer.

     OR, to create a Selected Entity Project diagram for selected resources
   - In Studio Explorer, select resources of interest, right-click, and select **Create Selected Entity Project Diagram**.

   > **Note:** When you select a folder all elements in that folder are selected.
2. The Selected Entity Project diagram appears in the editor area. The diagram tab label reflects the project name and the type of diagram: *`projectName`*`.projectview`, for example, `ExternalEventRepo.projectview`.

   By default, the Selected Entity Project diagram displays the selected objects and their immediate dependencies.
3. You can change the depth of dependencies shown using preferences.
4. To filter what you see, select and clear the options in the Project Filter list within the Palette view.
5. Click **Apply**.

Result

Options show or hide elements of the specified type, or group them:

Show Options

Show Concepts, Show Events, Show Decision Tables, Show Domain Model, Show State Models, Show Archives, Show Rules, Show Rule Functions, Show Scorecard, Show Channels, Show Scope Links, Show Usage Links, Show Archived Destinations, Show Archived Rules, Show Archived Rules (All), Show Rules in Folders, Show Tooltips. If add-on products are used additional options may appear.

Group Options

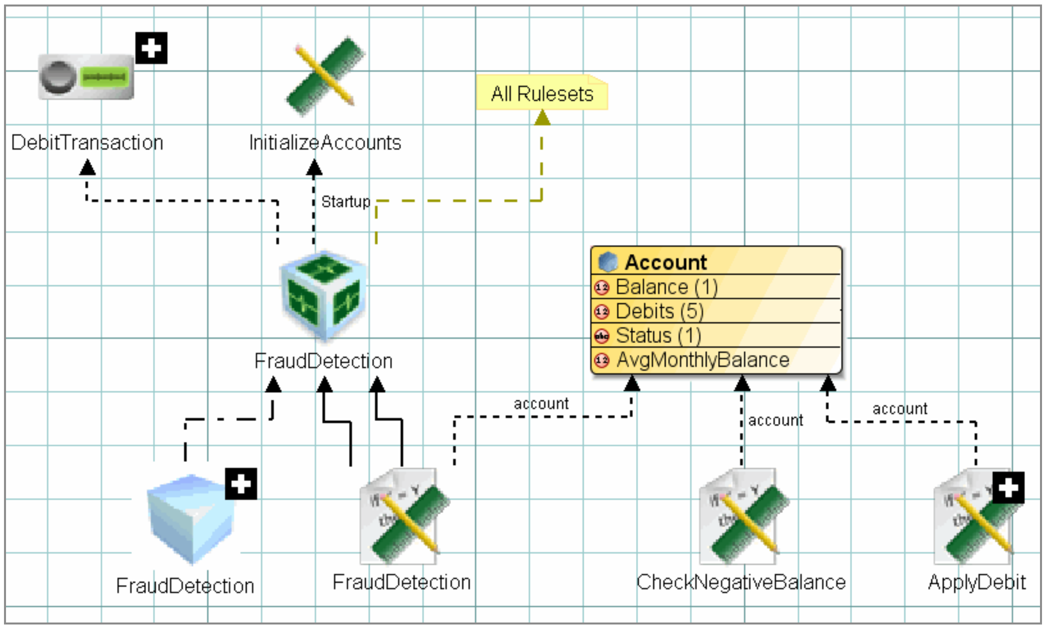
Group Concepts, Group Events, Group Rules, Group Rule Functions. The filter options in the Palette view are loaded from diagram preferences.

Dependency Diagrams

A dependency diagram shows the dependencies for a selected project resource or for the entire project (Selected Entity Project Diagram).

When two resources are dependent and one changes, it can cause changes to the other resource as well. You can control how many levels of dependencies to view in a dependency diagram - one, two, or all levels.

Figure 22: Dependency Diagram of a Entity Project



Rule actions are not represented in rule dependency diagrams. Use Debugger to analyze rule actions. See Testing and Debugging Projects.

Create a Dependency Diagram

Procedure

1. Do one of the following:
   - In BusinessEvents Studio Explorer, right-click a project resource and select **Create Dependency Diagram**.
   - Open the project element for editing and click the **Dependency Diagram** (

) button in the top right of the editor.
   - In a Selected Entity Project diagram, right-click a resource and select **Create Dependency Diagram**.

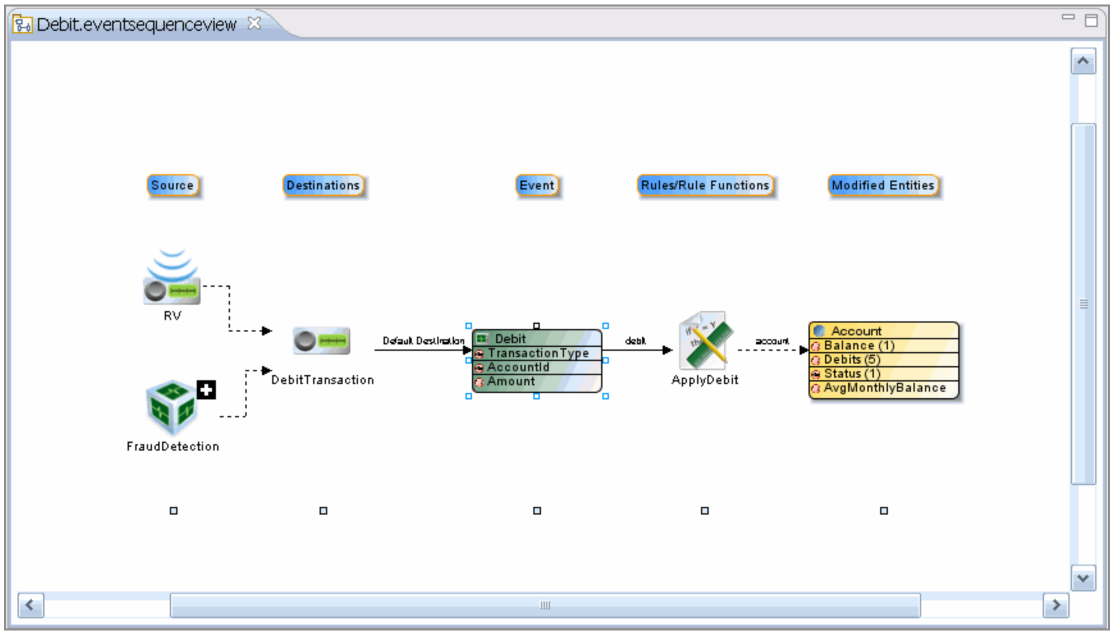
     The dependency diagram appears in the editor area. By default, one level of dependency is shown which displays only direct dependents. White crosses indicate hidden dependencies. The default appearance is also driven by the diagram preferences.

     The diagram tab label reflects the item being graphed and the type of diagram: `*itemName*.*item-type*dependencyview`, for example, `Transaction.eventdependencyview`.
2. As desired, select a dependency level from the palette:
   - One: Shows the direct dependencies of the selected item
   - Two: Shows the direct dependencies of the selected item as well as the direct dependencies of its direct dependencies
   - All: Shows all dependency relationships that involve the selected item

Sequence Diagrams

Sequence diagrams capture the behavior of objects and the messages that are passed between them. You can view sequence diagrams for rule functions, rules, and events.

Figure 23: Sequence Diagram of a Project



Create a Sequence Diagram

Open an event, rule function, or rule for editing and click the Sequence diagram ( ) button in the top right of the editor.

The sequence diagram for the item appears in the editor area.

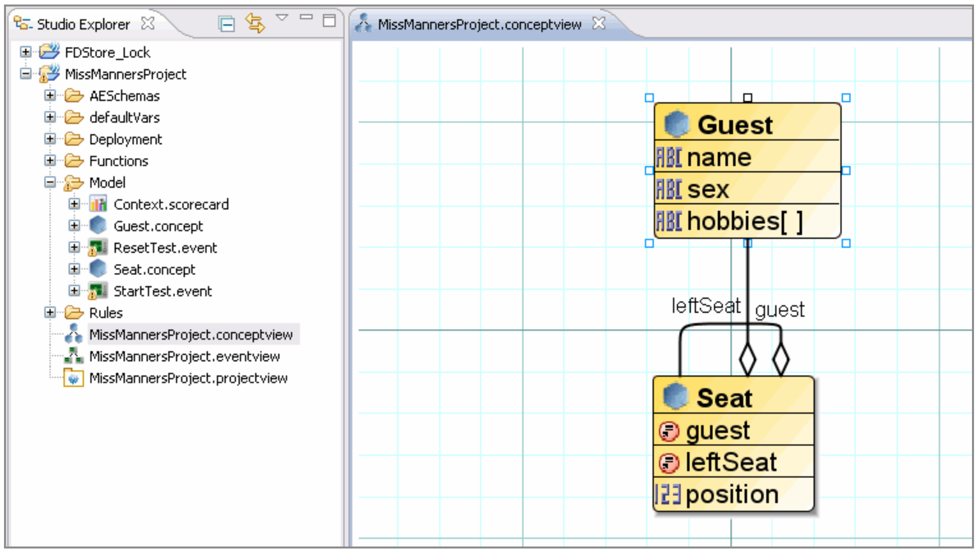
The diagram tab label reflects the item being graphed and the type of diagram: `*itemName*.*item-type*sequenceview`. For example, `Transaction.eventsequenceview`.

Concept Model Diagrams

The concept model diagrams show inheritance, containment, and reference relationships between all concepts in a project.

Concept model diagrams are created fresh each time you view them (they do not persist on disk).

Figure 24: Concept Model Diagram



Create a Concept Model Diagram

Do one of the following:

- Right-click a resource or a project in BusinessEvents Studio Explorer and select **Create Concept Model Diagram**
- Open a concept editor and click the Concept Model Diagram ( ) button in the top right of the editor
- Double-click the `*.conceptview` file in the Studio Explorer.

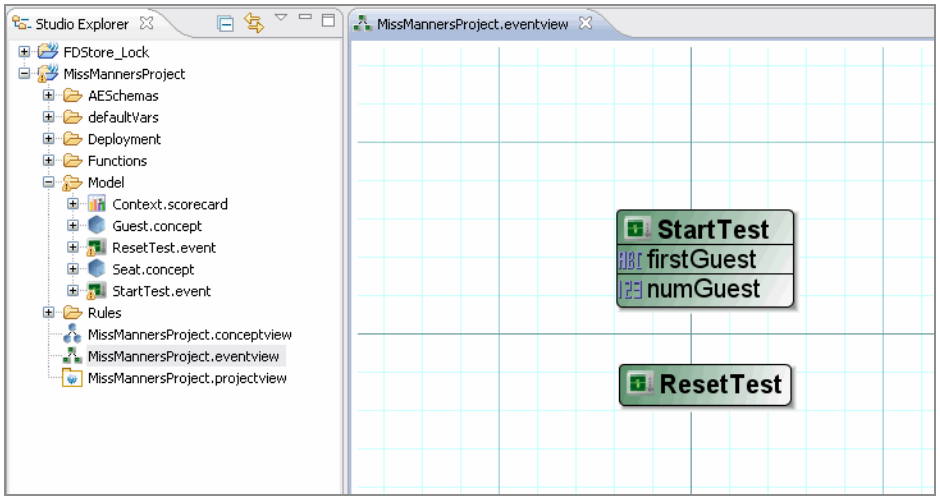
The concept model diagram for that project appears, showing only its direct dependents (Dependency Level One).

Event Model Diagrams

The event model diagrams show inheritance relationships between all types of events, except for advisory events.

Event model diagrams are created fresh each time you view them. They do not persist on the disk.

Figure 25: Event Model Diagram



Create an Event Model Diagram

Do one of the following:

- Right-click a resource or a project in BusinessEvents Studio Explorer and select **Create Event Model Diagram**.
- Open an event editor and click the **Event Model Diagram** () button in the top right of the editor. The event model diagram for that project appears, showing only its direct dependents (Dependency Level One).
- Double-click the `*.eventview` file in the Studio Explorer.

Diagram Options and Tools Reference

Most diagram options and tools are self-explanatory, and are just shown in the images. This section provides details for a few that require some explanation.

TIBCO BusinessEvents Diagram Tools Reference

Button	Description
Interactive Tools Menu Options and Toolbar Buttons	

Button	Description
	Select (Pointer) tool.
	Pan tool.
	Magnify Select Magnify and hover over the diagram to display a magnified section of the diagram.
	Zoom Drag the cursor diagonally to define the area to which you want to zoom.
	Interactive Zoom Drag the cursor up to zoom out. Drag the cursor down to zoom in.
	Link Navigator Shows the path of a link using a simple animation.
Layout Menu Options and Toolbar Buttons	
	Default Layout Returns the diagram to the default layout. To display the layout list, click the arrow next to the button and select a new layout from the list of options.
	Circular Layout It is available in the layout list. See Circular Layout.
	Orthogonal Layout It is available in the layout list. See Orthogonal Layout.
	Symmetric Layout It is available in the layout list. See Symmetric Layout.

Button	Description
	Hierarchical Layout (Rectilinear) It is available in the layout list. See Hierarchical Layout.
	Hierarchical Layout (Oblique) It is available in the layout list. See Hierarchical Layout .
Other Diagram Menu Options and Toolbar Buttons	
	Print Setup Sets options for printing the diagram or elements of the diagram. See Printing a Diagram.
	Print Preview Lets you see how the printed diagram would look.
	Print Prints the diagram.
	Export to Image Exports the diagram to one of the five available image file formats, and lets you save the image file locally for future reference. See Exporting a Diagram to an Image.
	Labeling Prevents labels from overlapping with other elements and labels in the diagram.
	Link Routing Redraws the diagram, attempting to redraw only the links, leaving resource nodes in the same size and position where possible. The behavior of link routing is affected by the preference options Fix Node Size and Fix Node Position.

Button	Description
	Incremental Layout See Incremental Layout.
	Fit In Window Zooms the diagram to fit the size of the current diagram editor view.
Additional Toolbar Buttons	
	Refresh Diagram If you make changes in the project, clicking Refresh Diagram updates the diagram with the changes.
30%	**Zoom Percentage** Zooms the diagram to the specified percentage.
	Search Entities Allows you to search for TIBCO BusinessEvents entities in the diagram.

Layout Options

Layout options refer to the style of the diagram that is rendered in the diagram editor, Rule Debugger, and Dependency panel.

Note: Additional Layout Options

Various options affect the appearance of a diagram, no matter what layout is chosen. (The hierarchical layout offers additional options.)

Circular Layout

This layout is useful for ontologies where nodes tend to have a clustered (ring or star) structure (where each main node has a starburst of related nodes).

Orthogonal Layout

The orthogonal layout style uses only horizontal and vertical links. Diagrams are drawn quickly.

The algorithm places highly connected nodes closer together, resulting in a more compact diagram. Even when lines overlap, the algorithm ensures that they are still easy to follow. Because this style has no hierarchical or other visual constraints, the resulting diagrams are often very clear.

Symmetric Layout

The symmetric layout style looks for and emphasizes the symmetries in a project topology. It can produce a pleasing visual result, if there are no reasons to arrange the nodes. For example, there is no hierarchy or ring-clustering inherent in the structure of the nodes.

See Working with Diagrams for a property that affects whether symmetric or hierarchical layout is used as the default, depending on number of edges.

Hierarchical Layout

The hierarchical layout style indicates dependencies by positioning the nodes at different levels. The hierarchical layout style is useful when you need to show precedence relationships in the ontology.

Hierarchical layout is the default. However, see Working with Diagrams for a property that affects whether symmetric or hierarchical layout is used as the default, depending on number of edges. You can use Layout Preferences to determine in the direction of the hierarchy.

Preferences and options specific to the hierarchical layout are:

- Orientation Options (**Edit > Preferences**)—Left to Right, Top to Bottom, Right to Left, Bottom to Top
- Routing Options (**View > Layout**)—Orthogonal or normal (polyline) routing
- Routing Options (**Edit > Preferences**)—Orthogonal or polyline routing.

Incremental Layout

Choose Incremental Layout if you want the main arrangement of the diagram to remain stable when you make changes to your project and only re-route the entities and links that have changed since the last rendering.

Keeping most things in the same place makes it easier to see how changes you have made in the project affect the diagram.

If you want to refresh the entire diagram, click the desired layout option. The program has greater freedom to optimize the layout. The result is likely to be a more pleasing arrangement.

To perform an incremental layout update, do one of the following:

- Select **View > Layout > Incremental** Layout.
- Click the **Incremental Layout** button on the toolbar.

Preferences

You can set up preferences in TIBCO BusinessEvents Studio on different levels and for different resources.

Note: BusinessEvents Studio supports only left to right languages. If your operating system is configured to use any right to left language such as Arabic and you are using BusinessEvents Studio, you must not use the default layout of the operating system. Also, in **Window > Preferences > General > Globalization**, ensure that **Enable bidirectional support** is not selected and **Graphical layout direction** is selected as Left to right.

- TIBCO BusinessEvents (Top Level)

  Using the top level preferences, you can do the following:

  - Show or hide hover (tooltip) information in Catalog Functions view.
  - Show or hide the confirm open perspective message pop-up.
  - Switch or not switch to the default perspective every time an editor opens.
  - Clear all the "do not show again" settings and show all hidden dialogs again.
- Build Path and Classpath Variables

  Maintains a list of the classpath variables that have been defined. You can add, update, or remove classpath variables.

- Code Generation
    - Compilation Mode options are File System or In Memory
    - Source Java Version options are 1.5 or 1.6
    - Target Java Version options are 1.5 or 1.6
- Code Generation and Ignored Resources

  Maintain the list of ignored resource patterns here: **Code Generation > Ignored Resources**. These preferences define what file types are excluded from the EAR file shared archive section at build time.

  You can add and remove patterns, and select which existing patterns to ignore or stop ignoring.
- Compare/Merge, Decision Table

  Compare/Merge preferences are used for comparing two decision tables.
- Diagram Preferences

  In addition to a global preferences section, there are sections for each type of diagram.
- Error/Warning Preferences

  You can enable and disable error and warning messages for each type of resource in a project. Resource extensions are shown so you can select which ones to enable or disable. By default all are selected. If you make changes, a prompt appears asking if you want to rebuild the project.
- Printing Preferences

  These preferences apply to printing of diagrams, windows, and selections.
- Rules and Rule Debug Preferences

  In addition to the standard Eclipse Run/Debug preferences, TIBCO BusinessEvents requires some rule debug preferences.
- Rules Management Server Preferences

  These preferences are used with TIBCO BusinessEvents WebStudio.

Setting Preferences

To access preferences select **Window > Preferences** and expand **TIBCO BusinessEvents**.

Procedure

1. From the Window menu, select **Preference**s > **TIBCO BusinessEvents**.
2. Expand the options on the left and select each option to view a panel of options on the right.

   details about each set of options are provided in sections following.
3. Change the options from the list as desired, and then click **Apply**.
4. Click **OK** to see your preferred settings.

Decision Table Related Preferences

Decision Table Preferences

The Table View preferences set the initial state of the following features. Text in parentheses shows the names of corresponding buttons in the Decision Table editor. The settings can be toggled in the UI, but their initial state is set by preferences you set:

- Show column alias if present.
- Automatically resize columns to fit content (Fit Content button.)
- Automatically resize rows to fit content.
- Automatically merge rows (Merge Rows button.)
- Show expanded text (Show Text button.)
- Show domain descriptions if present.
- Show column filter.
- Show titled border.
- Use existing IDs when importing.
- Export column alias.
- Automatically update decision tables on change in virtual rule function arguments.
- Show condition area string.
- The Use Non-resizable Editor sections option prevents or allows users from resizing editor sections.

Decision Table Analyzer Preferences

You can control the following behavior of the decision table analyzer, as well as its appearance:

- Highlight partial ranges
- Use domain model for table completeness
- Show analyzer contents while opening table

Decision Table Appearance Preferences

These preferences let you set the color scheme for decision tables. For each item in the list, you can set color preferences. For condition data and action data, you can specify a font.

Compare/Merge Preferences

The tabbed dialog is in the TIBCO BusinessEvents preferences. It applies to decision tables.

In the **Structure Compare** tab (shown above), set preferences for initial settings and colors. The effect of your choices is shown in the lower panel.

In the **Text Compare** tab (shown below), you can set preferences for text. The effect of your choices is shown in the lower panel.

The **Merge** tab has one option: Allow columns to be automatically merged.

Diagram Preferences

The diagram preferences apply to the following types of diagrams:

- Concept
- Dependency
- Event
- Project
- Sequence
- State Model (This option is available only if TIBCO BusinessEvents Data Modeling is installed.)

Preferences set using a diagram’s palette are applicable only to the displayed diagram.

Preferences set using the Preferences dialog define the default preferences for diagrams of that type.

Preferences for specific diagrams override global preferences, where the same settings exist.

Global Diagram Preferences

Global Diagram Preferences

Option	Description
Reset Tool After Changes	If enabled, resets the tool to the Select tool after you add a node to the diagram. **Default:** Enabled.
Auto Hide Scrollbars	If enabled, then when the complete contents can display in the space available, scrollbars are hidden. If not enabled, scrollbars remain. Default: Auto hide scrollbars is enabled.
Show Tooltips	If enabled, you can see the tooltips when you hover the mouse on an element in the diagram. **Default:** Show Tooltips is enabled.
Link Types	Shows the links as straight lines or curved lines. **Default:** Straight.
Run Layout On Changes	After adding or deleting a node or an edge, refreshes the layout of the diagram. The layout options are None, Incremental, and Full. **Default:** None.
Animation	You can specify how a diagram behaves while it is changing from one layout option to another. Animation options allow you to see the nodes and edges moving from one arrangement to another. If you disable animation, the display simply switches from one kind of layout to another.

Option	Description
	Fade: When enabled, the diagram transitions from faded out to fully colored state. The transition completes in the duration specified in the Layout duration option. Interpolation: When enabled, the nodes and edges appear to move from one layout to another, when you select a different layout. The transition completes in the duration specified in the Layout duration option. Viewport Change: Enables the interpolation animation when the viewport changes as a result of zooming. If enabled, the zoom change is reflected gradually with animation. By default, animation is enabled along with Fade, Interpolation and Viewport Change. Duration: The time taken by animation to complete. Layout: Time, in milliseconds, for the layout change to complete. Applicable to the interpolation and fade options. **Default for Layout:** 500 Viewport Change: Time, in milliseconds, for the viewport change to complete. Default for Viewport Change: 1000
Opaque Movement	Interactive Zoom Sensitivity specifies the sensitivity of the mouse in interactive zooming. Pan Sensitivity specifies the sensitivity of the mouse in panning. **Default:** Opaque Movement is enabled, Interactive Zoom Sensitivity is 200.0, and Pan Sensitivity is 1.0.
Magnify Tool	The size the zoom window, and the level of zoom. **Default:** Window Size is 250, and Zoom Level is 3.

The preferences for all types of diagrams are listed in the following table. The preferences are organized alphabetically and not grouped according to the diagram types.

Specific Diagram Preferences (Alphabetical) (Sheet of)

Option	Description
Cluster Layout Style	Sets whether to show Selected Entity Project diagrams in a circular or symmetric clustered layout. **Default:** Circular.
Create view when analyzing	If enabled, creates a Selected Entity Project diagram when you choose to run the project analyzer. **Default:** Disabled.
Dependency Levels	Sets how many levels of dependencies to view in a dependency diagram - One, two, or all. **Default:** One.
Filter Options	Sets whether to show various project resources in a Selected Entity Project diagram: Show Concepts, Show Events, Show Decision Tables, Show Domain Model, Show State Machines, Show Archives, Show Rules, Show Rule functions, Show Scorecard, Show Channels, Show Scope Links, Show Usage Links, Show Archived Destinations, Show Archived Rules, Show Archived Rules (All), Show Rules in Folders, Show Tooltips, Group Concepts, Group Events, Group Rules, and Group Rule Functions. **Default:** All enabled except for Archived Rules, Group Concepts, Group Events, Group Rules, and Group Rule Functions.
Fix node and edge labels	Keeps the labels of the nodes and edges with their graphics. **Default:** Enabled.
Grid	Sets the display either to without grid, or grid with lines, or grid with points. **Default:** Lines.
Layout Style	Sets either the orthogonal or hierarchical layout. **Default:** Orthogonal.
Layout Quality	Options are Draft (lowest quality), Medium, and Proof (highest quality). It

Option	Description
	takes longer to generate a higher quality diagram than a lower quality diagram. A different algorithm is used in each case. **Default:** Draft.
Link Routing	Determines how links are routed when the hierarchical layout is used: Orthogonal: Routes the links using horizontal and vertical straight line segments (that is, using right angles) Polyline: Routes the links using straight line segments with arbitrary angles Overlapping lines are more likely with orthogonal routing than with polyline routing. With polyline routing the routing algorithm adds path nodes as needed to avoid overlapping lines. Note that the line segments can be joined using straight lines or curves in both cases. **Default:** Orthogonal.
Link Routing—Fix Node Positions	This setting affects link routing behavior for all layout options. If enabled, node positions do not change when you use the link routing feature. If disabled, link routing changes node positions as needed for clarity. **Default:** Disabled.
Link Routing—Fix Node Sizes	This setting affects link routing behavior for all layout options except hierarchical layouts. If enabled, node sizes do not change when you use the link routing feature. If disabled, link routing changes node sizes as needed for clarity. **Default:** Disabled.
Orientation	Defines the general direction in which the links display, to reflect hierarchical relationships between the entities. TIBCO BusinessEvents diagrams are not particularly hierarchical, but setting this option defines the general direction of the layout.

Option	Description
	Options are: Top to Bottom, Bottom to Top, Left to Right, and Right to Left. **Default:** Top to Bottom.
Orthogonal Fix Node Sizes	This setting affects link routing behavior for the hierarchical layout option only. If enabled, node sizes do not change when you use the link routing feature. If disabled, link routing changes node sizes as needed for clarity. **Default:** Disabled.
Run analysis when creating view	Runs the Project Analyzer when creating a Selected Entity Project diagram. **Default:** Enabled.
Run fast layout for large diagrams	Only for Selected Entity Project diagrams. If enabled, TIBCO BusinessEvents filters out certain properties before generating a large Selected Entity Project diagram to make it look simpler. **Default:** Disabled.
Show all properties in Concept Node	If enabled, a diagram shows all properties of a concept node, instead of showing only four default properties. **Default:** Disabled.
Show all properties in Event Node	If enabled, a diagram shows all properties of an event node, instead of showing only four default properties. **Default:** Disabled.
Show Catalog Functions	Sets whether to show catalog functions in a Sequence diagram. **Default:** Enabled.
Show Catalog Function Return Links	Sets whether to show return links of catalog functions in a Sequence diagram. **Default:** Disabled.

Option	Description
Show Expanded Names	Sets whether to show expanded names in a Sequence diagram. **Default:** Enabled.
Snap to grid	If you move nodes in a diagram and the grid is shown, the nodes snap to the grid lines if Snap to grid is enabled. **Default:** Disabled.
Undirected layout	Sets no orientation for the diagram. **Default:** Disabled.

Tester Preferences

The Tester preferences section lets you configure the tester behavior. See Testing and Debugging Projects.

Reference to Tester Preferences

Option	Description
Output Directory	The directory in which the result files are stored. **Default:**`/TestData/<Projectname>/<Processing Unit name>`
No of WM Objects	Maximum number of objects in working memory using a preference. **Default:** 50
Auto scroll Modified Result Tables	If enabled, you see scroll bars for the cells in Result Test Data after clicking Fit Content. These scroll bars move together for both the After and Before sections. **Default:** Enabled

When you test a project, the tester shows the values changed while running the engine and the instances created in Result Test Data. To set the text color preferences for background and foreground of the modified values, go to **Window > Preferences > TIBCO BusinessEvents > Tester > Appearance**.

Reference to Tester Appearance Preferences

Option	Description
Background of the modified value	The background color of the changed values. Click the color box, which opens a color palette. Select the new color from it. **Default:** Light Pink
Foreground of the modified value	The text color of the changed values. Click the color box, which opens a color palette. Select the new color from it. **Default:** Blue
Font for the modified value	The font in which the modified value appears. Click **Change** to change the font. **Default:** Tahoma Regular 11

Keywords and Other Reserved Words

Certain keywords and reserved words cannot be used in the rule language grammar.

Do not use the words listed in this section as identifiers, resource names, or folder names. Case sensitivity depends on context (as noted in documentation). The list also includes keywords and other reserved words for all add-on products.

`$lastmod`

`abs`

`abstract`

`accept`

`ACTION`

`AdvisoryEvent`

`after`

`alias`

`all`

`and`

`any`

`as`

`asc`

`assert`

`attribute`

`avg`

`backwardChain`

`between`

`body`

`boolean`

`break`

`by`

`byte`

```
case
catch
char
class
closure
Concept
CONDITION
const
ContainedConcept
continue
count
date
DateTime
day
days
dead
declare
default
define
delete
desc
distinct
do
double
during
emit
else
entity
enum
Event
```

```
except
exists
extends
extId
false
final
finally
fired
first
float
for
forwardChain
from
goto
group
having
hour
hours
id
if
implements
import
in
instanceof
int
interface
intersect
last
is_defined
is_undefined
```

```
key

last

latest

like

limit

lock

long

maintain

max

millisecond

milliseconds

min

minute

minutes

mod

moveto

native

new

next

not

null

object

offset

one

or

order

order

package

pattern

pending_count
```

```
policy
priority
private
process
protected
public
purge
QUERY
rank
repeat
requeue
return
rule
rulefunction
scope
second
seconds
select
short
SimpleEvent
sliding
starts
static
strictfp
String
such
sum
super
switch
synchronized
```

```
that

then

this

throw

throws

time

times

TimeEvent

timestamp

to

transient

true

try

ttl

tumbling

undefined

union

unique

using

validity

virtual

void

volatile

when

where

while

with

within
```

Reserved Global Variables

Certain global variables are reserved in TIBCO BusinessEvents. Do not use these keywords for creating global variables.

Reserved Global Variable	Default Value
Domain	domain
Deployment	*<Name of project>*
MessageEncoding	ISO8859-1

Keywords Reserved By Oracle

Certain keywords are reserved by Oracle and not necessarily by TIBCO BusinessEvents. Do not use these words when creating backing store scripts, else an error might occur.

`ACCESS`

`ADD`

`ALL`

`ALTER`

`AND`

`ANY`

`ARRAYLEN`

`AS`

`ASC`

`AUDIT`

`BETWEEN`

`BY`

`CHAR`

`CHECK`

`CLUSTER`

`COLUMN`

`COMMENT`

COMPRESS

CONNECT

CREATE

CURRENT

DATE

DECIMAL

DEFAULT

DELETE

DESC

DISTINCT

DROP

ELSE

EXCLUSIVE

EXISTS

FILE

FLOAT

FOR

FROM

GRANT

GROUP

HAVING

IDENTIFIED

IMMEDIATE

IN

INCREMENT

INDEX

INITIAL

INSERT

INTEGER

INTERSECT

```
INTO

IS

LEVEL

LIKE LOCK

LONG

MAXEXTENTS

MINUS

MODE

MODIFY

NOAUDIT

NOCOMPRESS

NOT

NOTFOUND

NOWAIT

NULL NUMBER

OF

OFFLINE

ON

ONLINE

OPTION

OR

ORDER

PCTFREE

PRIOR

PRIVILEGES

PUBLIC

RAW

RENAME

RESOURCE

REVOKE ROW
```

```
ROWID

ROWLABEL

ROWNUM

ROWS

START

SELECT

SESSION

SET

SHARE

SIZE

SMALLINT

SQLBUF

SUCCESSFUL

SYNONYM

SYSDATE

TABLE

THEN

TO

TRIGGER

UID

UNION UNIQUE

UPDATE

USER

VALIDATE

VALUES

VARCHAR

VARCHAR2

VIEW

WHENEVER

WHERE
```

```
WITH
```

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.