TIBCO BusinessEvents®

Developer’s Guide

Software Release 5.1
August 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

THIS 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.TXT FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This product is covered by U.S. Patent No. 7,472,101.

Contents

Figures

Tables

TIBCO BusinessEvents Resource Reference Tables

Preface

TIBCO BusinessEvents® allows you to abstract and correlate meaningful business information from the events and data flowing through your information systems, and take appropriate actions using business rules. By detecting patterns within the real-time flow of events, TIBCO BusinessEvents can help you to detect and understand unusual activities as well as recognize trends, problems, and opportunities. TIBCO BusinessEvents publishes this business-critical information in real time to your critical enterprise systems or dashboards. With TIBCO BusinessEvents you can predict the needs of your customers, make faster decisions, and take faster action.

TIBCO BusinessEvents
The Power to Predict®

Topics

Changes from the Previous Release of this Guide

This section itemizes the major changes from the previous release of this guide.

Documentation Restructuring

Chapters relating to setting up a cluster and a backing store have been moved from the *TIBCO BusinessEvents Administration* guide to *TIBCO BusinessEvents Developer's Guide* (this guide). Now the *TIBCO BusinessEvents Administration* guide contains only deploytime and runtime administration topics.

Studio-tools Changes

Topics relating to studio-tools command-line utilities used in TIBCO BusinessEvents Decision Manager are now shown only in *TIBCO BusinessEvents Decision Manager User's Guide*. Command-line syntax has also changed in this release. Documentation for tools in the studio-tools suite is provided in relevant sections of the guide.

Default for ParallelOps Changed

In release 5.0, parallel operations was enabled by default when cache aside database write strategy was used. Now parallel operations is enabled by default only when both cache aside *and* concurrent RTC features are used. See Table 53, CDD Agent Classes Tab Inference Agent and Query Agent Properties, on page 479.

Hawk Channel

The Hawk channel connects TIBCO BusinessEvents to a TIBCO Hawk domain and can be configured to use either TIBCO Rendezvous or TIBCO Enterprise Message Service transports. The Hawk channel enables TIBCO BusinessEvents to receive events from the Hawk monitor and transform them into events. Information about how to configure and work with the channel has been added in Chapter 7, HAWK Channel, on page 133.

Connection to TIBCO Hawk domain through a specific transport is configured using the shared resource Hawk connection. See Hawk Connection on page 214.

ActiveSpaces Channel

The ActiveSpaces channel connects TIBCO BusinessEvents to TIBCO ActiveSpaces metaspace. This enables TIBCO BusinessEvents to monitor the activities on the TIBCO ActiveSpaces metaspace and receive events from TIBCO ActiveSpaces and convert them into events in TIBCO BusinessEvents. Information about how to configure and work with the channel has been added in Chapter 8, ActiveSpaces Channel, on page 140.

Connection to a space in TIBCO ActiveSpaces is configured using the shared resource ActiveSpaces connection. See ActiveSpaces Connection on page 212.

HTTP Channel Changes

The BUILT-IN server option is no longer required and has been removed. Tomcat server version 7 is now supported, and it provides functionality previously provided by the BUILT-IN server type.

Backing Store Persistence Options

The backing store can now be configured with one of the following persistence options: None, Shared All, or Shared Nothing. Chapters related to backing store configuration have been updated with information about the new persistence options.

Load Balancer Configuration for Agents

Agents can be configured to work cooperatively as routers and receivers to ensure that related messages arriving from queue sources are handled by the same agent, so that related information is locally available. Chapter 25, Load Balancer Configuration, on page 425 explains the load balancing options and provides the steps to configure from the CDD file.

Generate JDBC Deployment Scripts From TIBCO BusinessEvents Studio

TIBCO BusinessEvents Studio provides an alternative approach to generate the project-schema-specific SQL scripts using the **JDBC Deployment** wizard. The wizard also allows you to generate SQL scripts to migrate existing backing store database. See Generating Scripts Using the JDBC Deployment Wizard on page 519 for details.

TIBCO BusinessEvents WebStudio and Rules Management Server

TIBCO BusinessEvents WebStudio is an online tool to define business rules and Rule Management Server (RMS) is its server component. Earlier RMS component was part of TIBCO BusinessEvents Decision Manager add-on only but now its part of the TIBCO BusinessEvents Standard. Following chapters are added in the guide for this release:

- Chapter 17, Rule Templates, page 269
- Chapter 31, Introduction to WebStudio, page 533
- Chapter 32, Configuring RMS and RMS Projects, page 541
- Chapter 33, Working With Projects in WebStudio, page 555
- Chapter 34, Working with Decision Tables in WebStudio, page 565
- Chapter 35, Business Rules, page 585
- Chapter 36, Working with the Approval Workflow in RMS, page 597
- Chapter 37, Deployable Files Generation, page 603

Miscellaneous

- The Berkeley DB persistence option has been removed from the product. All mention of Berkeley DB persistence has been removed from documentation.
- You can now use global variables in a project library and order project libraries to specify an override order for global variables that are listed more than one time. See Setting and Overriding Global Variables from a Project Library on page 14.
- Better information on post RTC to backing store tuning properties.
- Better information on backing store commit property.
- Domain model entries are case sensitive. For example, m and M are recognized as different entries. This is now noted.
- New information is added regarding JMS message acknowledgement. See Using CLIENT_ACKNOWLEDGE Mode with Websphere MQ and Cache-Aside on page 87.

- Terminology for certain threading options has been rationalized. Changes are shown in bold in the following table:

Old Name	New Name
• Shared Queue — Size — Workers	• Shared Queue — **Queue Size** — **Thread Count**
• Workers — Queue Size — Thread Count	• **Destination Queue** — Queue Size — Thread Count

- A new property is added to specify how a concept is stored in the TIBCO BusinessEvents DataGrid cache. See CDD Cluster Tab TIBCO BusinessEvents DataGrid Properties Reference on page 412.

TIBCO BusinessEvents Express

The TIBCO BusinessEvents Express edition provides more limited functionality than the TIBCO BusinessEvents Standard Edition. Some content in this documentation is not relevant to users of TIBCO BusinessEvents Express. Such content includes but is not limited to any chapters and major sections that contain a note indicating that the content does not apply to TIBCO BusinessEvents Express.

Minor references to unsupported features may not be called out in the text. Use the following general guidelines to understand what is and is not supported in these cases:

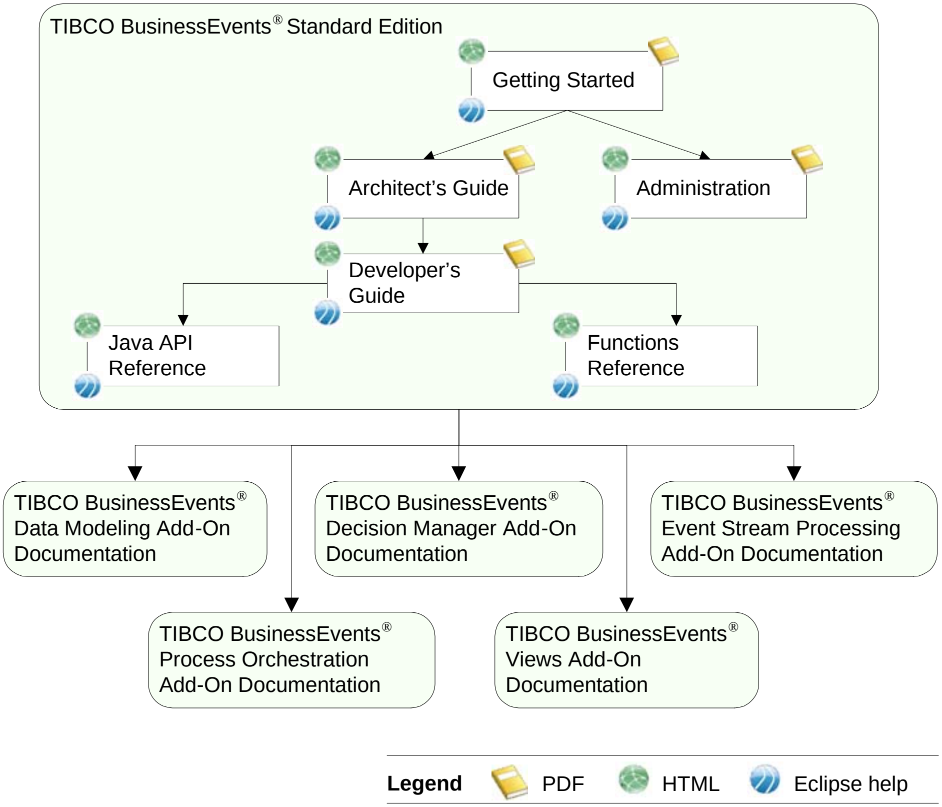
- Only In Memory object management (OM) is supported. Therefore all functionality that requires Cache OM, such as use of a backing store, is not available. Berkeley DB OM is also not supported with the TIBCO BusinessEvents Express edition.
- Only the TIBCO BusinessEvents Decision Manager add-on is supported with the TIBCO BusinessEvents Express edition in this release. Other add-on products are not supported.

Related Documentation

This section lists documentation resources you may find useful.

TIBCO BusinessEvents and Add-On Product Documentation

The following diagram shows the main documents in the TIBCO BusinessEvents documentation set, and the documentation sets for the optional add-on products.


TIBCO BusinessEvents® Standard Edition
Getting Started
Architect's Guide
Administration
Developer's Guide
Java API Reference
Functions Reference
TIBCO BusinessEvents® Data Modeling Add-On Documentation
TIBCO BusinessEvents® Decision Manager Add-On Documentation
TIBCO BusinessEvents® Event Stream Processing Add-On Documentation
TIBCO BusinessEvents® Process Orchestration Add-On Documentation
TIBCO BusinessEvents® Views Add-On Documentation
Legend
PDF
HTML
Eclipse help

Each set also contains an installation guide, release notes, and a readme file.

TIBCO BusinessEvents Documentation

TIBCO BusinessEvents Studio, the design-time UI, is supported on Windows and Linux. The documentation set for TIBCO BusinessEvents is as follows.

- *TIBCO BusinessEvents Installation*: Read this manual for instructions on site preparation, installation, upgrading from an earlier release, and project migration.
- *TIBCO BusinessEvents Getting Started*: After the product is installed, use this manual to learn the basics of TIBCO BusinessEvents: project design, cache OM, and backing store. This guide explains the main ideas so you gain understanding as well as practical knowledge.
- *TIBCO BusinessEvents Architect's Guide*: If you are architecting an application using TIBCO BusinessEvents, read this guide for overview and detailed technical information to guide your work.
- *TIBCO BusinessEvents Developer's Guide*: Use this guide when you implement a project design in TIBCO BusinessEvents Studio. It covers topics such as project-level tasks, resource-level tasks, debugging, and integration with TIBCO ActiveMatrix BusinessWorks. It also explains how to configure the CDD file for different object management options, and set up a backing store.
- *TIBCO BusinessEvents Administration*: This book explains how to configure, deploy, monitor, and manage a TIBCO BusinessEvents application and the data it generates using TIBCO BusinessEvents Monitoring and Management component, TIBCO Administrator, or at the command line. It includes authentication and authorization topics.
- Online References:
  - *TIBCO BusinessEvents Java API Reference*: This online reference is available from the HTML documentation interface. It provides the Javadoc-based documentation for the TIBCO BusinessEvents API.
  - *TIBCO BusinessEvents Functions Reference*: This reference is available from the HTML documentation interface. It provides a listing of all functions provided with TIBCO BusinessEvents, showing the same details as the tooltips available in TIBCO BusinessEvents Studio.
- *TIBCO BusinessEvents Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

TIBCO BusinessEvents Event Stream Processing

This TIBCO BusinessEvents add-on is available separately, and includes the TIBCO BusinessEvents Query Language features and the Pattern Matcher Service.

- *TIBCO BusinessEvents Event Stream Processing Installation*: Read this brief manual for installation instructions. A compatible version of TIBCO BusinessEvents must be installed before you install any add-on.

- *TIBCO BusinessEvents Event Stream Processing Query Developer's Guide*: This manual explains how to use the object query language to query various aspects of the running system For details on configuring and deploying query agents, see *TIBCO BusinessEvents Developer's Guide*.
- *TIBCO BusinessEvents Event Stream Processing Pattern Matcher Developer's Guide*: This manual explains how to use the pattern matcher language and engine to correlate event patterns in a running system.
- *TIBCO BusinessEvents Event Stream Processing Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

TIBCO BusinessEvents Decision Manager

This TIBCO BusinessEvents add-on is available separately. It incorporates the Decision Manager decision modeling business user interface (supported on Windows and Linux), and the Rules Management Server (supported on all platforms supported by TIBCO BusinessEvents).

- *TIBCO BusinessEvents Decision Manager Installation*: Read this brief manual for installation instructions. A compatible version of TIBCO BusinessEvents must be installed before you install any add-on.
- *TIBCO BusinessEvents Decision Manager User's Guide*: This manual explains how business users can use decision tables and other decision artifacts to create business rules. It also covers configuration and administration of Rules Management Server, which is used for authentication, authorization, and approval processes.
- *TIBCO BusinessEvents Decision Manager Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

TIBCO BusinessEvents Data Modeling

This TIBCO BusinessEvents add-on is available separately. It contains state models and database concept features.

- *TIBCO BusinessEvents Data Modeling Installation*: Read this brief manual for installation instructions. A compatible version of TIBCO BusinessEvents must be installed before you install any add-on.
- *TIBCO BusinessEvents Data Modeling Developer's Guide*: This manual explains data modeling add-on features for TIBCO BusinessEvents. The database concepts feature enables you to model TIBCO BusinessEvents concepts on Database tables. The state modeler feature enables you to create state machines.

- *TIBCO BusinessEvents Data Modeling Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

TIBCO BusinessEvents Process Orchestration

This TIBCO BusinessEvents add-on is available separately. It provides CEP functionality within the context of a BPM process, enabling you segregate different CEP rule sets within the flow of a BPM process.

- *TIBCO BusinessEvents Process Orchestration Installation*: Read this manual for instructions on site preparation and installation. A compatible version of TIBCO BusinessEvents must be installed before you install any add-on.
- *TIBCO BusinessEvents Process Orchestration Developer's Guide*: This guide explains how configure and deploy business processes whose actions are carried out using TIBCO BusinessEvents project resources.
- *TIBCO BusinessEvents Process Orchestration Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

TIBCO BusinessEvents Views

This TIBCO BusinessEvents add-on is available separately. It includes graphical dashboard components for run-time event monitoring.

- *TIBCO BusinessEvents Views Installation*: Read this manual for instructions on site preparation and installation. A compatible version of TIBCO BusinessEvents must be installed before you install any add-on.
- *TIBCO BusinessEvents Views Getting Started*: After the product is installed, use this manual to learn how to use TIBCO BusinessEvents Views to create and run a dashboard using a step-by-step tutorial.
- *TIBCO BusinessEvents Views Developer's Guide*: This guide explains how to use TIBCO BusinessEvents Views to create meaningful metrics that are presented to business users in real-time for proactive decision making.
- *TIBCO BusinessEvents Views User's Guide*: This book explains how to monitor metrics in TIBCO BusinessEvents TIBCO BusinessEvents Views and how to represent the business processes graphically.
- *TIBCO BusinessEvents Views Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Accessing TIBCO BusinessEvents Functions Reference Documentation

Reference documentation for functions, including those used in add-ons, is available in the HTML documentation interface for the TIBCO BusinessEvents documentation set, and as tooltips in TIBCO BusinessEvents Studio. To use the HTML-based functions reference from the file system do the following:

1. Browse to *BE_HOME*`/doc/standard/html` and click **`index.htm`**. The HTML documentation interface appears.
2. In the left panel, browse to Online References and in the right panel choose TIBCO BusinessEvents Functions Reference. The reference opens in a new tab.
3. Click the navigation links to browse to the functions as desired.

Other TIBCO Product Documentation

You may find it useful to refer to the documentation for the following TIBCO products:

- TIBCO ActiveSpaces®
- TIBCO Hawk®
- TIBCO Rendezvous®
- TIBCO Enterprise Message Service™
- TIBCO ActiveMatrix BusinessWorks™

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
ENV_NAME *TIBCO_HOME* *BE_HOME*	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • **Name** Identifies the installation environment. This name is referenced in documentation as *ENV_NAME*. On Microsoft Windows, the name is appended to the name of Windows services created by the installer and is a component of the path to the product shortcut in the Windows Start > All Programs menu. • **Path** The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*. TIBCO BusinessEvents installs into a directory within a *TIBCO_HOME*. This directory is referenced in documentation as *BE_HOME*. The default value of *BE_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\be\`5.1.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: MyCommand **`[enable`** | disable]

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrixBusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several *portlets*. Portlets are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
📝	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
💡	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
⚠	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical OR that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand param1 | param2 | param3`

Table 2 Syntax Typographical Conventions

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	`MyCommand {param1 param2} | {param3 param4}`
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	`MyCommand {param1 | param2} {param3 | param4}`
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

This section provides links to helpful TIBCO resources.

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts, a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Project Tasks

This chapter covers various project-level tasks for TIBCO BusinessEvents Studio, and presents a short introduction to project development.

For information about analyzing or viewing projects, see Chapter 41, Diagrams, on page 685.

Topics

Project Development Overview

TIBCO BusinessEvents Developer's Guide explains how to build a TIBCO BusinessEvents project using the TIBCO BusinessEvents core project resources such as channels, events, concepts, rules, and so on. This brief introductory chapter places this guide in the context of other documentation so you can understand its role in the overall documentation set.

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.

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's Guide*, gain a basic familiarity with the product by completing the tutorials in *TIBCO BusinessEvents Getting Started*.

TIBCO BusinessEvents Developer's Guide provides practical details on using TIBCO BusinessEvents Studio to build a project and configure its cluster deployment descriptor.

For in-depth explanations about designing a TIBCO BusinessEvents project and other topics, read *TIBCO BusinessEvents Architect's Guide*. References to relevant topics in that guide are provided in *TIBCO BusinessEvents Developer's 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 new project. See *TIBCO BusinessEvents Getting Started* for tutorials on building and populating a basic TIBCO BusinessEvents project.

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 Chapter 4, Migrating Projects from Earlier Versions in *TIBCO BusinessEvents Installation*.

To Create a Project

1. Start TIBCO BusinessEvents Studio. In Windows, click Start > All Programs > TIBCO > *YourEnvironment* > TIBCO BusinessEvents 5.1 > TIBCO BusinessEvents Studio.
2. If you are prompted, select or create the Eclipse workspace directory where your project files will be stored. (If you check the option to use this workspace as a default, you are not prompted again.)
3. You may be prompted to upload usage data collected by the Eclipse Usage Data Collector. Select an option as desired. (This is an Eclipse activity, and not part of using TIBCO BusinessEvents Studio.)
4. You see the TIBCO BusinessEvents Studio UI. From the File menu select **New > Project** > **TIBCO BusinessEvents > Studio Project** and click **Next**.
5. In the New Studio Project field, enter a project name.

   Names follow Java variable naming restrictions. Do not use any reserved words. See Identifier Naming Requirements on page 313.
6. Accept the default location or uncheck the Use default location checkbox and specify the desired location.
7. Click **Finish**.

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

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, you are prompted to make the file writable when a change is made.

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, you may find some of the below methods of finding elements more convenient than others.

Note that 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 wild cards.
- 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 `**`. You can also use the asterisk (`*`) as a wild card. 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. See Chapter 41, Diagrams, on page 685.

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

The Generate Schema utility lets you 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.

To Generate an XML Schema (XSD File)

1. Select the project whose schema 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 reach this utility from the option (right-click) menu anywhere in BusinessEvents Studio Explorer.
3. In the Schemas Folder field, browse to 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 displays. 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, 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**.

The XSD files for the selected resources (all concepts, all events, or both) are generated in 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

To Validate a Project, Project Folder, or Project Resource

In BusinessEvents Studio Explorer, do one of the following:

- Right click a project name, folder name, or a project resource name, and select **Validate Project**.
- Select a project name, folder name, or a project resource name and select **Project > Validate Project**.

A pop-up window displays the message, "Validation was successful" or summary information about any problems. Details about problems display in the Problems view.

To Fix Validation Errors

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 on page 261 for a related use of this feature.

Project Validation and Project Analysis

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 runtime. See Project Analyzer and Selected Entity Project Diagrams on page 696.

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, you must ensure that the engine can find all libraries (including dependencies on third-party libraries) and custom functions, if the project requires any.

If a project uses Rendezvous channels, for example, Java allows the user to set a property, `java.library.path`, to ensure that the environment path contains a reference to the directory containing Rendezvous 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 when you are configuring a run configuration or debug configuration.

Test Connection Feature To make the Test Connection feature work for JMS Connection and JDBC Shared Connection shared resources, see Enabling the Test Connection Feature on page 211.

To Add an External Library or Custom Function Path

This functionality is also available in the Debug Configurations > ClassPath tab, and the Run Configurations > ClassPath tab, for your convenience. See Chapter 40, Testing and Debugging Projects, on page 661.

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** and then select one of the following:
   - — The **Custom Functions** tab, to add custom functions paths
   - — The **Java Libraries** tab, to add third-party libraries paths
4. Click **Add Library** and browse to and select the JAR file.

5. If a library or custom function depends on additional third-party Java archives, do the following:

   a. Expand the library or custom function entry and double-click the **Native library location** entry.

   b. At the Native Library Folder Configuration dialog, select **External Folder** or **Workspace** as appropriate, and browse to and select the folder containing the native libraries.

   c. Click **OK** to dismiss this dialog.

6. Click **OK**. One of the following occurs:

   — If a custom function depends on third-party Java archives and you did not do step 5, a dialog displays when you click OK. It lists the name of the class or classes that could not be loaded. To resolve the problem, do step 5.

   — You see the message: "It is recommended that you rebuild your project after changing the build path. Would you like to rebuild now?" Click Yes to validate and fix any references already in code to the JARs you have added.

7. Save the resource.

Deploytime Action Required To make custom functions or external libraries available at runtime, do the following on all machines where TIBCO BusinessEvents is installed:

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

See Updating Classpath, Environment Variables, Path, and Copying JAR Files in *TIBCO BusinessEvents Administration*.

To Remove an External Library or Custom Functions JAR Path

Follow the instructions in To Add an External Library or Custom Function Path, but instead of clicking Add Library, select the JAR file you want to remove and click Remove Library. The entry for any native libraries it depends on is also removed.

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 (see Working With Project Libraries at the Command Line on page 30 for details).

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) 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.

Avoid Conflicting Elements 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 you export selected resources to an archive.

To Create a Project Library

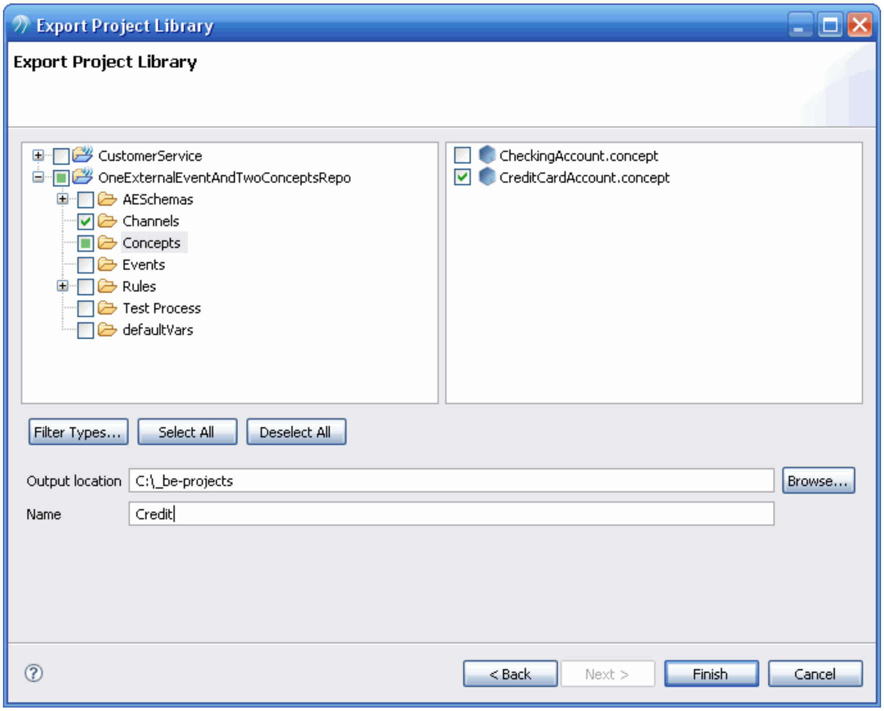
1. Optionally, select a project or any of its resources in TIBCO BusinessEvents Studio.
2. Do one of the following:

   — 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.)



Export Project Library

Export Project Library

- ☐ CustomerService
- ▣ OneExternalEventAndTwoConceptsRepo
  - ☐ AESchemas
  - ☑ Channels
  - ▣ Concepts
  - ☐ Events
  - ☐ Rules
  - ☐ Test Process
  - ☐ defaultVars

- ☐ CheckingAccount.concept
- ☑ CreditCardAccount.concept

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 button. 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**.

Remember to include the library you just exported, to avoid validation issues.

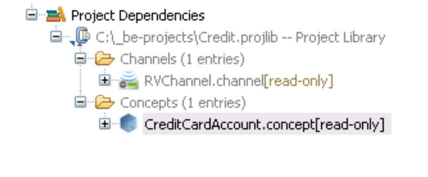
Adding (and Removing) Project Libraries in a Project

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.



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

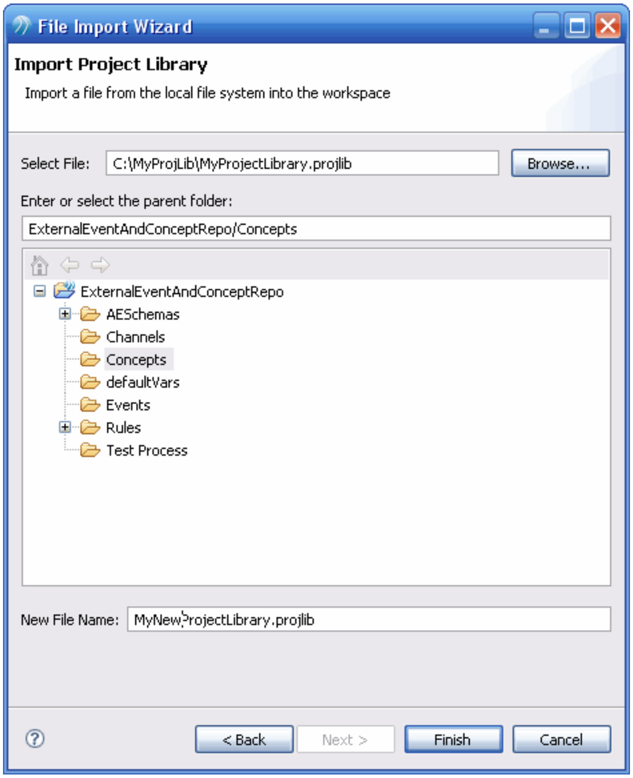
To Include a Project Library (Add it to the Build Path)

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.

To Import a Project Library

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 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. In the Import Project Library dialog Select 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.

To Remove a Project Library

To remove a project library, follow the instructions in To Include a Project Library (Add it to the Build Path), page 12, but instead of clicking Add Library, select the project library you want to remove and click Remove Library.

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.

For example, you could assign the value 7474 to the global variable `RvDaemon`. You can then use the variable in a Rendezvous Transport resource. At deploy time you can override the default value as needed.

- 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 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 on page 10).

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 override order is as follows:

- When multiple project libraries have the same-named global variable, the global variable from the library closest to the top of the project library list overrides any in lower libraries.
- When a local project global variable has the same name as a project library global variable, the local project global variable overrides the project library global variable.

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.

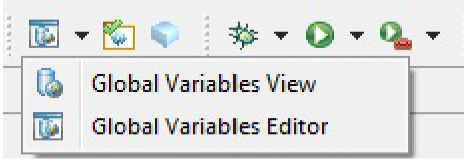
To see all global variables, including overridden ones, open the global variable editor and expand the Project Dependencies and project libraries. Global variables from project libraries are not editable.

Adding and Managing Global Variables

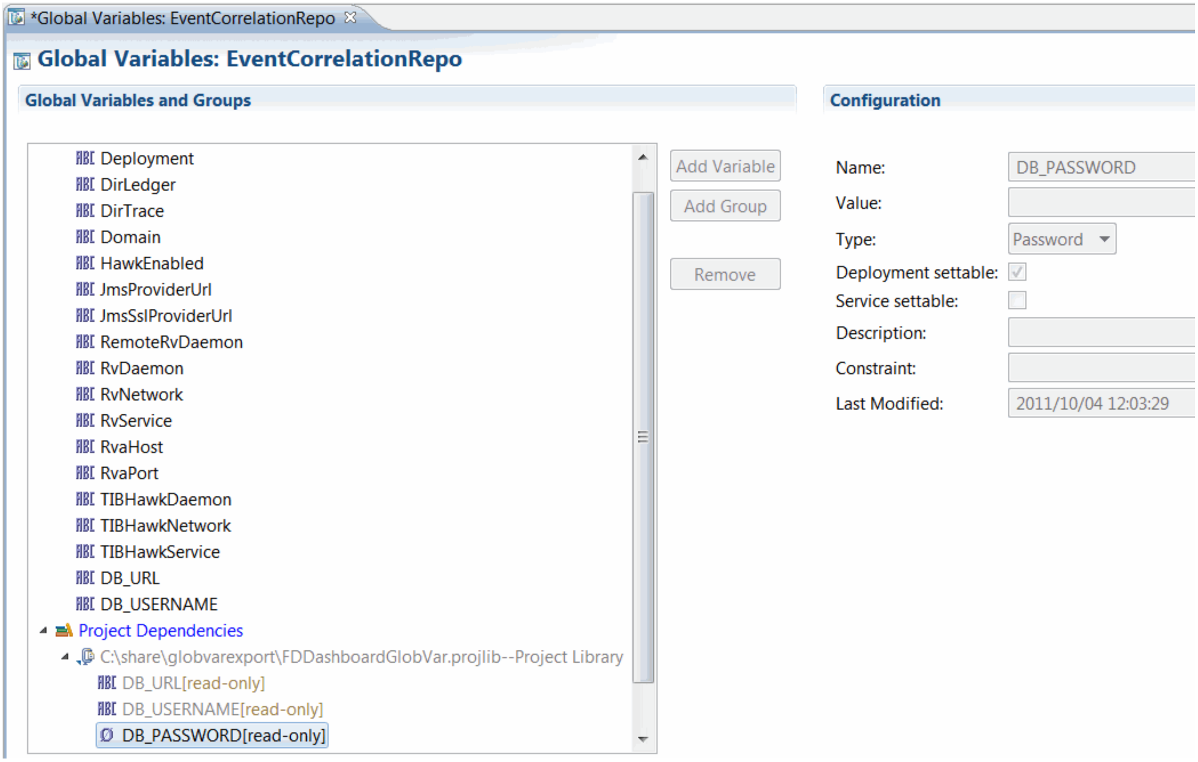
In a new TIBCO BusinessEvents Studio project no global variables are predefined. When you import a project from an earlier version of TIBCO BusinessEvents, however, you see predefined global variables (as well as any others defined in the project).

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 menus, select **Window > Show View > Other > TIBCO BusinessEvents > Global Variables**.



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

To Add and Manage Global Variables

1. Open the project in TIBCO BusinessEvents Studio.
2. Open the 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 Variable**s.

   You see the Global Variables Editor listing the variables available, if any.

3. Do any of the following. (See Table 3, Global Variable Reference, on page 16 for a guide to the fields):

   — 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**.

Using Groups 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%%`. (Because 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.)

You must add at least one variable to a group, or the group will not be 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.

Table 3 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.

Table 3 Global Variable Reference

Field	Description
Type	One of 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 BusinessEvents Monitoring and Management** You can only override global variables if both the Deployment settable and Service Settable checkboxes are checked. **For deployment with TIBCO Administrator** If checked, the variable is visible and settable when deploying using TIBCO Administrator. The values set at that time are saved in the project that TIBCO Administrator creates from the provided EAR file. If the checkbox is not checked, the variable is not visible in TIBCO Administrator. It is checked by default.
Service Settable	**For deployment with TIBCO BusinessEvents Monitoring and Management** You can only override global variables if both the Deployment settable and Service Settable checkboxes are checked. **For deployment with TIBCO Administrator** If both Deployment Settable and Service Settable are checked, the value of the global variable can be set differently for each deployable instance. **Note** Even if Service Settable is checked, 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. Not checked by default.
Description	A helpful description, as needed.

Table 3 Global Variable Reference

Field	Description
Constraint	Optional. For String and Integer types, allows you to 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 that constraints for Integers are currently not implemented in TIBCO Administrator.
Last Modified	Non-editable field records date and time this variable was last modified.

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*%%

As whown above, 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=*

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 will deploy:

- For starting at the command line:
  - In the design time CDD file. See Setting Global Variables in the CDD File (for Command Line Startup) on page 392.
  - 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 Starting a TIBCO BusinessEvents Engine at the Command Line in *TIBCO BusinessEvents Administration*.
- For deployment using TIBCO BusinessEvents Monitoring and Management component, set values in the CDD or MM Console. CDD values can be overridden in MM Console. See Setting Global Variables in MM in *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 checked (see Building an Enterprise Archive (EAR File) on page 22). Also see Deploying a Project in a TIBCO Administrator Domain in *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 the order of prededence, highest to lowest:

1. Command-line arguments at engine startup (highest priority).
2. Property files specified at command-line engine startup
3. MM Console or 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 or JMS Connection.

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 toTIBCO 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, you just 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/myGTCAFo
lder
```

To Store Trusted Certificates Outside of the Project

1. Create a directory where you want to store the trusted certificates. You must copy this directory to each machine where 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 Working with Global Variables on page 14 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_CA_STORE is used, and not the value you specify in the Trusted Certificates field.
5. Save the resource.

Building an Enterprise Archive (EAR File)

You can build an enterprise archive file using a TIBCO BusinessEvents Studio dialog, and also using a command-line utility (see Building an Enterprise Archive (EAR File) at the Command Line on page 26 for details).

For deployment using TIBCO Administrator, the configuration Name field value must match the project name The project does not deploy if they are different. The actual EAR file name, however, can differ from the configuration name.

Do not store the EAR file in a project folder Doing so includes the prior EAR file when you build the EAR file again, needlessly increasing the size.

Modifying Ignored Resources List Certain files (and folder names) are excluded from the EAR. To maintain the list of exclusions, in TIBCO BusinessEvents Studio, select Window > Preferences. > Shared Archive > Ignored Resources. See Setting Preferences on page 710 for more on this topic.

Options for Deploying the EAR

For deployment options see Chapter 5, Deploying and Managing Engines with MM and Chapter 8, Deploying With TIBCO Administrator in *TIBCO BusinessEvents Administration*. For testing purposes, you may find it useful to refer to the section Starting a TIBCO BusinessEvents Engine at the Command Line in *TIBCO BusinessEvents Administration*.

Building an EAR File in TIBCO BusinessEvents Studio

When testing project before development, you may want to build the EAR as explained here, and run the project at the command line.

1. In BusinessEvents Studio Explorer, highlight the project name, then from the top menus select **Project > Build Enterprise Archive**.

   If you see a message asking you to save all project resources, click Yes. It means an unsaved resource editor is open.

2. At the Build Enterprise Archive dialog, complete values according to guidelines provided in Table 4, Build Enterprise Archive Reference.

3. Click **Apply** to save the configuration details.

   (To revert to the version already saved, click Revert.)

4. Click **OK** to build the archive.

Table 4 Build Enterprise Archive Reference

Field	Description
Name	Name of this EAR configuration. (Not the EAR filename.) Default value is the project name. **Note** For deployment using TIBCO Administrator, the configuration Name field value must match the project name. The project does not deploy if they are different. The actual EAR file name, however, can differ from the configuration name.
Author	Person responsible for the EAR file. Default value is the currently logged-on user name.
Description	Optional description.
Archive Version	Increments on each build of the EAR. You can also manually enter a version identifier.
Generate Debug Info	Check this checkbox if you want to use the debugger. (See Chapter 40, Testing and Debugging Projects, on page 661.) Default setting is checked.
Include all service level global variables	Check to include service level global variables.
File Location	Browse to the directory in which you want to store the EAR and enter an EAR filename.
Delete Temporary Files	Before TIBCO BusinessEvents packages the EAR file, it generates the Java code in a temporary directory. After the files are packaged in the EAR file, then the temporary files and directory are deleted. You can choose to keep the generated Java files, for example to troubleshoot some problem with an EAR file. To do so, uncheck the Delete Temporary Files checkbox, and specify where to store the Java files in the Compilation Directory field. Default setting is checked, meaning that temporary files are not saved.
Compilation Directory	If you uncheck the Delete Temporary Files checkbox, specify the directory where you want to save the Java files generated during the process of building the EAR file.

Chapter 2 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's Guide* and *TIBCO BusinessEvents Installation*

The Studio Tools utility runs only on Microsoft Windows and Linux platforms as it has dependencies on Eclipse.

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

Topics

Building an Enterprise Archive (EAR File) at the Command Line

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

To Build an EAR File at the Command Line

1. Navigate to *BE_HOME*`/studio/bin/` and open a command prompt.
2. Execute a command with the following format (all on one line) at a command prompt:

```
studio-tools -core buildEar [-h] [-x] [-o outputEarFile>] -p studioProjectDir
```

For example:

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

Table 5, TIBCO BusinessEvents Studio Tools Options for Building an EAR File, provides detailed information about the options.

Table 5 TIBCO BusinessEvents Studio Tools Options for Building an EAR File

Option	Description
`-core buildEar`	Within the `core` category of operations, specifies the `buildear` operation for building EAR files.
`-h`	Optional. Displays help.
`-x`	Optional. Overwrites the specified output file if it exists.
`-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.

Importing a TIBCO BusinessEvents (4.x) Studio Project at the Command Line

This tool is equivalent to the following menu option in TIBCO BusinessEvents Studio: File > Import > TIBCO BusinessEvents > Existing TIBCO BusinessEvents 4.x Project. It is used to migrate 4.x projects to 5.x.

When you run this command-line utility, various migration actions are done, as explained in Migration from 4.x to 5.1 in *TIBCO BusinessEvents Installation*.

For migration of version 3.x projects, see Chapter 4, Migrating Projects from Earlier Versions in *TIBCO BusinessEvents Installation*.

Before you can use a 4.x project imported at the command-line in TIBCO BusinessEvents Studio you must do another procedure, explained in To Open a Project Imported at the Command Line in TIBCO BusinessEvents Studio on page 28.

To Import an Existing 4.x Project at the Command Line

1. Navigate to *BE_HOME*`/studio/bin/` and open a command prompt.
2. Execute a command with the following format (all on one line) at a command prompt:

```
studio-tools -op importExistingProject [-h] -p studioProjDir [-o targetProjDir] [-c
CDDprojectPath] [-u PUNameFromCDD]
```

For example:

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

If HTTP channel properties are migrated (from a specified CDD and processing unit to all HTTP channel resources' Advanced tab) you see a message like this:

```
Migrating HTTP properties of Processing Unit "PUName" from CDD "CDDprojectPath" to
HTTPChannel(s) present in the project
```

When the import has completed successfully, you see a message in the command window like the following:

```
The existing 4.0 TIBCO BusinessEvents project has been successfully imported to
c:\MyWorkspace\SomeProj.
```

Table 6, TIBCO BusinessEvents Studio Tools Options for Importing an Existing Project, provides detailed information about the options.

Table 6 TIBCO BusinessEvents Studio Tools Options for Importing an Existing Project

Option	Description
`-op importExistingProject`	Specifies the `importExistingProject` operation for importing a TIBCO BusinessEvents Studio project into the workspace.
`-h`	Optional. Displays help.
`-p`	Source project: absolute path to the project directory of the TIBCO BusinessEvents Studio project to be imported.
`-o`	Optional. Absolute path to the target project directory, where the project is imported to. If you specify the source project directory name as the last element in the path, it is used as the target project directory. If you specify a different directory as the last element in the path, the directory is created if it does not exist, and the source project directory is imported within the specified target directory. If you do not specify a target project directory, the original project contents are updated. If the project to be imported is a TIBCO BusinessEvents version 4 project, it is no longer compatible with version 4 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 but if specified `-c` 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.

To Open a Project Imported at the Command Line in TIBCO BusinessEvents Studio

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

1. Start TIBCO BusinessEvents Studio. In Windows, click Start > All Programs > TIBCO > *YourEnvironment* > TIBCO BusinessEvents 5.1 > 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 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 on page 10 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 do the following:

- 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.

To Work with Project Libraries at the Command Line

1. Navigate to *BE_HOME*`/studio/bin/` and open a command prompt.
2. Execute a command with the following format (all on one line) at a command prompt. Arguments for creating a project library are shown separately for clarity:

   To create a project library:

```
studio-tools -op buildLibrary [-h] -p studioProjDir [-f resources] [-x] -n projectLib
```

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

```
studio-tools -op buildLibrary [-h] -p studioProjDir [-a] [-r] -n projectLib
```

Examples for Creating a Project Library

Example 1 The following command creates a project library using the specified project's contents:

```
studio-tools -op 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 -op buildLibrary -x -p C:\workspace\MyProj -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 for Creating a Project Library, page 30, 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 -op buildLibrary -a -p C:\workspace\MyProj -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 -op buildLibrary -r -p C:\workspace\MyProj -n C:\test\myproj.projlib
```

Table 7 provides detailed information about the options.

Table 7 TIBCO BusinessEvents Studio Tools Options for Working with Project Libraries

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

Table 7 TIBCO BusinessEvents Studio Tools Options for Working with Project Libraries (Cont'd)

Option	Description
`-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.

Migrating Core Coherence Functions at the Command Line

The TIBCO BusinessEvents DataGrid component is the default cache provider for TIBCO BusinessEvents. Use of Oracle Coherence is also supported for those who want to use their own copy of this software.

A set of core Coherence functions was renamed in TIBCO BusinessEvents 5.0 (and additional internal changes were made) so that these functions can be used with either the Coherence or the TIBCO BusinessEvents DataGrid cache provider.

Migrating existing projects to use the new function names is one of the actions done automatically when you import a project from a previous release, as explained in Chapter 4, Migrating Projects from Earlier Versions in *TIBCO BusinessEvents Installation*.

Separate use of the `migrateCoherenceCalls` operation may be required in some cases. For example, you create a project in the current version with Coherence as the cache provider, and enable and use the Coherence-specific catalog functions, Then you switch to using TIBCO BusinessEvents DataGrid as the cache provider. You would then use the `migrateCoherenceCalls` operation to migrate those functions.

To Migrate Core Coherence Functions at the Command Line

1. Navigate to *BE_HOME*`/studio/bin/` and open a command prompt.
2. Execute a command with the following format (all on one line) at a command prompt:

```
studio-tools -core migrateCoherenceCalls –projectDir studioProjectDir
```

When the migration has completed successfully, you see a message in the command window:

```
All Coherence function calls have been migrated successfully.
```

Generating All Project Class Files at 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.

To Generate Class Files at the Command Line

1. Navigate to *BE_HOME*`/studio/bin/` and open a command prompt.
2. Execute a command with the following format (all on one line) at a command prompt:

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

For example:

```
studio-tools -core generateClass -p D:\Workspace\FraudDetection -o c:\temp -x true
-cp c:\tibco\be\5.1\lib\myjar.jar
```

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

Table 8, TIBCO BusinessEvents Studio Tools for Generating Class Files, provides detailed information about the options.

Table 8 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`	Absolute path to the TIBCO BusinessEvents Studio project directory.
`-n`	Optional. Specifies the name of the TIBCO BusinessEvents Studio project 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.

Table 8 TIBCO BusinessEvents Studio Tools for Generating Class Files (Cont'd)

Option	Description
`-o`	Specifies the output directory for generated classes. If you do not specify a directory, files are placed in a user temporary directory. For example, on Windows files might go in a directory like the following: `C:\Documents and Settings\User\Local Settings\Temp\BE_1322046141896`
`-x`	Optional. If `true`, overwrites any existing class file with the same name.
`-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 might add the following: `C:\customjars\custom.jar;C:\customjars\custom2.jar`

Chapter 3 Element Refactoring Operations

When you make changes to a project element, all references to that element must be updated accordingly. When such changes affect 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.

Topics

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.

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 make changes to 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 on page 42).

This section explains the refactoring (and related) procedures. See Automatic Refactoring Actions and Limitations on page 47 to understand what TIBCO BusinessEvents does for each type of refactoring operation.

Updating All References is Strongly Recommended

To ensure the integrity of the project, it is strongly recommended that you make all changes to all locations where a renamed or moved items is referenced. Only disable such changes if you are certain there are no references to the element, or there are unusual circumstances that justify such action.

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 Eclipse also allows you to compare your work with local history and to revert to an 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 (use File > Rename) and you can 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.

To Rename a Project Element

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, then select **File > Rename**.

   — Highlight the element, 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, uncheck the Update References checkbox (but see Updating All References is Strongly Recommended on page 38).

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 on page 42 for details.

   — The problems page displays if the rename 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. This is not generally recommended (see Updating All References is Strongly Recommended on page 38).

6. To complete the change, click **OK**.

To Move an Element to a Different Project Folder

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 on page 42 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. This is not generally recommended (see Updating All References is Strongly Recommended on page 38).
5. To complete the change, click **OK**.

To Delete an Element or Folder

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

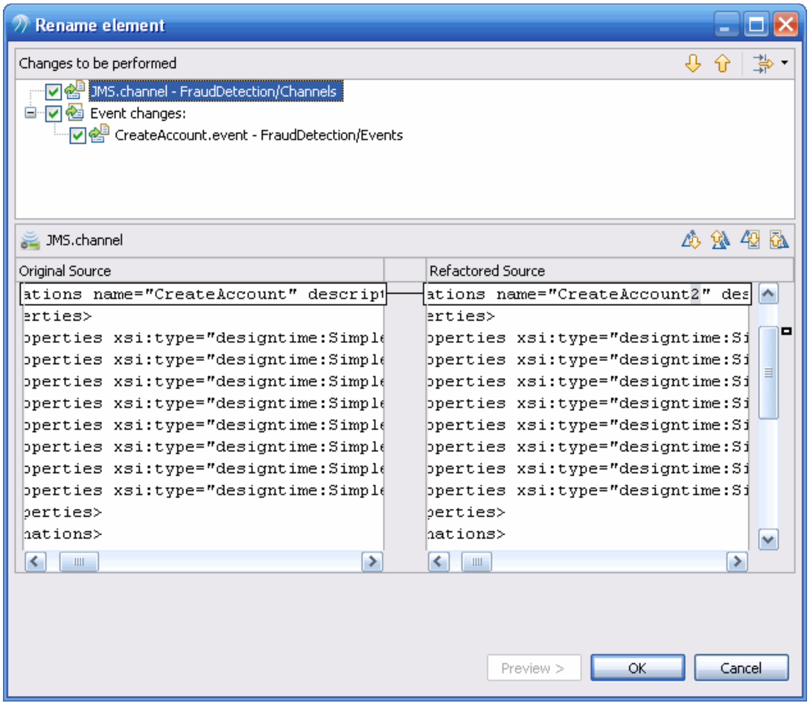
1. Do one of the following:
   - Right-click a project element or folder name in BusinessEvents Studio Explorer and select **Delete**.
   - Highlight the element, then select **Edit > Delete**.
   - Highlight the element, then press the **Delete** key.
2. At the Delete Resources page, click **OK** to 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 displays so you can examine the effect of this change. (For deletions, there is generally no information.) Click **OK** to delete.
   - The problems page displays 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).

To Delete a Project

1. Do one 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, check the Delete Project Contents on Disk check box. If you do not check 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 displays so you can examine the effect of this change. (For deletions, there is generally no information.) Click **OK** to delete.
   - The problems page displays 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.) This section explains how to use the preview page.



☑ JMS.channel - FraudDetection/Channels
☑ Event changes:
  ☑ CreateAccount.event - FraudDetection/Events

JMS.channel

Original Source

```
ations name="CreateAccount" descript
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
perties>
nations>
```

Refactored Source

```
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
perties>
nations>
```

Checks in the upper panel indicate elements that will 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, uncheck the checkboxes next to the elements as desired. For example, you may wish to replace or delete an element after you have completed the refactoring operation, so you do not need to apply the change to that element.

It is recommended that you 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
```

To Copy-Paste an Element

1. In BusinessEvents Studio Explorer, highlight the element (or folder) and do one of the following:

   — 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 do one of the following:

   — Right-click the folder and select **Paste**.

   — Press **`Ctrl+V`**.

   — Select **Edit > Paste**.

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

Migrating Core Coherence Functions

The TIBCO BusinessEvents DataGrid component is used as the default cache provider. Use of Oracle Coherence is also supported for those who want to use their own copy of this software.

A set of core functions has been renamed (and additional internal changes have been made) so that these functions can be used with either the Coherence or the TIBCO BusinessEvents DataGrid cache provider.

The renamed functions are in the Standard catalog `Cluster.DataGrid` category. You must use these functions going forward, and not the prior versions. See Table 9, Mapping of Coherence Functions to DataGrid Functions, on page 45 for details.

For all projects created prior to version 5.0.0, use the 4.x project import wizard, which migrates the core Coherence function calls automatically.

For occasional use cases where the function calls are migrated manually, you can use the manual refactoring operation. You can also use a command-line option as explained in Migrating Core Coherence Functions at the Command Line on page 33.

To Migrate Coherence Function Calls in TIBCO BusinessEvents Studio

See Table 9, Mapping of Coherence Functions to DataGrid Functions, on page 45 for details of the functions that are mapped using this feature.

1. Open the project in TIBCO BusinessEvents Studio (3.x projects must be migrated first). Migrate the function calls using one of the following methods:

   — Right-click the project name in Studio Explorer and select **Refactor > Migrate Coherence Function Calls**.

   The first page of the Migrate Coherence Function Calls wizard appears. It displays an informational list of the functions that the utility renames.

2. Click **OK** to make all necessary changes without previewing, or click **Preview** to preview the effects of the change.

3. If you click Preview, one of the following occurs:

   — The preview page displays so you can see the list of affected elements (which include rules, rule functions, expiry actions and state modeler transition, exit, and entry rules). You can scroll the list and view the original and refactored sources for each affected element. You can uncheck the checkboxes of elements you wish to remain unchanged (and later make changes manually as needed). Click **OK** to make the changes to all or selected elements.

— The problems page displays if there is a problem with the change. Click **Back** to fix the problem, or **Continue** to force the change (or **Cancel** to cancel the change).

— The preview page displays this message: The refactoring does not change any source code. Click **OK** and complete the migration anyway.

Complete the refactoring operation even if no project code is affected This refactoring operation makes additional changes to your projects so that your projects will work correctly with the new function names.

Mapping of Coherence Functions to TIBCO BusinessEvents DataGrid Functions

Table 9 Mapping of Coherence Functions to DataGrid Functions

4.x Coherence Category Function Name	DataGrid Category Function Name
C_CacheGetEntityById()	CacheGetEntityById()
C_CacheLoadConceptByExtId()	CacheLoadConceptByExtId()
C_CacheLoadConceptById()	CacheLoadConceptById()
C_CacheLoadConceptIndexedByExtId()	CacheLoadConceptIndexedByExtId()
C_CacheLoadConceptsByExtId()	CacheLoadConceptsByExtId()
C_CacheLoadEntity()	CacheLoadEntity()
C_CacheLoadEventByExtId()	CacheLoadEventByExtId()
C_CacheLoadEventById()	CacheLoadEventById()
C_CacheLoadParent()	CacheLoadParent()
C_CacheName()	CacheName()
C_CacheReevaluate()	CacheReevaluate()
C_ClassName()	ClassName()
C_EnableCacheUpdate()	EnableCacheUpdate()
C_Index()	Index()
C_Lock()	Lock()
C_TransactionProperties	TransactionProperties()

Table 9 Mapping of Coherence Functions to DataGrid Functions (Cont'd)

4.x Coherence Category Function Name	DataGrid Category Function Name
`C_UnLock()`	`UnLock()`

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 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.

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 Table 10. Projects can be complex; this list covers the main cases.

Table 10 Refactoring for Move and Rename Operations

Renaming this...	Updates references 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 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

Table 10 Refactoring for Move and Rename Operations (Cont'd)

Renaming this...	Updates references in these places...
Folder	All location that this folder is used in a path, for example 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 on page 47.

Refactoring for Delete Operations

Element deletion can affect project behavior. You must take care to 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 concepts that refer to it.
- Reference to a deleted state model are removed from the owning concept

Chapter 4 Channels and Destinations

This chapter explains how to configure channels and destinations, and includes information about Rendezvous and local channels. Additional information about JMS channel is provided in Chapter 5, JMS Channels, on page 79, HTTP channel in Chapter 6, HTTP and SOAP Channels, on page 93, Hawk channels in Chapter 7, HAWK Channel, on page 133, and ActiveSpaces channel in Chapter 8, ActiveSpaces Channel, on page 140.

Topics

Overview of Channels and Destinations

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

See Also

- Chapter 2, Channels and Events in *TIBCO BusinessEvents Architect's Guide* to understand more about the role of channels in a project, and how messages are acknowledged.
- Chapter 41, Diagrams, page 685 for information on using channel dependency diagrams.
- For TRA file updates required for Rendezvous and JMS channels that use TIBCO Enterprise Message Service, see the following section In *TIBCO BusinessEvents Administration*: For TIBCO Enterprise Message Service and TIBCO Rendezvous Channels ().

Types of Channels

You can choose from the following types of channels:

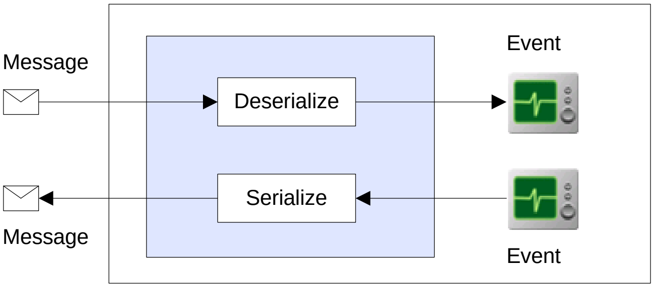
- **TIBCO Rendezvous channels** Connect TIBCO BusinessEvents to TIBCO Rendezvous sources and sinks. See Working with Rendezvous Channels on page 56.
- **Local channels** Connect co-located agents at runtime. See Working with Local Channels on page 59
- **JMS channels** Connect TIBCO BusinessEvents to TIBCO Enterprise Message Service provider sources and sinks. See Chapter 5, JMS Channels, on page 79.
- **HTTP channels, including SOAP support** An HTTP channel acts as an HTTP server at runtime. This enables TIBCO BusinessEvents to serve requests from clients, and to act as a client of other servers. Support for SOAP protocol is provided by a specialized event type and a set of functions for extracting information from SOAP request messages and constructing response messages. See Chapter 6, HTTP and SOAP Channels, on page 93.
- **Hawk channels** Connect TIBCO BusinessEvents to TIBCO Hawk domain. This enables TIBCO BusinessEvents to receive events from the Hawk monitor and transform them into events. A set of catalog functions are also provided which are used to control the Hawk microagents through Hawk APIs.See Chapter 7, HAWK Channel, on page 133 for details.

- **ActiveSpaces channels** Connect TIBCO BusinessEvents to TIBCO ActiveSpaces metaspace. This enables TIBCO BusinessEvents to monitor the activities on the TIBCO ActiveSpaces metaspace, receive events from TIBCO ActiveSpaces and convert them into events in TIBCO BusinessEvents. A set of catalog functions are provided to control the type of the TIBCO ActiveSpaces events. See Chapter 8, ActiveSpaces Channel, on page 140 for details.

Selecting a Serializer

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 also provides you the option of using action rule functions on the message instead of converting messages to event using deserializer.

Figure 1 Serializer and Deserializer Behavior



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

For JMS and Rendezvous messages, different serializers are available for different needs. For details see Selecting a JMS Serializer on page 81 and Working with Rendezvous Channels on page 56.

For HTTP and SOAP, specific serializers are used. See Mapping of HTTP Requests to Events, with the REST serializer on page 95 for HTTP channel configuration. See Configuring TIBCO BusinessEvents as a SOAP Server and Client on page 116 for more on SOAP.

For TIBCO ActiveSpaces tuples, the AS Serializer is used. See **Working with ActiveSpaces Channels on page 147** for ActiveSpaces channel configuration.

For TIBCO Hawk, specific serializers are used depending on the transport you choose for the channel. See

Deploy-time Configuration

In the Cluster Deployment Descriptor, you specify what destinations an agent listens to at runtime, and you configure the destinations with an event preprocessor (as needed) and threading options. See the section Configuring Collections of Rules, Rule Functions, and Destinations on page 470.

Mapping Incoming Messages to Non-default Events

Incoming messages can be mapped to a default event that you specify when you configure the destination. (See Default Destinations and Default Events in *TIBCO BusinessEvents Architect's Guide* for more details). You can also map incoming messages to specified event types.

Two 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`

These fields are added and filled automatically for events created using TIBCO BusinessEvents rules. You can also add these fields to messages coming in from other sources, if you have control of those messages.

See also:

Mapping of HTTP Request URI to Destination on page 96

Mapping of SOAP Request URI to Destination on page 117

Specifying Default and Non-Default Destinations on page 154

Working with Rendezvous Channels

This section explains the behavior of the provided Rendezvous serializer classes and other related information.

Note that the Rendezvous serializers use UTF8 encoding for XML payloads.

The type of the event from which the outbound message is serialized is added to the Rendezvous message header using `_nm_` and `_ns_` fields so that if the message is used in TIBCO BusinessEvents again, the correct event type is used to deserialize the message (ignoring the default event specified for the destination).

See Adding Channels and Destinations on page 60 for details on adding Rendezvous channels.

For TRA file updates required for Rendezvous and JMS channels that use TIBCO Enterprise Message Service, see the following section In *TIBCO BusinessEvents Administration*: For TIBCO Enterprise Message Service and TIBCO Rendezvous Channels ().

Rendezvous Message Header

For Rendezvous messages, the only header that TIBCO BusinessEvents interprets is `_sendsubject_` which is of type String. It is a read-only property. The event has to define this property to receive the value. The value is the actual Subject on which the message was sent.

Basic Serializer

The basic serializer, `TibRvMsgSerializer` serializer, is for efficient handling of events and messages that do not have payloads. It ignores payloads in messages and in events if any exist.

First level Rendezvous property values are used as values for matching event properties. Any additional (non-matching) Rendezvous properties are ignored.

Serializer For Use with Payloads

To include a payload in a Rendezvous message, ensure that the message has a `_payload_` field. To pass contents between the `Rendezvous` message `_payload_` field and an event payload, use the `TibRvMsgSerializerWithXMLPayload` serializer.

Deserializing from Rendezvous Message to Event

First level Rendezvous property values are used as values for matching event properties. Any additional (non-matching) Rendezvous properties are ignored.

The `_payload_` field contents are passed into the event payload. Supported `_payload_` field datatypes and Rendezvous wire format types are as follows:

Data Type	Wire Format Type
`String`	`TibrvMsg.STRING`
`TibrvXML`	`TibrvMsg.XML`
`byte[]`	`TibrvMsg.OPAQUE` `TibrvMsg.I8ARRAY`

If the event defines a payload, but the incoming Rendezvous message does not have a `_payload_` field, TIBCO BusinessEvents attempts to map the entire message as the event payload.

Serializing from Event to Rendezvous Message

Event properties are transformed to first level Rendezvous message properties.

The event payload is passed to the Rendezvous message `_payload_` field.

If the `_payload_` field is of an unsupported type, or is missing, or if the event has not been configured for a payload, the payload is ignored.

Avoiding REGISTRATION COLLISION RVCM Advisory Messages

This section applies to configuring RVCM when using a Rendezvous transport shared resource (see Rendezvous Transport on page 236).

`REGISTRATION.COLLISION` RVCM advisory messages can be thrown when when the same CM name (or CMQ name) is used on all engines. These name collisions can result in thrashing.

To prevent this issue, you must ensure that the CM names (and CMQ names) are different on all engines. To do so, add global variables to the CM Name or CMQ Name, and to the ledger file name if a ledger file is used for RVCM, to ensure the uniqueness of these names. Add one or more of the following variables, depending on need (as explained below):

```
%%EngineName%%
%%ChannelName%%
%%ChannelURI%%
```

The `%%EngineName%%` variable is generally required for all names. Note that you must start engines using unique names so that the value of each engine's `%%EngineName%%` variable be different at runtime.

In addition, if different channels use the same RVCM shared resource, you also need to add `%%ChannelName%%` or `%%ChannelURI%%`.

Use `%%ChannelURI%%` in cases where channels using the same RVCM shared resource have the same name but are in different folders.

You must define any of the above String type global variables you use. They are not predefined. However, TIBCO BusinessEvents provides the value at runtime, so you can use any string value or use an empty string as the value when you define the variables.

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's Guide*.

See Adding Channels and Destinations on page 60 for details on adding local channels.

Event Use Count

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 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.

Using Local Channel Destinations

Local channel destinations can use an event preprocessor.

Local channel destinations do not have 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:

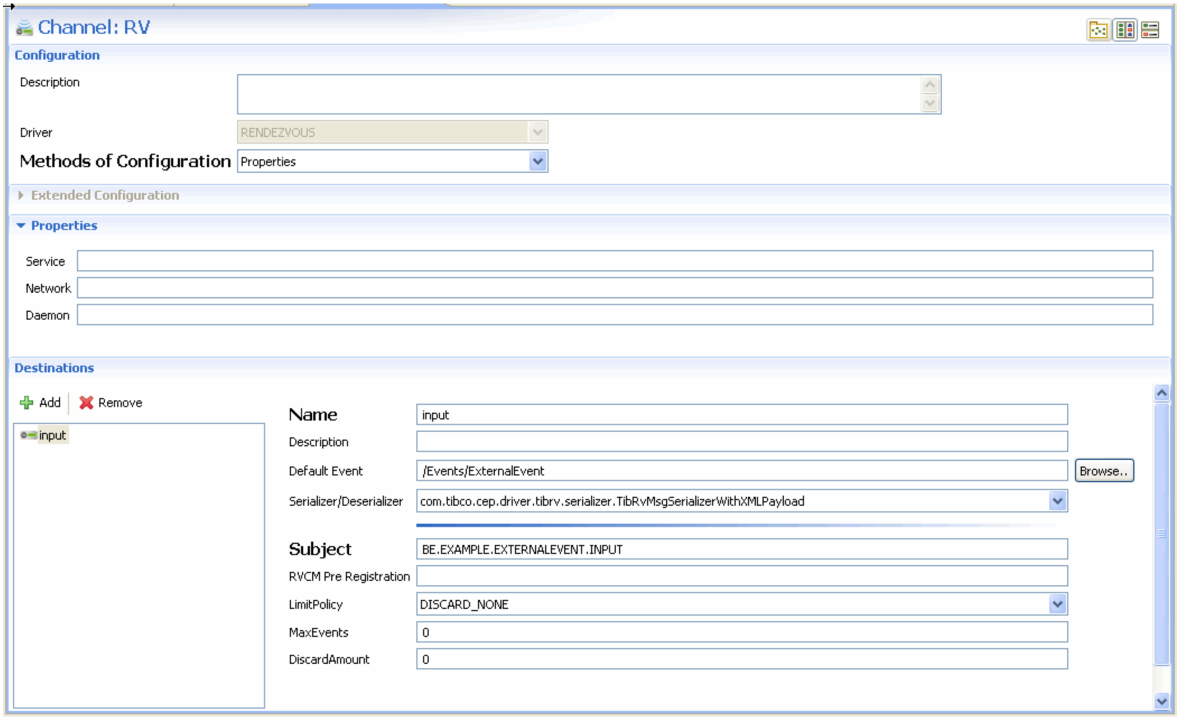
```
Event.routeTo(order, "/Channels/Local/toSmall", "");
```

The signature of this function is as follows:

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

Adding Channels and Destinations

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



Channel: RV

Configuration

Description:

Driver: RENDEZVOUS

Methods of Configuration: Properties

Extended Configuration

Properties

Service:

Network:

Daemon:

Destinations

Name: input

Description:

Default Event: /Events/ExternalEvent

Serializer/Deserializer: com.tibco.cep.driver.tibrv.serializer.TibRvMsgSerializerWithXMLPayload

Subject: BE.EXAMPLE.EXTERNALEVENT.INPUT

RVCM Pre Registration:

LimitPolicy: DISCARD_NONE

MaxEvents: 0

DiscardAmount: 0

To Add a Channel

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.

You cannot change the name in the editor. To change the name of any project element, right-click the element in Studio Explorer and select **Refactor** > **Rename**. See Chapter 3, Element Refactoring Operations, on page 37 for more details.

3. In the Driver type field, select the appropriate driver:
   - Rendezvous
   - Local
   - HTTP
   - JMS
   - Hawk-EMS
   - Hawk-RV
   - ActiveSpaces

   If you select Local, channel configuration is complete.
4. Click **Finish**. You see the Channel editor.

To Edit a Channel

1. Edit the 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 the appropriate driver:
   - Rendezvous
   - Local
   - HTTP
   - JMS
   - Hawk-EMS
   - Hawk-RV
   - ActiveSpaces
3. For all channel types (except Local and HTTP), from the Method of Configuration drop-down list, select one of the following:
   - **Resource** Select Resource if you have a shared resource in your project whose properties you want to reuse for this channel. For HTTP channels, Resource is preselected and cannot be changed.
   - **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 on page 67 for help in entering the correct values.

Resource names and directory names The path to the resource and the resource name cannot contain any of the keywords or other reserved words listed in Keywords and Other Reserved Words on page 318.

4. For HTTP channels only, in the serverType field of the Extended Configuration section, TOMCAT is preselected as the server type and cannot be changed.
5. 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.
6. If you are configuring an HTTP channel, click Advanced and configure tuning properties. See Configuring TIBCO BusinessEvents to Receive and Send HTTP Requests on page 98 for a more detailed procedure and HTTP Channel Advanced Configuration Settings on page 102 for details.
7. Continue to the section Adding a Destination to a Channel on page 62 and add destinations to the channel as needed.

Adding a Destination to a Channel

One channel can have multiple destinations. Each is shown in the Destinations section of the Channel editor.

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**.

Common Fields

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: Selecting a JMS Serializer on page 81
   - For Rendezvous destinations: Working with Rendezvous Channels on page 56
   - For HTTP and SOAP destinations: Task D, Add a Destination, on page 100, in the section Configuring TIBCO BusinessEvents to Receive and Send

HTTP Requests on page 98. (Destination configuration is the same for SOAP and HTTP destinations. See Manually Creating Resources to Work with SOAP Services on page 117 and Creating Resources Using the WSDL Import Utility on page 119 for more information about creating SOAP channels and destinations)

- For Hawk destinations: see
- For ActiveSpaces:

Various fields appear, depending on your selection.

Channel-Specific Fields

6. Complete the rest of the properties for the type of destination you are creating. See Destination Resource Reference on page 73 for details.
7. Save the project.

Communicating with Other Sources using TCP

In addition to Channels, TIBCO BusinessEvents can also communicate with other data sources using TCP. You can create a local TCP server and a TCP client so that TIBCO BusinessEvents can communicate with data sources not otherwise available through channels, using TCP.

TCP communication is available as a Communication Built-in Function in the Catalog Functions view. Using this set of functions you do the following to communicate with TCP servers:

- Create a local TCP server in a startup function using `TCP.createLocalServer()`
- Connect to a Remote TCP server as a client using `TCP.connectToRemoteServer()`
- Register the session listener using `TCP.registerSessionListener()`
- Start the local server using `TCP.startLocalServer()`
- Create callback rule functions and register them as callbacks to the TCP listeners. These callback rule functions create events that are sent to an appropriate destination.

  `TCP.readIntoPayload(SessionName)`

Example TCP Rule Function to Start a Local TCP Server

Here is a sample rule function to start a local TCP server:

```
void rulefunction RuleFunctions.InitTCPServers {
   attribute {
      validity = ACTION;
   }
   scope {

   }
   body {
      System.debugOut( "Initializing TCP servers" ) ;
      try {
         TCP.createLocalServer("MyTCPServer", "localhost",
            System.getGlobalVariableAsInt("NSN/SocketAdaptor/Port", 8055));
            TCP.registerSessionListener("MyTCPServer",
            "/RuleFunctions/RawCDRCallback");
         TCP.startLocalServer("MyTCPServer");
         } catch (Exception ex) {
            System.debugOut("Exception occurred while initializing TCP server: " +
               ex@message);
         }
```

```
        System.debugOut("TCP server initialization done");
      }
   }
```

Example TCP Rule Function to Connect to a Remote TCP Server

Here is a sample rule function to connect to a remote TCP server as a client:

```
Events.RemoteMsgResponseEvent rulefunction RuleFunctions.RemoteTCPSender {
   attribute {
      validity = ACTION;
   }
   scope {
      String host;
      int port;
      String message;
   }
   body {
      String tcpNickName = "TCP-" + host + "-" + port + "-"
         + uri + "-" + closure + "-" + System.nanoTime();
      Events.RemoteMsgRequestEvent requestEvent =
         Events.RemoteMsgRequestEvent.RemoteMsgRequestEvent(null, message);
      TCP.connectToRemoteServer(tcpNickName, host, port);
      TCP.write(tcpNickName, requestEvent);
      TCP.endWrite(tcpNickName);
      Events.RemoteMsgResponseEvent responseEvent =
         TCP.readIntoPayloadFully(tcpNickName,
         "/Events/RemoteMsgResponseEvent");
      TCP.disconnectFromRemoteServer(tcpNickName);
      return responseEvent;
   }
}
```

APIs for TCP Communication

The Catalog Functions view lists the following functions (APIs) for TCP communication:

- `connectToRemoteServer()`
- `createLocalServer()`
- `disconnectFromRemoteServer()`
- `disconnectLocalSession()`
- `endRead()`
- `endWrite()`

- `readIntoPayload()`
- `readIntoPayloadFully()`
- `registerSessionListener()`
- `startLocalServer()`
- `stopLocalServer()`
- `write()`
- Advanced
  - `getReaderInputStream()`
  - `readIntoByteArray()`

Documentation for functions is provided in the tooltips you can see when browsing the functions in TIBCO BusinessEvents Studio. You can also see this documentation in the *TIBCO BusinessEvents Functions Reference*, available in the HTML product documentation.

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 Overview of Channels and Destinations on page 52.

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

Local channels are in memory; information in a local channel could be lost if the TIBCO BusinessEvents engine fails.

Wizard and Configuration Section

The Wizard and the Configuration section have the following fields.

Field	Global Var?	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 Java variable naming restrictions. Do not use any reserved words. See Identifier Naming Requirements on page 313.
Description	No	Short description of the resource.

Field	Global Var?	Description
Driver	No	Select the driver for the type of channel you are configuring: • TIBCO Rendezvous (see Working with Rendezvous Channels on page 56) • Local (see Working with Local Channels on page 59) • JMS (see Chapter 5, JMS Channels, on page 79) • HTTP (See Chapter 6, HTTP and SOAP Channels, on page 93) • Hawk-EMS (see Chapter 7, HAWK Channel, on page 133) • Hawk-RV (see Chapter 7, HAWK Channel, on page 133) • ActiveSpaces (see Chapter 8, ActiveSpaces Channel, on page 140)
Method of Configuration	No	**Resource** Select Resource if you have a shared resource in your project whose properties you want to reuse for this channel. **Note** The path to the resource and the resource name cannot contain any of the words listed in Keywords and Other Reserved Words on page 318. **Properties** Select Properties to configure this channel resource using properties. See: Configuration for TIBCO Rendezvous Channels on page 69 Configuration for JMS Channels on page 69
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. As a convenience, you can open the selected resource by clicking the underlined label.

Field	Global Var?	Description
Configuration for TIBCO Rendezvous Channels		
Service	Yes	The name of the service or port number through which Rendezvous sends messages. In most cases you can leave this field empty, accepting the default value. For more information about the Rendezvous service parameter, see *TIBCO Rendezvous Concepts* or *TIBCO Rendezvous Administration*. Default is 7500 (defined in Global Variables).
Network	Yes	The network over which Rendezvous sends messages. In most cases you can leave this field empty. For more information about the network parameter, see *TIBCO Rendezvous Administration*. Default is an empty string (defined in Global Variables).
Daemon	Yes	The location of the Rendezvous daemon, which is usually expressed as a client socket number, for example, "6555." In most cases, you can leave this field empty, accepting the default value. For more information about the daemon parameter, see *TIBCO Rendezvous Concepts*. Default is `tcp:7500` (defined in Global Variables).
Configuration for JMS Channels		
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 assigned to the username, above, for the purpose of accessing the Enterprise Message Service server.

Field	Global Var?	Description
IsTransacted	Yes	Accepts `true` or `false`. Specify **`true`** if the session has transaction semantics. Specify **`false`** if it has non-transaction semantics. For more information about the `IsTransacted` property, see TIBCO Enterprise Message Service documentation.
ClientID	Yes	The unique client ID of the connection.
Configuration for ActiveSpaces Channels See Basic TIBCO ActiveSpaces Concepts and Terminology on page 141 for the fundamental concepts and terminology.		
Metaspace Name	No	The name of a particular metaspace instance in TIBCO ActiveSpaces that the channel must connect to. The metaspace must be created and initialized before the channel can use it at runtime.

Field	Global Var?	Description
Multicast Url	No	The mechanism used by the metaspace to discover the current metaspace members. Multicast discovery can use either PGM - Pragmatic General Multicast or RV - TIBCO Rendezvous 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 will default 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, and the values of those optional arguments should only be changed when necessary, and with care as inappropriate values could easily result in degraded performance of the product.

Field	Global Var?	Description
Multicast Url (continued...)		If using TIBCO Rendezvous, the multicast URL is expressed in the following format: `tibrv://`*service*`/`*network*`/`*daemon*, where • *service* specifies the TIBCO Rendezvous service number (UDP port) that will be used. If not specified, the value defaults to 7889. • *network* specifies the interface and discovery group address that will be used to send Rendezvous discovery packets. The format is: `interface; discovery_group_address` If not specified, the default interface and discovery group address will be used. (`tibrv://7889/;239.8.8.9/`). • *daemon* specifies where to find the Rendezvous daemon. If not specified, it will try to connect to a local daemon on port 7500.
Unicast Url	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 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).

Destination Resource Reference

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

The Destinations section of a channel has 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. See Identifier Naming Requirements on page 313.
Description	No	Short description of the resource.
Default Event	No	The event to be created from incoming messages unless otherwise specified. As a convenience, you can open the selected event resource by clicking the underlined label. Optional, but only if you always specify an event type in the incoming message. Not used for local channel.
Serializer/ Deserializer	No	Specify a serializer class to convert messages to simple events and simple events to messages. See the following sections for details: • Selecting a JMS Serializer on page 81 • Working with Rendezvous Channels on page 56 • Task D, Add a Destination, on page 100
Configuration for TIBCO Rendezvous Destinations See TIBCO Rendezvous documentation for more details on these settings.		
Subject	Yes	The TIBCO Rendezvous subject for incoming and outgoing messages.

Field	Global Var?	Description (Cont'd)
RVCM Pre Registration	Yes	For TIBCO Rendezvous certified message publishers, specify pre-registered listener names as a comma separated list.

LimitPolicy	Yes	How you want the Rendezvous listener to behave when it receives more messages than the MaxEvents limit. Choose one of:
		`Discard_None` (default)
		`Discard_First`
		`Discard_Last`
		`Discard_New`
		When MaxEvents or DiscardAmount are zero (unlimited), the LimitPolicy must be `Discard_None`.

MaxEvents	No	Maximum number of message events that the queue can hold. The default value, zero (0) means an unlimited number of events.
DiscardAmount	No	The number of events to discard when the queue exceeds its maximum event limit. The default value, zero (0) means events are never discarded.
Configuration for JMS Destinations See TIBCO Enterprise Message Service documentation for more detail on these settings.		
Queue	No	Specifies whether the destination is a queue or a topic. Check the checkbox if the destination is a queue. If the destination is a topic, do not check it.

Field	Global Var?	Description (Cont'd)
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 throws 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.
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 TIBCO Enterprise Message Service. In message headers this is represented by the code 22. You can also set a delivery mode in an event. See Using JMS Header Properties in Incoming and Outgoing Messages on page 89.
AckMode	No	The acknowledgement mode. See Setting the JMS Message Acknowledgement Mode on page 85 for a table explaining the various modes. You can also set an acknowledgement mode in a node's engine properties. The setting in the destination overrides the engine property setting. Default is `EXPLICIT_CLIENT_ACKNOWLEDGE`

Field	Global Var?	Description (Cont'd)
Priority	No	The message priority. Takes a numerical value between 0 and 9. Larger numbers represent higher priority. You can also set a priority in an event. See Using JMS Header Properties in Incoming and Outgoing Messages on page 89. Default is 4
TTL	No	The length of time that the message will live (in milliseconds) before expiration. If set to 0, the message does not expire. You can also set a TTL (`JMSExpiration`) in an event. See Using JMS Header Properties in Incoming and Outgoing Messages on page 89. Default is 0
Durable Subscriber Name	Yes	For destinations that are JMS Topics, if you provide a DurableSubscriber Name, the destination becomes a JMS 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. See Creating Unique JMS DurableSubscriber Name Properties on page 83 for details. Default is: `%%EngineName%%:%%SessionName%%:%%ChannelURI%%:%%DestinationName%%`
Configuration for Local Destinations Local destinations do not use serializers, deserializers, or default events.		
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.

Field	Global Var?	Description (Cont'd)
TimeOut	No	Time to wait when sending an event to this local destination: -1 Waits indefinitely 0 Does not wait >0 Waits for the number of milliseconds specified Default is -1.
Configuration for HTTP Destinations		
Is Page Flow	No	Enables Action Rule Function based approach. If checked, the system disables the Default Event and Serializer/Deserializer fields for input.
ContextPath	No	Context URI for the web application. The field is active for input if the Is Page Flow checkbox is checked.
Action Rule Function	No	The Action Rule function to be executed when an HTTP message arrives at the context URI for the web application. The field is active for input if the Is Page Flow checkbox is checked.

Chapter 5 JMS Channels

See Chapter 4, Channels and Destinations, on page 51 for basic information and procedures that apply to all types of channels. This chapter provides additional information about working with JMS channels.

Topics

Overview of JMS Channels

This chapter addresses tasks you may have to complete when configuring a JMS channel.

- To understand how to choose a serializer to handle the JMS message types that will be sent to the destination, see Selecting a JMS Serializer on page 81
- To understand how you can create unique DurableSubscriber Name property values for JMS Topic destinations, see Creating Unique JMS DurableSubscriber Name Properties on page 83.
- To learn how to change the JMS message acknowledgement type from the default type, see Setting the JMS Message Acknowledgement Mode on page 85.
- To understand when JMS messages are acknowledged, see When JMS Messages are Acknowledged on page 88
- To understand how TIBCO BusinessEvents handles certain JMS header properties, and how you can work with these header properties, see Using JMS Header Properties in Incoming and Outgoing Messages, page 89 and the related reference section, JMS Header Field Names on page 91.

See Also

- CDD Processing Units Tab JMS Server Connection Properties on page 499 for details about run-time properties used to reconnect to a JMS server.
- For TRA file updates required for Rendezvous and JMS channels that use TIBCO Enterprise Message Service, see the following section In *TIBCO BusinessEvents Administration*: For TIBCO Enterprise Message Service and TIBCO Rendezvous Channels ().

Selecting a JMS Serializer

This section explains the purpose of the provided JMS serializer classes. Choose the serializer that handles the JMS message types that will be sent to the destination you are configuring.

Table 11 Which Serializers to Use for JMS Message Types

Message Type	BytesMessage Serializer/ UtfBytesMessage Serializer	TextMessage Serializer	MessageWithNoBody
`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

For `MapMessage` messages, you create properties whose names match the message keys.

All serializers support reading and writing application header properties and JMS header properties. The difference between the serializers is in how they handle payloads.

`ByteMessageSerializer` decodes the body as a sequence of bytes and parses them to create an XML structure according to the payload definition in the event. `UtfBytesMessageSerializer` is similar to the `ByteMessageSerializer` except that it serializes the payload using `writeUTF()` instead of `writeBytes()`, and deserializes the payload using `readUTF()` instead of `readBytes()`. With `TextMessageSerializer`, the text from the message is decoded as an XML string. `MessageWithNoBody` does not serialize or deserialize the payload.

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. TIBCO recommends that you choose the serializer for a defined message type (MapMessage, TextMessage, and so on) that most closely matches the expected usage.

See Using JMS Header Properties in Incoming and Outgoing Messages on page 89.

BytesMessageSerializer

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() and deserializes the payload using readUTF().

For outgoing events, the serializer converts XML payloads to JMS BytesMessage message bodies.

TextMessageSerializer

For incoming messages, the TextMessageSerializer serializer converts JMS `TextMessage` messages to event payloads. The payloads are XML String type, and are human-readable. You can access them 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. For outgoing events, the serializer converts the payload to messages of type Message.

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.

- 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 check box 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 tab > *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%%:%%DestinationName%%
```

The first variable, `%%Deployment%%`, is a standard `TIBCO` global variable. The other three are only for use with the `DurableSubscriberName` property within TIBCO BusinessEvents. See Table 12, Variables for Use with DurableSubscriberName, for details.

Do not attempt to use `%%EngineName%%`, `%%SessionName%%`, `%%ChannelURI%%`, or `%%DestinationURI%%` in any area of TIBCO BusinessEvents software except the `DurableSubscriberName` property.

Table 12 Variables for Use with DurableSubscriberName

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 Chapter 29, Agent and Processing Unit Configuration, on page 467 for details.

Table 12 Variables for Use with DurableSubscriberName (Cont'd)

Variable	Description
`%%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%%`.

Setting the JMS Message Acknowledgement Mode

JMS channels support connection to TIBCO Enterprise Message Service destinations. The default acknowledgement mode is EXPLICIT.

You can set the acknowledgement mode either of these two ways:

- In the AckMode field of a destination resource.
- Using a configuration property.

The destination setting overrides the property setting. To set the acknowledgement mode using a configuration property, use one of the following, depending on whether you are using topics or queues:

`be.channel.tibjms.topic.ack.mode` *acknowledgement_mode_number*

`be.channel.tibjms.queue.ack.mode` *acknowledgement_mode_number*

Where *acknowledgement_mode_number* is one of the numbers that represent an acknowledgement mode, as shown in Table 13.

Note that in TIBCO Enterprise Message Service, mode names are slightly different. They are prefixed with `TIBEMS-`. See *TIBCO Enterprise Message Service User's Guide* for more details.

Table 13 JMS Message Acknowledgement Modes

No.	Mode	Description
1	`AUTO_ACKNOWLEDGE`	
		Specifies that the session is to automatically acknowledge consumer receipt of messages when message processing has finished.
2	`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 on page 87 for required configuration for Websphere MQ when cache-aside database write strategy is used.

Table 13 JMS Message Acknowledgement Modes (Cont'd)

No.	Mode	Description
3	`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.
23	`EXPLICIT_CLIENT_ACKNOWLEDGE` (TIBCO Proprietary)	
		TIBCO Enterprise Message Service extension to JMS acknowledge modes. This is the default. `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 may 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.
24	`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.

Table 13 JMS Message Acknowledgement Modes (Cont'd)

No.	Mode	Description
22	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. **Note** 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 do the following:

1. Define the destination using Caller's Thread in the Threading Model setting (in the CDD Collections tab or Agent Classes tab).

   See CDD Collections Tab Input Destination Settings Reference on page 472.

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 acknowledgement mode, time to live (TTL) setting, and object management (OM) type, as shown in the table below.

Table 14 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	[1]When retracted

[1] That is, 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, see Post-RTC and Epilog Handling and Tuning Options in *TIBCO BusinessEvents Architect's 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 configure the following three header properties:

- DeliveryMode (`JMSDeliveryMode`)
- Priority (`JMSPriority`)
- TTL (`JMSExpiration`)

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 Table 15, JMS Header Field Names, on page 91. 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 on page 90 for special handling of the `JMSReplyTo` header.

JMS header properties defined in events take precedence over properties defined in destinations.

You can add the JMS properties to the Base event in your project so that the properties are inherited by all other events.

See Table 15, JMS Header Field Names, on page 91 for details on all properties.

How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages

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 the `javax.jms.Destination` on the connected JMS server using this queue or topic name. If TIBCO BusinessEvents cannot find one, it creates a new `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

If you use the catalog function `Event.replyEvent(`*requestEvent*`,` *replyEvent*`)` during the RTC in which the *requestevent* is received, then the *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 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.

Table 15 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 specified by the sender. This property instructs the server concerning 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 delivery modes. You can also set a delivery mode for a destination. See Setting Certain Header Properties in Destinations on page 89.
`JMSExpiration`	Long	The length of time that the message will live (in milliseconds) 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 on page 89.
`JMSPriority`	Integer	The message priority, a numerical value between 0 and 9. Larger numbers represent higher priority. You can also set a priority for a destination. See Setting Certain Header Properties in Destinations on page 89.

Table 15 JMS Header Field Names (Cont'd)

Field name	Type	Description
`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	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. **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 which creates an exception, and the message is not sent. See How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages on page 90.
`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.

Chapter 6 HTTP and SOAP Channels

See Chapter 4, Channels and Destinations, on page 51 for basic information and procedures that apply to all types of channels. This chapter provides additional information about working with HTTP channels, including those configured to serve SOAP messages.

Topics

Overview of HTTP and 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 Support

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 on page 119.

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 Exporting Resources as a WSDL File on page 121.

See Configuring TIBCO BusinessEvents as a SOAP Server and Client on page 116 for more details.

Working with HTTP Requests

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

- **Event based approach** HTTP request is mapped to an event using deserializer.
- **Action Rule Function based approach** HTTP request parameters and data are retrieved using HTTP catalog functions and processed using rule function.

The **Is Page Flow** parameter of the destination identifies the approach followed by destination for processing HTTP requests. If the **Is Page Flow** parameter is set to true the HTTP request is processed using the Action Rule Function based approach otherwise Event based approach is used. This section explains the two different approaches for working with HTTP requests, and how to add an HTTP channel and destination.

Event Based Approach

The following paragraphs explain how an HTTP request is mapped to an event. It assumes basic familiarity with HTTP and with the procedure of setting up a channel. See Adding Channels and Destinations on page 60 if you need basic instructions.

Mapping of HTTP Requests to Events, 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 as defined in the corresponding event. If there is no payload defined for the event, the POST data is translated into a `ByteArray` payload, and is not accessible via the mapper. In other words, only XML payloads are visible via the mapper.

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 will always be used as payload in the event.

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.

Action Rule Function Based Approach

The following paragraphs explain how a HTTP request is processed in TIBCO BusinessEvents Studio without mapping to an event.

HTTP Request Processing

The TIBCO BusinessEvent 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.

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.

Sample Code for Action Rule Function

```
void rulefunction RuleFunctions.Callback {
   attribute {
      validity = ACTION;
   }
   scope {
      Object asyncContextObject;
      Concept sessionConcept;
   }
   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);
   }
}
```

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.

Configuring TIBCO BusinessEvents to Receive and Send HTTP Requests

To configure TIBCO BusinessEvents to receive and send HTTP requests, you need to perform the following tasks:

Task A Add 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 Check 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 Task B).

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.

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

Task B For SSL Only—Add an Identity Resource

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.

Task C Add 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 Task A.
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. See HTTP Channel Advanced Configuration Settings on page 102.

Task D Add a Destination

Add a destination to the channel in the usual way. (See Adding Channels and Destinations on page 60 for details.) The fields for destination differs based on different approaches.

To follow the action rule function based approach Check the Is Page Flow checkbox 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.

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`.

See Mapping of HTTP Request URI to Destination on page 96.

Task E Create Events as Needed and Set 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.

Task F Configure 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:

- Create a servlet request object for the HTTP request using the catalog functions to extract data from the HTTP request. See Chapter 18, Functions, on page 279 for more information on catalog functions.
- 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:

- 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:

- 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`.

Task G In the CDD, Configure the Processing Unit

In the Cluster Deployment Descriptor (CDD) Configure the processing unit for deployment as needed (see Configuring Processing Units (All OM Types) on page 492 for details). Note the following requirement.

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.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 1 of 7)

Property	Notes
Server Type	The only server type in this release is TOMCAT
Debug Request Info	Flag that identifies if the information is displayed for request debugging.
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.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 2 of 7)

Property	Notes
Max Processors	The maximum number of simultaneous requests that can be handled by 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 checked, the `TCP_NO_DELAY` option is not set on the server socket. If the checkbox is checked, the `TCP_NO_DELAY` option is set on the server socket. Using `TCP_NO_DELAY` improves performance under most circumstances. The default is checked.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 3 of 7)

Property	Notes
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. Allowable values are: **off** 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 will be 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 checked, the encoding specified in the `contentType` HTTP header is used to decode the request URI. If unchecked, the value in 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. See Sending Non-ASCII Content to Event Properties on page 96 for more details. For an example showing `contentType`, see Working with Outgoing SOAP Messages (Event Payloads) on page 126. The default is unchecked.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 4 of 7)

Property	Notes
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. See Sending Non-ASCII Content to Event Properties on page 96 for more details.
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.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 5 of 7)

Property	Notes
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 while 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.
Protocol	Specifies the HTTP protocol to use. Options are as follows: **`http`** HTTP protocol. **`https`** For secure communications (SSL). **`memory`** Memory protocol. **`ajp`** Apache JServ Protocol, a binary protocol. Default is `http`.
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.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 6 of 7)

Property	Notes
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`.
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.
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.

Table 16 HTTP Channel Advanced Configuration Settings (Sheet 7 of 7)

Property	Notes
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 suites (SSL protocols) used by the server. Add each suite name on a separate line.
	The following cipher suites are supported:
	`SSL_RSA_WITH_RC4_128_MD5`
	`SSL_RSA_WITH_RC4_128_SHA`
	`TLS_RSA_WITH_AES_128_CBC_SHA`
	`TLS_DHE_RSA_WITH_AES_128_CBC_SHA`
	`TLS_DHE_DSS_WITH_AES_128_CBC_SHA`
	`SSL_RSA_WITH_3DES_EDE_CBC_SHA`
	`SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA`
	`SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
	`SSL_RSA_WITH_DES_CBC_SHA`
	`SSL_DHE_RSA_WITH_DES_CBC_SHA`
	`SSL_DHE_DSS_WITH_DES_CBC_SHA`
	`SSL_RSA_EXPORT_WITH_RC4_40_MD5`
	`SSL_RSA_EXPORT_WITH_DES40_CBC_SHA`
	`SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`
	`SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
	Leave blank to use the default cipher suites for your SSL provider.

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
```

Using HTTP Functions to Configure HTTP Request Messages

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 located in the HTTP section of the Standard function catalog.

See HTTP Channel Advanced Configuration Settings on page 102 for details about setting up TIBCO BusinessEvents as a secure server.

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).

It is not recommended for production to use self-signed certificates.

If you use trusted certificates, see Loading Trusted Certificates on page 111.

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 Task A, Add an HTTP Connection, on page 98, 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 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 construct 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)
```

Description This function sends an event as a response to the request.

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.

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 this function.

Signature

`String sendAsynchronousRequest(String` *url*`, SimpleEvent` *requestEvent*`, String` *correlationId*`, String` *callbackRuleFunctionURL*`, String` *methodType*`)`

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.

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
`correlationID`	`String`	An optional ID to correlate the request and the response. If not specified, the ID is generated by the server.
`callbackRuleFunctionURL`	`String`	The fully-qualified path of a rule function. This rule function is called when the response is received. The response event would contain the correlation ID. The rule function must have correlation ID, RequestEvent, and ResponseEvent as parameters.
`methodType`	`String`	The HTTP method type. Valid values are: `GET` or `POST`.

Returns

Type	Description
`String`	A correlation ID

Sending a Secure Asynchronous Request

To send asynchronous requests using SSL, use this function.

Signature

`String sendSecureAsynchronousRequest(String` *url*`, SimpleEvent` *requestEvent*`, String` *correlationID*`, String` *callbackRuleFunctionURL*`, String` *methodType*`, Object` *clientIdKeystore*`, String` *clientIdPassword*`, Object` *trustedCertsKeystore*`, String` *trustedCertsPassword* `boolean` *verifyHostName*`)`

Description

This function is the same as the `sendAsynchronousRequest()` function, with the addition of the SSL-related parameters, shown below.

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.

Name	Type	Description
`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.

Returns

Type	Description
`String`	A correlation ID

Sending a Secure Synchronous Request

To send synchronous requests using SSL, use this function.

Signature

`Event sendSecureRequest(String` *url*`, SimpleEvent` *requestEvent*`, String` *responseEventURI*`, Object` *clientIdKeystore*`, String` *clientIdPassword*`, Object` *trustedCertsKeystore*`, String` *trustedCertsPassword*`, boolean` *verifyHostName*`,long` *timeoutMillis*`)`

Description

This function is the same as the `sendSecureAsynchronousRequest()` function, except that the requests are synchronous. The parameters are shown below.

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.
`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.

Name	Type	Description
`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.

Returns

Type	Description
`Event`	An event as a response to the request

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 in order to work with this feature. See http://www.w3.org/TR/soap/ for details.

How Clients View the WSDL 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 (see HTTP Channel Advanced Configuration Settings on page 102 for details about the document root setting). You must place the WSDL file in that location.

Overview of SOAP Related Resources

The following project resources must be configured so that TIBCO BusinessEvents can send and receive SOAP messages.

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.

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

See Parsing and Building SOAP Messages on page 123 for more details.

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 `SOAPAcion` 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 on page 119 explains how to use the WSDL import feature to create the resources.

Also see Understanding the WSDL to Project Resource Mapping on page 128 for details about how the WSDL elements correspond to project resources.

Task A Configure the HTTP Channel and SOAP Destination

Configure an HTTP channel, an HTTP connection resource for the channel. See Working with HTTP Requests on page 95 for details. 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 Task B, set that `SOAPEvent` as the default event for the destination.

Task B Add a SOAPEvent (and Other Ontology as Needed)

Configure the SOAPevents that will receive 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:

1. Add a simple event in the usual way. See Adding a Simple Event on page 155.
2. In the Inherits From field, select **SOAPEvent**.
3. Set the default destination to the destination you created in Task A.

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.

Task C 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. See Working with HTTP Requests on page 95. 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 Task B as input and return arguments of the preprocessor. See Parsing and Building SOAP Messages on page 123 for details.

Task D Configure an Event Preprocessor

Open the project CDD for editing, and configure an event preprocessor for the destination you configured in Task A. Use the rule function you configured in Task C.

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.

- 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.

See Understanding the WSDL to Project Resource Mapping on page 128 to understand which sections of the WSDL are used to create each resource.

To Import a WSDL File

First you must import the WSDL file into the project. Then you can use the WSDL import utility.

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. Check 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.

Re-importing 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:

Re-importing a WSDL file fails if the project artifacts are already present.

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 menus 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 on the **Mapper Check and Repair** button and check only the mappings in red.
7. Click **Ok** to fix the mappings.

8. If there are new or changed elements, you may have to manually map the new or changed elements.

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.

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/

Table 17 shows how related project resources are exported as a WSDL file.

Table 17 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 Task B.	Creates a message. The contents of Envelope > Header > Body become message parts.

Table 17 Exporting Project Artifacts as WSDL (Cont'd)

Project Folders and Resources	WSDL Artifact after Exporting
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.
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.

To Export a Rule or Rule Function as a WSDL

1. Right-click the project, and click **Export...**
2. In the Export dialog, select **WSDL** under TIBCO BusinessEvents, 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 (get) 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>
```

To Parse (and Optionally Remove) 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 will be 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()
```

To Parse 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.

The `getSOAPBodyParts() function` allows you to 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/Schema.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>
```

To Parse 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.

To Parse 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.Rowling</ns0:A
uthor></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, a ActiveMatrix BusinessWorks `SOAPRequestReply` activity.

For WSDL import to be successful, all names in the WSDL must conform to TIBCO BusinessEvents folder and entity naming requirements. See Identifier Naming Requirements on page 313.

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 Table 18, Imported WSDL Project Artifacts, on page 129 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 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.

Table 18 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>`.

Table 18 Imported WSDL Project Artifacts (Cont'd)

Project Folders and Resources	Project Resource Type	WSDL Source
`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	
`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>`

Table 18 Imported WSDL Project Artifacts (Cont'd)

Project Folders and Resources	Project Resource Type	WSDL Source
`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 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>`
`Service/`	TIBCO BusinessEvents folder	
`getBook`	Channel See Channel Folders on page 131.	`<soap:address location>` (from the last part of the location URL)
`Service_getBook`	Destination See Destination Names on page 132	`<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.

Chapter 7 HAWK Channel

See Chapter 4, Channels and Destinations, on page 51 for basic information and procedures that apply to all types of channels. This chapter provides additional information about working with the Hawk channel.

Topics

Working with the Hawk Channel

The Hawk channel allows TIBCO BusinessEvents to receive events from the Hawk monitor and transform them to events in TIBCO BusinessEvents. Create a channel with driver type Hawk as described in Adding Channels and Destinations on page 60.

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

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.RV_HOME=<absolute_path_where_TIBCO Rendezvous_is_installed>
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;%TRA_HOME%/lib;%EMS_HOME%/
lib
```

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

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 - used to listen on the events for any alerts being posted.
- RuleBaseListMonitor - used to listen on the events for changes in the rulebase. For example, if some rulebases are added, updated, or removed.
- Agent Monitor - 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 - used to listen on the events for any microagents added or removed.
- ErrorMonitor - used to listen on the event for errors.
- WarningMonitor - used to listen on the event for warnings.
- Microagent Method Subscription - used to subscribe to a microagent method at specified time intervals.

Runtime Configuration for the Hawk Channel

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

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

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 will be monitored. Click **Next**.

   If you selected MicroAgent Method Subscription, proceed to step 4. Else go to step 7.

If you select MicroAgent Subscription Method

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 displays. Select a microagent from the list of available Hawk microagents and click **Next**.
6. The Select Hawk MicroAgent Method dialog displays. 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.

Create Event for Hawk Catalog Function Using Wizard

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 displays. Select a microagent from the list of available Hawk microagents and click **Next**.
5. The Select Hawk MicroAgent Method dialog displays. 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 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
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.

Field Name	Description
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.

Chapter 8 ActiveSpaces Channel

See Chapter 4, Channels and Destinations, on page 51 for basic information and procedures that apply to all types of channels. This chapter provides additional information about working with the ActiveSpaces channel.

Topics

Basic TIBCO ActiveSpaces Concepts and Terminology

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

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 ActiveSpaces. An application or member first joins a metaspace, and through it, gets access to other objects and functionality.

A metaspace is a virtual entity: it 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 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.

ActiveSpaces handles changes in this set of processes automatically: processes may 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 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 ActiveSpaces. In effect, the seeding applications have a portion of the Space embedded in their process.

ActiveSpaces distributes the data stored in the space evenly between all of the processes that have joined the space as seeders. 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.

ActiveSpacesSerializer

The default serializer used to deserialize ActiveSpaces tuple to a simple event in TIBCO BusinessEvents and to serialize an event in TIBCO BusinessEvents to an ActiveSpaces tuple. You must configure the serializer (`com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer`) for every destination created on the 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: it may be that the best solution is to join some spaces as a seeder and others as a leech.

Consumption mode

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

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 or Take.

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) which 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

Table 19 shows the operators that are supported in the ActiveSpaces filters:

Table 19 Operators for ActiveSpaces Filters

Operator	Meaning
>, >=	greater than
NOT or ! or <>	not
*	multiply
=	equal
!=	not equal
ABS	absolute value
MOD	modulo
NOTBETWEEN	not between
BETWEEN	between

Table 19 Operators for ActiveSpaces Filters

Operator	Meaning
||	string concatenation
NOTLIKE	not like (regex)
LIKE	like (regex)
<, <=	less than
+	addition
OR	or
-	subtraction
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

Table 20 shows the formats for values used in filters.

Table 20 Formats for Filter Values

octal value	\oXXX
hexadecimal value	\xXXX

Table 20 Formats for Filter Values

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 ActiveSpaces Channels

Create a channel of type ActiveSpaces as described in Adding Channels and Destinations on page 60. In order to use the channel to monitor a space in ActiveSpaces, you need to add and configure one or more destinations for the channel.

Creating a Destination and Event Using the Wizard

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

The wizard requires the metaspace and spaces in TIBCO 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.

1. In BusinessEvents Studio Explorer, select the ActiveSpaces channel for which you need to create a destination and event. From the menu, select File > New > Other > TIBCO Channel Wizards > **ActiveSpaces Destination and Event** and click **Next**.
2. The Select Space dialog lists the available spaces for every ActiveSpaces channel in the project.

   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.

   Select a space and click **Next**.
3. On the New SimpleEvent and Destination dialog, enter the values to configure the simple event and destination for the selected channel and space.

Table 21: Configuring an 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.

Table 21: Configuring an ActiveSpaces Destination and Event

Field Name	Description
Destination	
Name	Name of the destination to be created. The default name on the screen is of the format <*spaceName*>Dest.
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 ActiveSpaces to simple events in TIBCO BusinessEvents. The default value is com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer. See ActiveSpacesSerializer on page 143 for more information.
Space Name	Name of the space which the destination connects to. This space is selected in the Select space dialog. See Space on page 141 for more information about spaces in TIBCO ActiveSpaces.
Distribution Role	The level of participation of the space member: seeder or leech. See Distribution role on page 143 for more information.
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 on page 144 for more information.
Consumption Mode	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. • Entry Browser - can listen and retrive tuples from the space using the Get or Take methods. See Consumption mode on page 143 for more information.

Table 21: Configuring an ActiveSpaces Destination and Event

Field Name	Description
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* or *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.
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. 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.

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

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

Adding a Destination for the ActiveSpaces Channel

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

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

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**.

Common Fields

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 on page 150 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 Table 21:, Configuring an ActiveSpaces Destination and Event, on page 147.

Assigning a Default Event to the Destination

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

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

Chapter 9 Simple 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.

Topics

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, it refers to a simple event.

See also

- Effect of Cache Only Cache Mode on page 243
- *TIBCO BusinessEvents Getting Started* provides a practical introduction to events, including use of default events and default destinations.
- Chapter 2, Channels and Events in *TIBCO BusinessEvents Architect's Guide* for a more detailed overview and information about the following topics:
  - Message Acknowledgement
  - Default Destinations and Default Events
  - Simple Events — Time to Live and Expiry Actions
- Working with Concept and Event Properties on page 321.
- Chapter 41, Diagrams, on page 685 for information on using event dependency and sequence diagrams, and event model views.

Using Inheritance

Use of inheritance can simplify event configuration. A child event inherits the following:

- 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.

- A parent event cannot have a payload.
- 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

This section explains various aspects of using events in rules.

See also Using Scheduler Functions (Requires Cache OM) on page 175.

Explicitly Assert Events Created 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 other hand, are automatically asserted.

Specifying Default and Non-Default Destinations

TIBCO 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.
- `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, for example, the network field in a Rendezvous channel.

Scheduling Simple Events

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.

Each method is appropriate in different situations. See Chapter 10, Time Events and Scheduler Functions, on page 165 for details.

Adding a Simple Event

This section provides summary steps for adding a simple event. See Simple Event Reference on page 157 for details on how to complete the values. Also see Overview of Simple Events on page 152.

To Add a Simple Event

1. In 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.

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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

3. Click **Finish**. You see the Simple Event Editor.
4. Complete the Standard tab and Advanced tab sections as explained in Simple Event Reference on page 157.
5. Save the resource.

The ability to add metadata groups and properties is provided for customization purposes and is not documented.

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 rules you can use the @payload attribute, as well as Event, Mapper, and XPath functions. For simple XML payloads you can use string functions.

To Define an XML Schema File in TIBCO BusinessEvents Studio

An Eclipse resource is available for this task.

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.

To Import an XML Schema File

An Eclipse resource is available for this task.

1. From the File menu select 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**.

To Use 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, do the following.

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. Click **OK** to return to the Advanced tab. 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.

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. See Chapter 28, Domain Objects Configuration, on page 455 for details.

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 on page 313.
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. As a convenience, you can open the selected event resource by clicking the underlined label.

Field	Global Var?	Description
Time to Live	No	Specify a numerical value and a time unit. Time units available are milliseconds, seconds, minutes, hours, and days. (Default unit is seconds.) The numerical value is interpreted as follows: • One or higher (>0): the event expires after the specified number of units elapse. Specify the time units in the drop-down list field to the right of the Time to Live field. Choose one of: Seconds, Minutes, Hours, Days, WeekDays, Weeks, Months, Years • 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) on page 159.
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. As a 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 check box setting: • When this checkbox is checked 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 unchecked TIBCO BusinessEvents does not attempt to reprocess the event instance that failed. See Event Preprocessors on page 265 for more information.

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 on page 313. **Note**: In addition to standard naming restrictions, do not begin an event property name with `_ns_` or `_nm_`. These have a special use. See Mapping Incoming Messages to Non-default Events on page 55. **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. See Using JMS Header Properties in Incoming and Outgoing Messages on page 89 for more details. Table 15, JMS Header Field Names, on page 91 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 Chapter 14, Domain Models, on page 197 for details on 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 on page 253 for details.

See Simple Events — Time to Live and Expiry Actions in *TIBCO BusinessEvents Architect's Guide* for background 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 drop-down list for the Content field in the Payload section:

Table 22 Simple Event Payload Element Parameters (Sheet 1 of 3)

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 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.

Table 22 Simple Event Payload Element Parameters (Sheet 2 of 3)

Content/Parameter		Description
	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.
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 of 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 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.

Table 22 Simple Event Payload Element Parameters (Sheet 3 of 3)

Content/Parameter		Description
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. • Skip: Do not validate. • Lax: Validate if a declaration for the element is available.

Simple Event Attributes Reference

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`	long	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.createEvent()` function. The value 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 for duplicate `extIds` across the cluster. See CDD Agent Classes Tab Properties Reference on page 479 for details.
`@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 on page 157.
`@payload`	string	The payload as a string value. See Payload (Advanced Tab) on page 160 for more on specifying the payload in a SimpleEvent resource.

Chapter 10 Time Events and Scheduler Functions

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 time events.

In addition you can use functions to schedule events. This method can be preferable to using time events in some cases.

Topics

Overview of Time Events

TIBCO BusinessEvents offers two kinds of time events, repeating and rule-based. In addition you can schedule events using functions.

Time events do not go through an event preprocessor. If you are using cache-only cache mode, ensure that any objects are properly loaded. Events scheduled using scheduler functions, however, are sent through channels and would therefore go through event preprocessors in the usual way.

See Also

- Chapter 9, Simple Events, on page 151 for overview information about events in general.
- Chapter 41, Diagrams, on page 685 for information on using diagrams.

Scheduled Time Events

Repeat (Interval) based time events are not supported in multi-engine mode 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.

At engine startup

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 (see Rule Based TimeEvent Function Reference on page 174). You can schedule the event to be asserted after a period of time, and you can pass information to the event and specify its time to live. 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 in order 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 sixty 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 executes, 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.

Working With Time Events

This section explains how to create and configure a time event.

Adding a Time Event

See TimeEvent Resource Reference on page 170 for information on how to complete the values.

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.

   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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

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 on page 168.

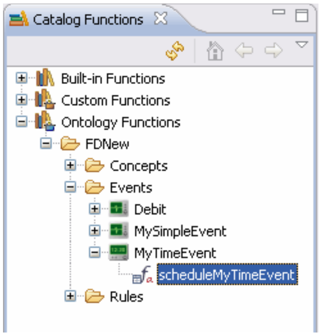
   — repeat: The event occurs at specified intervals. Select Repeat and configure the time event following guidelines in TimeEvent Resource Reference on page 170.

5. Click **Apply** and save the resource.

Configuring a Rule Based Time Event in a Rule or Rule Function

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*).



Catalog Functions
- Built-in Functions
- Custom Functions
- Ontology Functions
  - FDNew
    - Concepts
    - Events
      - Debit
      - MySimpleEvent
      - MyTimeEvent
        - scheduleMyTimeEvent
    - Rules

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 on page 174.
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.

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. See Chapter 28, Domain Objects Configuration, on page 455 for details.

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 on page 313. **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. See Using JMS Header Properties in Incoming and Outgoing Messages on page 89 for more details.
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.

Field	Global Var?	Description
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

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`	long	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 on page 174.

Attribute	Type	Returns
`@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 on page 174.

Rule Based TimeEvent Function Reference

Signature `Event` Schedule*TimeEventName*`(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 executes.
`closure`	`String`	The TimeEvent created will simply store the information passed in the closure parameter. you can put the closure string value in rule conditions to specify the time events that will trigger the rule.
`ttl`	`long`	The event's time to live. Follows the same rules as the time to live setting in a simple event. See Simple Event Reference on page 157.

Returns

Type	Description
`Event`	An instance of the event type specified in the function name (Schedule*TimeEventName).*

Using Scheduler Functions (Requires Cache OM)

The TIBCO BusinessEvents Express edition supports only In Memory object management. Therefore it does not support scheduler functions.

You schedule time events as explained in Overview of Time Events on page 166, and 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, perhaps thousands, 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.

Task A Create a Scheduler

First you 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.

In this release, the larger value of *pollInterval* and *refreshAhead* is used for both these settings.

For example:

```
Cluster.createScheduler("myScheduler", 1000, 5000);
```

Task B Schedule the Event to be Sent

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(String schedulerName, String workKey, SimpleEvent evt, long
scheduledTime)
```

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)
```

Chapter 11 Advisory 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 advisory events.

Topics

Working With Advisory Events

You never have to add or configure an event of type AdvisoryEvent. Advisory events are asserted into the Rete network automatically when certain conditions, for example, exceptions, occur. Add the AdvisoryEvent 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 on page 180).

An advisory event expires after the completion of the first RTC cycle (that is, the time to live is set internally to zero).

Uses of Advisory Events

See Advisory Event Attributes Reference on page 180 for details about use of attributes in each of the following uses.

Exceptions in User Code

The TIBCO BusinessEvents engine automatically asserts an advisory event when it catches an exception that originates in user code but that is not caught with the `catch` command of the TIBCO BusinessEvents Exception type.

For information on working with other kinds of exceptions, see Exception Handling on page 324.

ActiveMatrix BusinessWorks Process Fails or Times Out

Advisory events are also used in the container mode method of TIBCO BusinessEvents-ActiveMatrix BusinessWorks integration feature `invokeProcess()` function. Such events are asserted when the ActiveMatrix BusinessWorks process fails or times out (or is cancelled). See Chapter 38, ActiveMatrix BusinessWorks Integration, on page 609

Engine is Activated

An advisory event (`engine.primary.activated`) is asserted when an engine has finished starting up and executing startup functions, if any (see Engine Startup and Shutdown Sequence in *TIBCO BusinessEvents Administration*).

Adding an Advisory Event to a Rule

1. Open the rule editor for the rule where you want to use the advisory event. (See Adding a Rule on page 244.)

2. Do one of the following:

   — 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.

The AdvisoryEvent 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 on page 180.

Advisory Event Attributes Reference

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 rules to return information about an advisory event. Add-on products may use additional advisory event types.

Name	Type	Description
`@id`	`long`	The event's unique internal ID.
`@extId`	`String`	Null.
`@category`	`String`	Broad category 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 Table 23, Attributes Used for Each Type of Advisory Event, on page 180 for details.
`@message`	`String`	Message content depends on the type of advisory event. See Table 23, Attributes Used for Each Type of Advisory Event, on page 180 for details.

Table 23 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.

Table 23 Attributes Used for Each Type of Advisory Event

Description	@category	@type	@message
Engine is activated	`Engine`	`engine.primary.activated` Indicates the engine has been activated.	`Engine` *EngineName* `activated.`
Hot deployment	`Deployment`	`deployment.hotdeploy.success` `deployment.hotdeploy.fail`	Hot deployed project *ProjectPath* Failed to hot deploy project P*rojectPath*

Chapter 12 Concepts

This chapter explains how to work with concepts.

Topics

Overview of Concepts

You can add concept definitions so that information that arrives in events or from other sources can be organized and persisted as needed, and used in rules. You can add definitions manually. You can also import database tables as concept definitions. Concept instances are created in rules.

See Also

- To gain an understanding of concepts including a detailed discussion of concept history, and concept relationships, Chapter 3, Concepts in *TIBCO BusinessEvents Architect's Guide*
- For working in rules, Working with Concept and Event Properties on page 321.
- To configure runtime properties, including backing store behavior, Chapter 28, Domain Objects Configuration, page 455
- For information on using concept dependency and concept model diagrams Chapter 41, Diagrams, on page 685

Concept Serialization and Handling of Null Value Properties at Runtime

By default, when concept instance objects are serialized to XML, properties with null values are excluded. You can change this behavior so that null values are included. You can also change the XSD for a concept object to allow null values, using the nillable attribute. See Appendix B, Handling Null Properties, on page 729 for details.

Concepts and State Machines

If you are using the TIBCO BusinessEvents Data Modeling add-on product, you can associate a concept with a state machine. See *TIBCO BusinessEvents Data Modeling Developer's Guide* for details.

Adding Concepts and Concept Relationships

This section explains how to work with concepts, and how to set up relationships between concepts.

Adding a Concept

See Concept Resource Reference on page 187 for details on completing values.

To Add a Concept

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

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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

3. Click **Finish**. You see the Concept Editor.
4. Do the following:
   - — Configure the remaining fields as explained in Wizard and Configuration Tab on page 187.
   - — As needed, add and configure properties as explained in Properties on page 188.
5. Save the resource.

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's Guide* for details.

Adding Concept Relationships

See Concept Relationships in *TIBCO BusinessEvents Architect's 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.

To Create a Containment Relationship

A concept can be contained by only one concept at a time.

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: Check the checkbox if this property is an array.

   Policy: Changes Only or All Values.

   History: The number of historical values to keep.

To Create a Reference Relationship

A concept can refer to itself, and can be referred to by more than one concept.

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: Check the checkbox if this property is an array.

   Policy: Changes Only or All Values.

   History: The number of historical values to keep.

   See Concept Property History in *TIBCO BusinessEvents Architect's Guide* for more on this topic.

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.

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.

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. See Chapter 28, Domain Objects Configuration, on page 455 for details.

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 on page 313. 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. • If two unrelated concepts have properties that share a name, an inheritance relationship cannot exist between them.

Field	Global Var?	Description
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 checked then when a concept is asserted, at runtime, its main state machine (if any) is started. If unchecked, 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 checked. Requires TIBCO BusinessEvents Data Modeling.

Properties

The Properties tab has the following fields:

Field	Global Var?	Description
Name	No	The property name.
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. When you create a property of type `ConceptReference` you are creating a property that references another concept. See Concept Relationships in *TIBCO BusinessEvents Architect's 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`.

Field	Global Var?	Description
Multiple	No	Check 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 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 m aximum 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 Concept Property History in *TIBCO BusinessEvents Architect's Guide* for more details. **Note**: Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.

Metadata

The Metadata section is used in a special way with database concepts. Requires TIBCO BusinessEvents Data Modeling. See *TIBCO BusinessEvents Data Modeling Developer's Guide* for details. It is not used otherwise (except for customization, which is not documented).

Concept Attributes Reference

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 rules to return information about a concept instance.

Entity	Attributes	Type	Returns
Concept	@id	long	The concept instance's unique internal ID.
	@extId	string	The concept instance's 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 for duplicate `extIds` across the cluster. See CDD Agent Classes Tab Properties Reference on page 479 for details.
ContainedConcept	@id	long	The contained concept instance's unique internal ID.
	@extId	string	The contained concept instance's unique external ID.
	@parent	concept	The parent concept instance. (This is treated as a concept reference in the language.)

Entity	Attributes	Type	Returns
ConceptReference	@id	long	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`.

Chapter 13 Scorecards

This brief chapter explains how to work with scorecards.

Topics

Understanding and Working With Scorecards

A scorecard is a special type of concept. A scorecard serves as a set of static variables that is available throughout the project. You can use a scorecard resource to track key performance indicators or any other information.

Unlike concepts and events, each scorecard resource is itself a single instance — it is not a description for creation of instances. You create the scorecard at design time. Its values can be viewed and updated using rules.

It is more accurate to say there is one instance of a scorecard per inference agent. Each inference agent in an application has its own instance of the score card. Scorecards are not shared between agents.

Any agent that uses scorecards, and also uses Cache OM, must be assigned a unique key so that the correct scorecard can be retrieved from the cache. The key is set in the Processing Unit tab of the CDD. See Configuring Processing Units (All OM Types) on page 492.

It is not necessary to add scorecards to the declaration of a rule. Because there is only one instance of each scorecard in a deployed TIBCO BusinessEvents agent, any change causes all rules that use the scorecard in their conditions to be evaluated.

The `Instance.isModified()` function works differently with scorecards than with concepts. There is only one instance of a scorecard per agent, rather than one per RTC. So after a scorecard is modified it will return true until the agent is restarted.

(In the case of a concept instance, `Instance.isModified()` returns true after the instance has been modified only for the rest of the RTC in which it is modified.)

See Also

- Chapter 12, Concepts, on page 183.
- Chapter 41, Diagrams, on page 685 for information on using scorecard dependency diagrams.

Adding a Scorecard

Configuring a scorecard is similar to configuring a concept, except that scorecards do not have relationships.

1. In 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. In the Scorecard Name field, type a name for the scorecard. In the Description field, type a description.

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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

3. Click **Finish**. You see the Scorecard Editor.
4. Add and configure properties as explained in Concept Resource Reference on page 187.
5. Save the resource.

The ability to add metadata groups and properties is provided for customization purposes and is not documented.

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 on page 187 for details about scorecard properties

Chapter 14 Domain Models

Domain model resources enable you to control user input for decision tables and test data. This chapter describes how to add and import domain models and associate them with entity properties.

Topics

Overview of Domain Models

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. Decision tables are available in the TIBCO BusinessEvents Decision Manager add-on. See TIBCO BusinessEvents Decision Manager documentation for details about working with decision tables.

You cannot add domain models for concept properties of the `ContainedConcept` or `ConceptReference` type.

Setting up domain models has two steps:

1. Add domain models. For example, create a folder and add all the models you need for a project in this folder. See Adding a Domain Model on page 199 for details.

   You can also import (and export) domain models. See Importing and Exporting Domain Models on page 203.

2. Associate domain models with properties. See Associating Domain Models with Properties on page 206 for details.

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 will 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, where 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 on page 206

To Add a Domain Model

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.

> 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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

3. Click **Finish**. You see the Domain Model editor.

4. In the Domain Type field, select the data type for this domain model. See Supported Data Types on page 200 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. See To Add Domain Entries on page 200 for examples.

8. When you have created the entries for the domain model, save the domain model resource.

To Add Domain Entries

When you add a domain model, you 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 selected for the domain model.

Supported Data Types

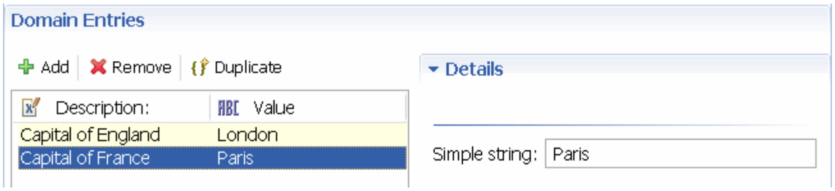
Domain Models support the following data types:

- String
- Integer
- Long
- Double
- Boolean
- DateTime

Sections below show the user interface for each type.

String

String entries are simple text strings.



Description:	Value
Capital of England	London
Capital of France	Paris

Simple string: Paris

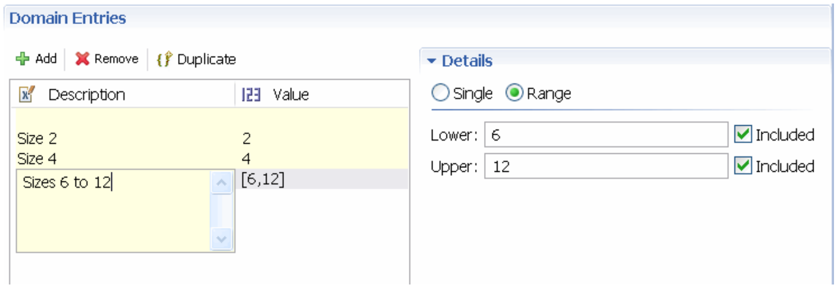
Numeric values in a String domain type When you use a numeric value (Integer, Double, or Long) in a domain model of type String, TIBCO BusinessEvents adds double quotes around the value. (These are visible after you save and reopen the domain model.)

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 datatypes.

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 datatypes. Here are some single values and a range value for an integer domain model:



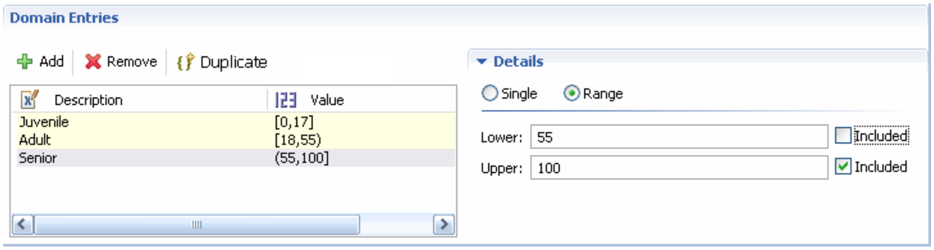
Description	Value
Size 2	2
Size 4	4
Sizes 6 to 12	[6,12]

○ Single ● Range

Lower: 6 ☑ Included

Upper: 12 ☑ Included

The next figure shows a set of ranges. Note that 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 check box. For example, the figure shows that the lower bound of the Senior range, 55, is not included.



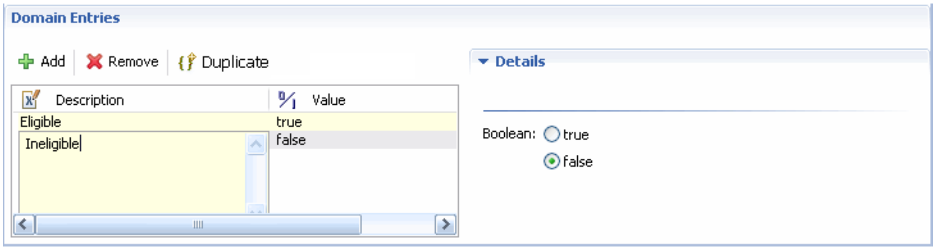
Description	Value
Juvenile	[0,17]
Adult	[18,55)
Senior	(55,100]

○ Single ● Range

Lower: 55 □ Included

Upper: 100 ☑ Included

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.

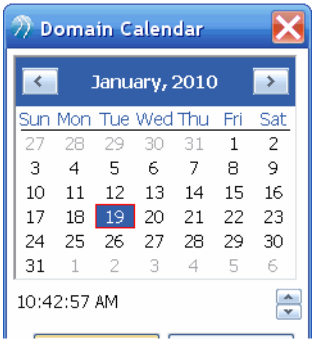


Domain Entries

Description	Value
Eligible	true
Ineligible	false

Boolean: ○ true ● false

DateTime In a single domain model, you can enter single values, range values, or a mixture of both. You specify a date and a time. A date and time picker enable you to select the date and time:



Domain Calendar

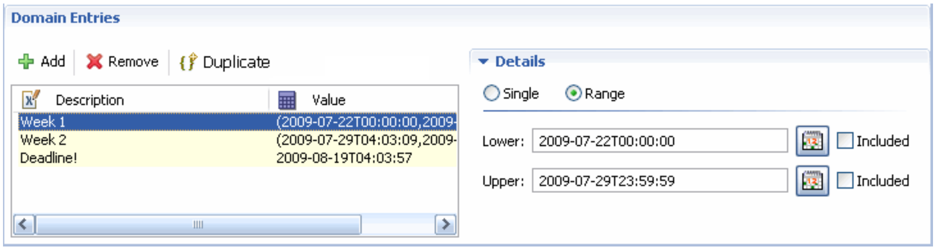
January, 2010

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

10:42:57 AM

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.



Domain Entries

Description	Value
Week 1	(2009-07-22T00:00:00,2009-
Week 2	(2009-07-29T04:03:09,2009-
Deadline!	2009-08-19T04:03:57

○ Single ● Range

Lower: 2009-07-22T00:00:00 □ Included

Upper: 2009-07-29T23:59:59 □ Included

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. (Database concepts are available in the TIBCO BusinessEvents Data Modeling add-on).

After you import a domain model, associate it with a property. See Associating Domain Models with Properties on page 206

Import From the Source of a Database Concept

You can import all values from the database column that corresponds to a database concept property for use as the domain model entry for that property. This is explained in *TIBCO BusinessEvents Data Modeling Developer's Guide*.

Export To and Import 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's Guide* for details.

To Import Domain Model Entries from Excel

1. In BusinessEvents Studio Explorer, do one of the following:
   — Right-click the folder where you want to create the domain model and select **Import > TIBCO BusinessEvents > Domain Model**.
   — In 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**. You see a message, "Domain Import Successful." Click **OK**.
8. You see the Domain Model editor. The column values appear as domain entries. You can add, edit, duplicate, and remove entries as appropriate.

To Export Domain Values to Excel

1. In BusinessEvents Studio Explorer do one of the following:

   — Right-click the domain you want to export and select **Export > TIBCO BusinessEvents > Export Domain to Excel**.

   — In 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**. You see a message, "Export Domain Model to File Successful." Click **OK**.

Import 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 Database.

To Import Domain Model Entries from a Database Table

1. Add a JDBC Connection resource and configure it to connect to the database from which you want to import the domain model. See JDBC Connection on page 221 for details.

2. In BusinessEvents Studio Explorer, do one of the following:

   — Right-click the folder where you want to create the domain model and select **Import > TIBCO BusinessEvents > Domain Model**.

   — In BusinessEvents Studio Explorer, select any item in the project entity and select **File > Import > TIBCO BusinessEvents > Domain Model**.

3. 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.

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. Do one of the following:

   — 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 resultset 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**. You see a message, "Domain Import Successful." Click **OK**.

10. 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.

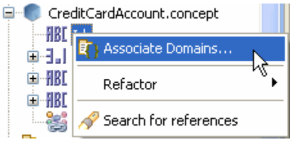
If domain models have the same data type as that of the properties, you can associate multiple domain models with multiple properties.

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.

To Associate Domain Models with a Property

1. Do one of the following:

   — In BusinessEvents Studio Explorer, expand the concept, event, or scorecard to display its properties. Right-click the desired property and select **Associate Domains**.

   — In 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.

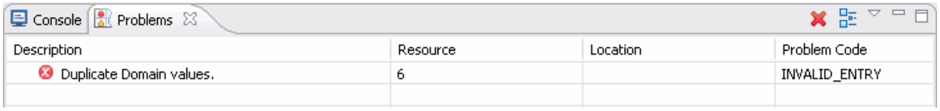
3. Save the domain model resource.

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. For example:

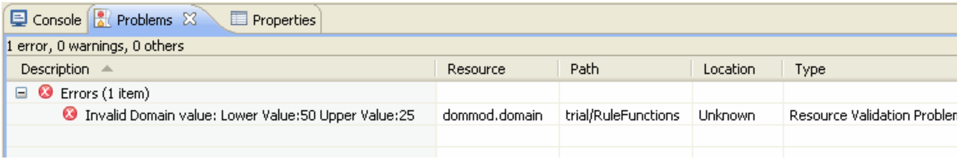


Description	Resource	Location	Problem Code
Duplicate Domain values.	6		INVALID_ENTRY

The Resource column displays the duplicate value.

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. For example:



1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
Invalid Domain value: Lower Value:50 Upper Value:25	dommod.domain	trial/RuleFunctions	Unknown	Resource Validation Probler

Chapter 15 Shared Resources

This chapter describes a set of resources that can be used for various purposes in your projects.

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 Chapter 38, ActiveMatrix BusinessWorks Integration, on page 609.

Topics

Adding a Shared Resource

See the section Working with External Library and Custom Function Paths on page 8 for actions you must do to run, test, or debug projects that use third-party libraries in TIBCO BusinessEvents Studio.

If you are adding a JDBC connection for a backing store, see the procedure, Add a JDBC Connection Resource (Now or Later) on page 512.

It's a good idea to use global variables in shared resources so that projects can be quickly adapted to run in different environments.

To Add a Shared Resource

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:

   — ActiveSpaces Connection, page 212

   — Hawk Connection, page 214

   — HTTP Connection, page 216

   — Identity Resource, page 219

   — JDBC Connection, page 221, and also see Enabling the Test Connection Feature, page 211

   — JMS Application Properties, page 225

   — JMS Connection, page 226, and also see Enabling the Test Connection Feature, page 211

   — JNDI Configuration, page 233

   — Rendezvous Transport, page 236
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`.)

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.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.

ActiveSpaces Connection

The ActiveSpaces Connection resource describes the connection to a TIBCO ActiveSpaces metaspace. This section provides a reference to the fields. For procedures see Adding a Shared Resource on page 210.

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 on page 313. Unlike other resource identifiers, however, 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 TIBCO ActiveSpaces metaspace. A metaspace is an instance of a cluster of application processes deployed using TIBCO 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 (Cont'd)
Multicast URL	Yes	Specifies the URL to be used for multicast discovery. The multicast URL is specified using the following format: `tibpgm://[`*dport*`]/[`*interface*`]/[`*discovery IP multicast address*`]/[`*option=value;*`]*` where, • *dport* specifies the destination port used by the PGM transport. If not specified, it will use the default value of 7888. • *interface;discovery IP multicast 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 will default to the default interface and discovery address, 239.8.8.8. • *option=value* specifies a semicolon separated list of optional Pragmatic General Multicast (PGM) transport arguments. For example: source_max_trans_rate=100000000 (in bits per second) would limit the PGM transport to limit its transmission rate to 100 megabits per second.
Unicast Url	Yes	Specifies the URL to be used for unicast discovery. The URL specifies the interface and port to be used to create the listening socket on. The unicast URL is specified using the following format: `tcp://`*ip1*`[:`*port1*`];`*ip2*`[:`*port2*`],...` where any number of *ip*`[:`*port*`]`well known address can be listed. If port is not specified, the default port number value of 50000 will be assumed.

Hawk Connection

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 on page 210.

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 on page 313. Unlike other resource identifiers, however, 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.
Transport	No	Type of transport used byHawk 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.

Field	Global Var?	Description (Cont'd)
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_ServerURL	Yes	Specifies the EMS server to connect to. The server URL is 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

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 on page 210.

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 Configure SSL on page 217 for more information.

⚠ 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.

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 on page 313. 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.

Field	Global Var?	Description (Cont'd)
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 will be 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 will 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 socket layer, or SSL) protocol. This protocol authenticates the server to the client, and optionally authenticates the client to the server. Enabling this field allows you to specify SSL parameters with the Configure SSL button (see Configure SSL on page 217).

Configure SSL

The SSL Configuration for HTTPS Connections dialog (accessed by configuring the Configure SSL button) allows you to specify the SSL parameters for the HTTP connection.

The following are the fields in the SSL Configuration for HTTPS Connections dialog:

The HTTPComponent server type does not support Entrust based SSL.

Field	Description
Requires Client Authentication	Checking this field requires clients to present their digital certificate before connecting to the HTTP server. When this field is checked, 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 checked. This field specifies a folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked 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 on page 219 for more information.
Strong Cipher Suites Only	When checked, 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 unchecked, only cipher suites with an effective key length of up to 128 bits can be used.

Identity Resource

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 on page 210.

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		
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 on page 313. Unlike other resource identifiers, however, 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).

Field	Global Var?	Description (Cont'd)
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 Use this option if the private key and the certificate are in two separate files.		
Certificate URL	No	Location of the certificate. Click the browse icon or type in a URL.
Key URL	No	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

The JDBC Connection resource describes a JDBC connection. JDBC connections are used with backing stores (see Chapter 30, JDBC Backing Store Setup, on page 501) and with database concepts (available in the TIBCO BusinessEvents Data Modeling add-on). This section provides a reference to the fields. For procedures see Adding a Shared Resource on page 210.

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 on page 313. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.
Configuration		
Description	Yes	Short description of the resource.
Connection Type	No	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.

Field	Global Var?	Description (Cont'd)
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`
		`com.sybase.jdbc3.jdbc.SybDriver`
		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.
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 on page 224 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.

Field	Global Var?	Description (Cont'd)
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 checked, the JNDI Configuration field appears, allowing you to choose a shared JNDI Configuration resource. When this checkbox is unchecked, configuration fields appear.
JNDI Configuration	No	This field only appears when the Use Shared JNDI Configuration field is checked. This field allows you to choose a JNDI Configuration shared resource that specifies the JNDI connection information.
JNDI Context Factory	No	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	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	The user name 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	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.

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 will 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. See Configuring Backing Store Settings and Properties on page 434 for details.

Test Connection Button

The Test Connection button allows you to test the connection specified in the configuration of this resource. See Enabling the Test Connection Feature on page 211 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 on page 210.

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 on page 313. 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 datatype 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

JMS Connection resource describes a JMS connection. This section provides a reference to the fields. For procedures see Adding a Shared Resource on page 210.

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 on page 313. Unlike other resource identifiers, however, 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	User name 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.

Field	Global Var?	Description (Cont'd)
Auto-Generate Client ID	No	Checking 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 checked, if a value is specified 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 checked, 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. If the value specified in the Client ID field is already in use, an exception is thrown.
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 on page 230 for more information.
Use JNDI for Connection Factory	No	Specifies whether JNDI should be used to look up the ConnectionFactory object. If this field is unchecked, the Provider URL and Use XA Connection Factory fields appear. If this field is checked, JNDI configuration fields appear.
Provider URL	Yes	This field is only available when the Use JNDI for Connection Factory field is unchecked. This is the URL to use to connect to the JMS server.
Use XA Connection Factory	No	This field is only available when the Use JNDI for Connection Factory field is unchecked. When checked, this field specifies that an XA connection factory is to be used.

Field	Global Var?	Description (Cont'd)
Connection Factory SSL Password	Yes	This field is only available when the Use SSL checkbox is checked, and the User Shared JNDI Configuration checkbox is unchecked. 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	No	When this field is checked, the JNDI Configuration field appears. It allows you to choose a JNDI Configuration. When this field is unchecked, JNDI configuration fields appear.
JNDI Configuration	No	This field only appears when the Use Shared JNDI Configuration field is checked. This field allows you to choose a JNDI Configuration that specifies the JNDI connection information. See JNDI Configuration on page 233 for more information.
JNDI Context Factory	Yes	This field only appears when the Use Shared JNDI Configuration field is unchecked. 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 unchecked. Required. This is 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.

Field	Global Var?	Description (Cont'd)
JNDI User Name	Yes	This field only appears when the Use Shared JNDI Configuration field is unchecked. User name 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 unchecked. 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

The Test Connection button allows you to test the connection specified in the configuration of this resource. See Enabling the Test Connection Feature on page 211 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 checked), 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 checked. 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.

Field	Global Var?	Description (Cont'd)
Queue Connection Factory	Yes	This field is only available when the Use JNDI for Connection Factory field on the Configuration tab is checked. 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.
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.

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 will 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.

Field	Description (Cont'd)
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 on page 219 for more information.
Trace	Specifies whether SSL tracing should be enabled during the connection. If checked, 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 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)`

Field	Description (Cont'd)
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 checked. 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 checked, 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 unchecked, only cipher suites with an effective key length of up to 128 bits can be used.

JNDI Configuration

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 on page 221 and JMS Connection on page 226 can use JNDI connections. This section provides a reference to the fields. For procedures see Adding a Shared Resource on page 210.

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 on page 313. Unlike other resource identifiers, however, 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.

Field	Global Var?	Description (Cont'd)
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. See your JNDI provider documentation for the syntax of the URL.
JNDI User Name	Yes	User name 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 checked. Checking 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. Checking this field causes additional overhead for activities that use this resource. Therefore, only check this field when necessary.

Field	Global Var?	Description (Cont'd)
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 the documentation for your JNDI provider for more information about properties that can be passed to the JNDI server.

Rendezvous Transport

The Rendezvous Transport resource describes a TIBCO Rendezvous transport. See the TIBCO Rendezvous documentation for more information about specifying these fields.

This section provides a reference to the fields. For procedures see Adding a Shared Resource on page 210.

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 on page 313. Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name. This field appears on the New Rendezvous Transport Wizard. The name then appears in the title of the resource.
Configuration		
Description	Yes	Short description of the resource.
Daemon	Yes	In the case of TIBCO Rendezvous daemon running on the same machine as TIBCO BusinessEvents engine, this is not specified. If Rendezvous is running on a different machine, then the Daemon field is specified as the remote host name followed by the socket number. For example: `ssl:acct:5785`

Field	Global Var?	Description (Cont'd)
Network	Yes	This field contains the host name, IP address, network name, or interface name. For example: `;224.34.103.4`
Service	Yes	The TIBCO Rendezvous service name, specified in one of the following formats: *service name* or *port number*.
Use SSL	No	The Use SSL checkbox specifies that Secure Sockets Layer (SSL) should be used when communicating with the TIBCO Rendezvous daemon. When this field is checked, the Configure SSL button is enabled. See Configure SSL Button on page 237 for more information about configuring SSL parameters.

Configure SSL Button

The Configure SSL button allows you to configure the SSL parameters for communicating with the TIBCO Rendezvous daemon. See the TIBCO Rendezvous documentation for more information about how SSL is configured for TIBCO Rendezvous daemons and clients.

The SSL Configuration for TIBCO Rendezvous dialog has the following fields:

Field	Description
Daemon Certificate	File containing one or more certificates from trusted certificate authorities. This file is checked when connecting to a daemon to ensure that the connection is to a daemon that is trusted. This prevents connections to rogue TIBCO Rendezvous daemons that attempt to impersonate trusted daemons. You can retrieve a daemon's certificate using the administration interface in TIBCO Rendezvous. See the TIBCO Rendezvous documentation for more information about obtaining certificates through the administration interface. Once retrieved, you can select a folder in your project and import this certificate into the folder using the Tools > Trusted Certificates > Import Into PEM Format menu item.

Field	Description (Cont'd)
Identity	This is an Identity resource used to authenticate to the TIBCO Rendezvous daemon. The Browse button allows you to select from a list of appropriately configured Identity resources. Only Identity resources whose Type field is set to Identity File or Username/Password are listed. See Identity Resource on page 219 for more information.

Advanced Section

The Advanced section of the UI has the following fields.

Field	Global Var?	Description
RV Type	No	The type of TIBCO Rendezvous connection to use. This can be reliable (standard RV transport), certified (RVCM), or Distributed Queue (RVCMQ). The fields of the Advanced tab correspond to the value selected for this field.
Certified Transport		
CM Name	Yes	The name of the delivery-tracking session. This name is in the same format as TIBCO Rendezvous subject names.
Ledger File	Yes	The name and location of the persistent ledger file that tracks certified messages. If not specified, the certified message ledger is kept in process memory only.
Sync Ledger File	Yes	Specifies whether to keep the ledger file synchronous with the current messages.
Relay Agent	Yes	Name of the relay agent to use. Relay agents are useful when clients are disconnected from the network from time to time. The relay agents store inbound certified messages and labeled messages (and other messages related to certified delivery features) on behalf of their disconnected client programs. When a client is connected, it receives inbound messages immediately.

Field	Global Var?	Description (Cont'd)
Require Old Message	Yes	Check this box if you wish to require the retention of messages for which delivery has not been confirmed. These messages will be resent.
Message Timeout (sec)	Yes	The time limit (in seconds) for certified message delivery.
Distributed Queue		
CMQ Name	Yes	The name of the distributed queue. This name is in the same format as TIBCO Rendezvous subject names.
Worker Weight	Yes	The weight of the worker (this pertains to the worker processing queue requests). Relative worker weights assist the scheduler in assigning tasks. When the scheduler receives a task, it assigns the task to the available worker with the greatest worker weight.
Worker Tasks	Yes	Sets the task capacity for the worker (this pertains to the worker processing queue requests). Task capacity is the maximum number of tasks that a worker can accept. When the number of accepted tasks reaches this maximum, the worker cannot accept additional tasks until it completes one or more of them.
Worker Complete Time	Yes	The amount of time the scheduler waits for a worker process to complete. If the worker process does not complete in the specified period, the scheduler reassigns the message to another worker.
Scheduler Weight	Yes	Weight represents the ability of this member to fulfill the role of scheduler, relative to other members with the same name. Cooperating distributed queue transports use relative scheduler weight values to elect one transport as the scheduler; members with higher scheduler weight take precedence. Acceptable values range from 1 to 65535.
Scheduler Heartbeat	Yes	The scheduler sends heartbeat messages at this interval (in seconds). All members with the same name must specify the same value for this parameter. The value must be strictly positive.

Field	Global Var?	Description (Cont'd)
Scheduler Activation	Yes	When the heartbeat signal from the scheduler has been silent for this interval (in seconds), the member with the greatest scheduler weight takes its place as the new scheduler. All members with the same name must specify the same value for this parameter. The value must be positive

Chapter 16 Rules and Rule Functions

This chapter explains how to work with rules and rule functions.

For documentation on catalog functions see Chapter 18, Functions, page 279

Topics

Overview of Rules and Rule Functions

This chapter explains how to work with the rule editor to create rules and rule functions. It also explains some specific uses for rule functions.

To learn how rules are executed at runtime and for other background information see *TIBCO BusinessEvents Architect's Guide*, especially the section Understanding Conflict Resolution and Run to Completion Cycles.

See Also

- Chapter 4, Rules and Functions in *TIBCO BusinessEvents Architect's Guide*
- Chapter 18, Functions, on page 279
- Chapter 19, Rule Language Grammar, on page 311
- Chapter 20, Rule Language Datatypes, on page 329
- Chapter 41, Diagrams, on page 685 for information on using rule dependency and sequence diagrams.

Form-based and Source Rule Editors

When you work with rules and rule functions, you can choose how to work:

- Using a form-based rule editor, similar to the rule editor in earlier versions of TIBCO BusinessEvents
- Using a source editor, which is closer to a Java programming environment.

You can switch between editors and changes made in one editor are reflected in the other one. You cannot switch from the source editor to the form editor if there are validation errors in the code.

Rule Components

A TIBCO BusinessEvents rule has three components:

- **Declaration** — Use the declaration to declare which concepts and events the rule will depend on, and the names by which instances of these entities can be referred to in the conditions and actions. Aliases must be valid identifiers. Declaring multiple terms of the same type allows the rule to consider multiple instances of the corresponding entity.

- **Conditions —** Each statement in the condition must evaluate to a boolean value. All of these statements must be true for the rule's action to be executed. Assignments and calls to certain functions are disallowed in the condition.
- **Actions —** List of statements that will be executed, when the rule is fired, for each combination of terms that matches all the conditions.

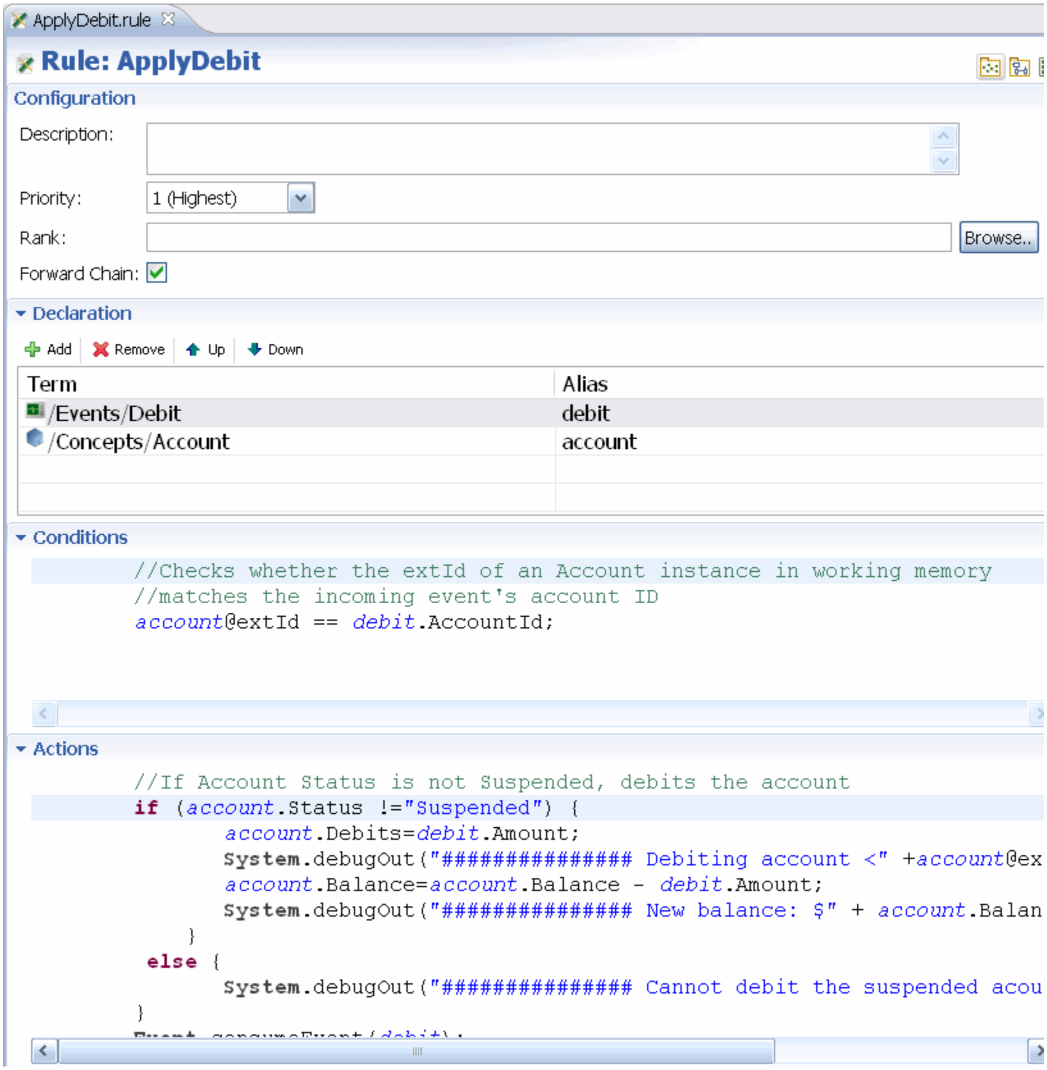
Effect of Cache Only Cache Mode

When using Cache Only cache mode for one or more entities, you must consider how to handle the cache-only entities when you write rules and preprocessor rule functions. See Working With Cache Modes and Loading Cache Only Objects into the Rete Network in *TIBCO BusinessEvents Architect's Guide*.

Adding a Rule

These instructions focus on the form-based editor and mention the equivalent settings in the source editor. Adapt the instructions if you are using the source editor.

Figure 2 Rule Form Editor



Description:

Priority: 1 (Highest)

Rank:

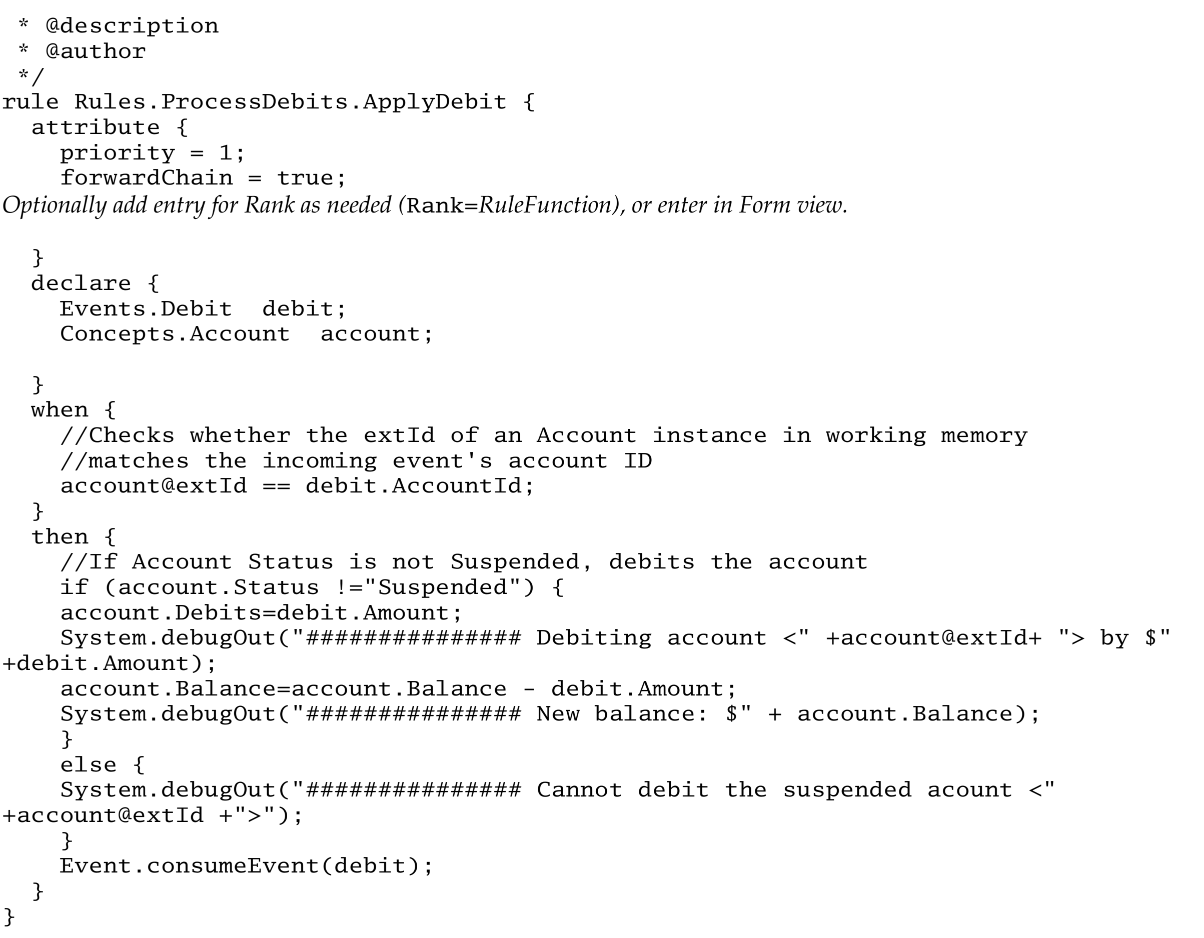
Forward Chain: ☑

Term	Alias
/Events/Debit	debit
/Concepts/Account	account

```
        //Checks whether the extId of an Account instance in working memory
        //matches the incoming event's account ID
        account@extId == debit.AccountId;
```

```
        //If Account Status is not Suspended, debits the account
        if (account.Status !="Suspended") {
               account.Debits=debit.Amount;
               System.debugOut("############### Debiting account <" +account@ex
               account.Balance=account.Balance - debit.Amount;
               System.debugOut("############### New balance: $" + account.Balan
            }
         else {
               System.debugOut("############### Cannot debit the suspended acou
        }
        Event.consumeEvent(debit);
```

Figure 3 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);
  }
}
```

To Add a Rule

See Rule Editor Reference on page 248 for details on the settings.

1. In BusinessEvents Studio Explorer, right click the folder where you want to store the 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.)

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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

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.

Rule Editor Preference To set which mode the editor uses upon first opening, go to Window > Preferences > TIBCO BusinessEvents > Rules and check or uncheck 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. If you want to control the order in which rules execute, set the Priority field accordingly. Highest priority is 1. (In the source editor set `priority = ` *n* where n is the priority number.)

      If you want to also control the order in which rules with the same *priority* execute, set the Rank field accordingly. Browse to and select the rule function you created for this purpose. (If you want 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 on page 259 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 do the following:

   a. Click **Add** to add resources that you will be 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 re-order 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 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 evaluate to true for the conditions to be satisfied. See Order of Evaluation of Rule Conditions in *TIBCO BusinessEvents Architect's Guide* for more information.

   See Using Variables and Functions in the Rule Editor on page 255 and Tips for Working in the Rule Editor on page 263 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 on page 313.
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 on page 259 for details. Default is 0.0.
Forward Chain	Determines if the rule is used in forward chaining. If the checkbox is unchecked, changes made by this rule won't trigger other rules. Default is checked.

Property	Description
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`. 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 on page 313.
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 (see Order of Evaluation of Rule Conditions in *TIBCO BusinessEvents Architect's Guide*.)
Actions Section	
	List of statements that will be executed when the rule is fired.

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. Virtual rule functions are used only with the TIBCO BusinessEvents Decision Manager add-on software. The implementation for virtual rule functions is provided by one or more decision tables. See TIBCO BusinessEvents Decision Manager product documentation for more details.

To Add a Rule Function

See Rule Function Resource Reference on page 253 for details on completing values.

1. In 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), check the **Virtual** checkbox.

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 Chapter 3, Element Refactoring Operations, on page 37 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.

Rule Function Editor Preference To set which mode the editor uses upon first opening, go to Window > Preferences > TIBCO BusinessEvents > Rules and check or uncheck the following checkbox as desired: **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 check box according to your need. Check the check box if you are creating a virtual rule function (to be implemented by a TIBCO BusinessEvents Decision Manager decision table).

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 BusinessEvents Studio Explorer, OR do the following:

   a. Click **Add** to add resources that you will be 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 you want to specify an array, check 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 on page 255 and Tips for Working in the Rule Editor on page 263 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 on page 313.
Description (Editor and Wizard)	Short description of the resource.
Return Type (Editor and Wizard)	If the rule function returns a value, specify the Return Type, otherwise leave set to void. See the list of valid types for the Scope Section area, next.
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 is disabled and so is the Return Type field. The body is provided by a decision table. See *TIBCO BusinessEvents Decision Manager User's Guide*.
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 (if you have the TIBCO BusinessEvents Event Stream Processing add-on).

Property	Description
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). **Note**: Only Action rule functions can be used as startup rule functions or shutdown rule functions.
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. For more details, see Chapter 20, Rule Language Datatypes, on page 329.
Alias	Each argument requires a type and an alias. Names must be valid identifiers. See Identifier Naming Requirements on page 313.
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.

1. When adding code in the rule editor, do one of the following.

   — Begin to type the function name, beginning 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 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*.

See Overview of Catalog Functions on page 280 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 Mapper on page 256.

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 on page 14 for more details about global variables.

Using the Function Argument Mapper

For functions known as mapper functions you can use the Function Argument Mapper to map inputs from a source to the function arguments.

See Chapter 21, Mapping and Transforming Data, on page 335 for a reference to using the Function Argument Mapper.

To Open the Function Argument Mapping Wizard

You can open the Function Argument Mapping Wizard in various ways. For example here is one way:

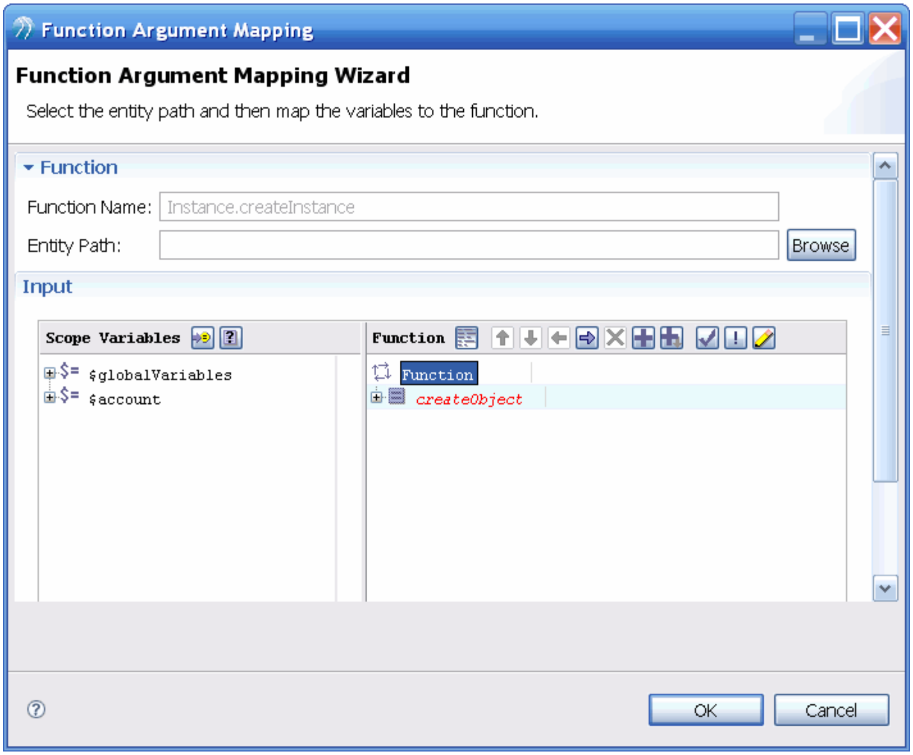
1. In the rule editor, type the category of function you want to use. A list of functions in that category appears.
2. Select a mapper function from the list of functions in that category. The function argument area contains the text `"xslt://"`.
3. Control-click the text `"xslt://"` to display a hypertext link. Click the link to open the Function Argument Mapping Wizard.

Other ways to perform the same action are as follows:

- Type the name of a mapper function category into the **Actions** or **Conditions** areas, then type an open parenthesis ("(") TIBCO BusinessEvents displays `"xslt://"`. Control-click 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.

To Work with the Function Argument Mapper Wizard

See Chapter 21, Mapping and Transforming Data, on page 335 for full details on using the wizard. This section provides summary information only.



Function Argument Mapping

Function Argument Mapping Wizard

Select the entity path and then map the variables to the function.

Function Name: Instance.createInstance

Entity Path:

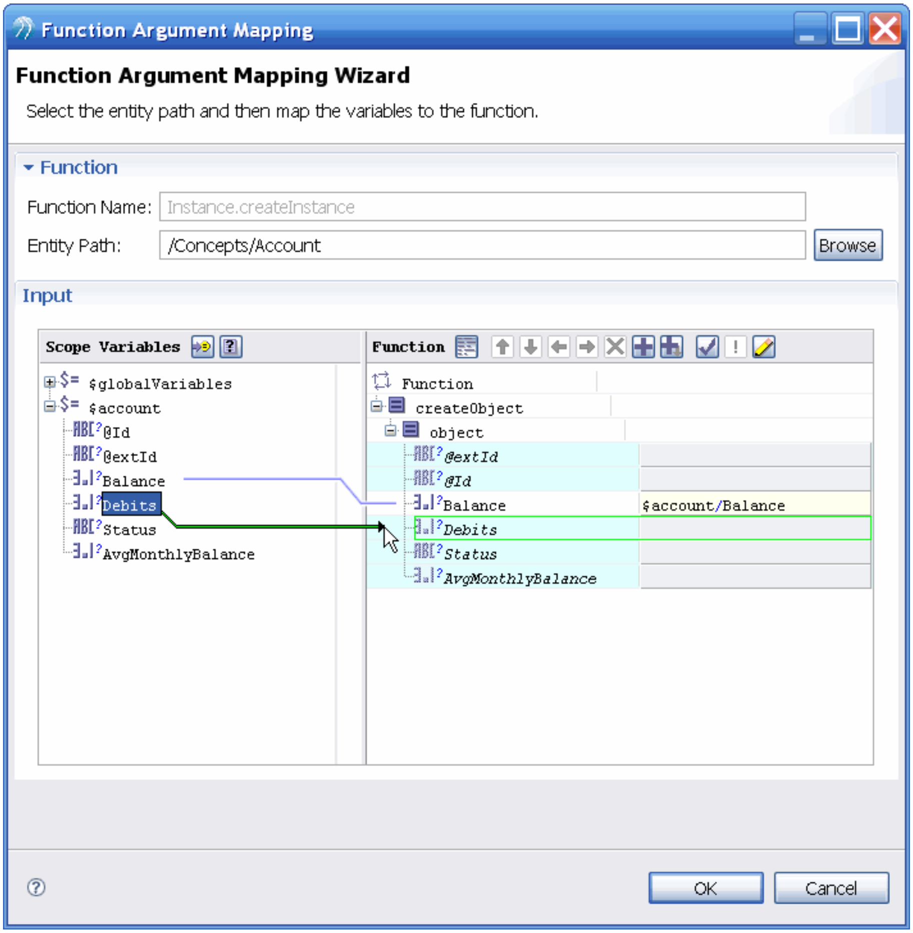
1. In the Function section, take appropriate action. For example, if you are using `Instance.createInstance()`, you would click the browse icon (🔍) to the right of the **Entity Path** field and select a concept type so you can create an instance of it.

2. Select a resource and click **Apply**.

3. Select the **Input** tab. TIBCO BusinessEvents displays a list of variables associated with the project on the left and a list of properties for the selected resource type on the right.

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, as shown:



Function Argument Mapping

Function Argument Mapping Wizard

Select the entity path and then map the variables to the function.

Function

Function Name: Instance.createInstance

Entity Path: /Concepts/Account

Input

6. If you want to define the arguments using more complex logic, type the code or click the XPath Formula Builder (✎) button to use the XPath Formula Builder. See Chapter 22, XPath Formula Builder, on page 373.

7. Click **OK**.

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'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.

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. This can be a subset of what's in the rule declaration. (As with rules, primitives are not allowed.)
  - 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 two rules declare CustName and SupportLevel (plus other things that can be different). 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 rule tuples with priority 1, those for customers with Gold support execute before those for customers with Silver support, which execute 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 1 (Tuple X)	Priority: 5	Rank: 10.0
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 executed as follows. (This could change during a conflict resolution cycle, of course, depending on the effect of rule actions.)

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 execute before those that were asserted later. The time values returned by three rules are 100, 150, and 300. Using -1.0 * *time*), they will be 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.

> 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.

To Use the Quick Fix Feature

1. Type the code in the rule editor as if the entity, property, or rule function you want to use already exists. See Table 24, Quick Fix Feature Options, on page 261 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 Table 24:

Table 24 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.

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.

Table 24 Quick Fix Feature Options

Unknown Reference	Quick Fix Editor
Qualifier is a concept or event and reference is unknown. Example:	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.

`Concepts.Person person;`
`person.someProperty = "123"`
	In the example, the `someProperty` property does not already exist as a property of `Concepts.Person`.

Tips for Working in the Rule Editor

This section has some tips to help you work in 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 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 execute in a particular order, leave the Priority set to the default and let the runtime engine determine rule order.

Declaring multiple terms of the same type 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 doesn't 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).

When you’re 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.

Table 25 Tips for Working in the Rule Editor

To do this...	Do this...
Switch Between Source and Form Editors	Click the bottom tabs 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 in the item name and press **F3**, or right-click and select **Open Declaration**. • Press and hold the Control key while you move (hover) the mouse pointer 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.

Event Preprocessors

Event preprocessors are rule functions with one argument of type simple event. They perform tasks after an incoming message is transformed into a simple event but before it is asserted into the Rete network.

An event preprocessor is assigned to a destination and acts on all events arriving at that destination.

See Event Preprocessors in *TIBCO BusinessEvents Architect's Guide* for important information on the use of preprocessors.

If an event preprocessor fails due to an exception, the Retry on Exception field setting on the event type determines subsequent behavior (see Simple Event Reference on page 157).

Loading Cache Only Entities If you are using the Cache Only cache mode for any entities, it is recommended that you load the relevant entities in the event preprocessor. See Cache Related Functions on page 292.

Configuring an Event Preprocessor

To configure a preprocessor you associate the desired rule function with a destination using the Cluster Definition Descriptor editor. See Configuring Collections of Rules, Rule Functions, and Destinations on page 470.

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.

This feature requires cache-aside database write strategy. When backing store is enabled, the default value for the property `Agent.`*agentclassname*`.enableParallelOps` is true. The feature is not supported when the property value is set to false:

`Agent.`*agentclassname*`.enableParallelOps=false`

When the backing store is disabled, the default values for the cache-aside database write strategy and the property `Agent.`*agentclassname*`.enableParallelOps` are false.

Set these two values to true explicitly in the CDD file in order to use the transaction error handler rule function.

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 retrying 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 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. • `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: — have been created in the current RTC, — and have not been deleted. • `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.

Argument	Notes
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.

Chapter 17 Rule Templates

Rule templates and rule template views are used for creating business rule in TIBCO BusinessEvent WebStudio. This chapter explains what they are for and how to create them. See Chapter 35, Business Rules, on page 585 for more information on business rules.

Topics

Rule Template and Rule Template View Overview

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.

This chapter assumes familiarity with rules. For details on creating rules, see Chapter 16, Rules and Rule Functions, on page 241

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.

Another difference is that the Pre-conditions section can contain non-conditional statements, for example, local variable declarations, to provide more flexible and comprehensive condition checking capabilities. Here is a simple example of such a section:

```
int i=con.prop1;
int j=con2.prop1;
```

```
i > j;
i == j;
```

This feature enables complex calculations to be done before the definition of the condition. The Pre-conditions section could also simply 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, 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 only has to enter 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. 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. 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 users and enable them to define 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 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 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. 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 will display the input widget as a drop down. Otherwise, an input widget of the appropriate type is displayed, for example, a check-box for a boolean field.

Adding a Rule Template

See Rule Template and Rule Template View Overview on page 270 for an introduction to this feature, and Rule Template Editor Reference on page 277 for a reference to the editor.

You can associate domain models with bindings to restrict the values WebStudio users can enter. Add the domain models first.

Some familiarity with rules is assumed. See Chapter 16, Rules and Rule Functions, on page 241 for details.

To Add a Rule Template

1. Right-click the folder where you want to store rule template, and select **New > Other > TIBCO BusinessEvents > Rule Template**. You see the New Rule Template Wizard.
2. In the Rule Template Name field, type a name for the rule template. In the Description field, type a description.

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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

3. Click **Finish**. You see the Rule Template editor.

Configuration

4. Complete the Configuration section as you would for a rule (see Adding a Rule on page 244).

Declarations and Variables

5. In the Declarations and Variables section (equivalent to the `declare` statements in the source editor), do the following:
   a. Click **Add**. You see the Select Rule Declaration Arguments dialog.
   b. In the upper half of the dialog, select the type you want to use.
   c. If you choose an entity type, select an entity of that type, from the lower half of the dialog.
   d. Click **OK**.
   e. Your selection appears in the list. TIBCO BusinessEvents assigns an alias to the selected type. As desired, edit the alias.
   f. For primitives you can also add an expression to define the initial value.

Add more types as desired.

Bindings

6. Click the Bindings tab. In the Bindings section, click **Add**, select a data type, and click **OK**. A new entry appears in the Bindings list. Complete the entry as desired. You can do any of the following:

   — Replace the generated alias.

   — Define an expression to use as the initial value.

   — Specify a previously created domain model to limit what WebStudio users can enter. See Rule Template Views on page 273 for an explanation.

   Add more bindings as desired.

Pre-conditions and Action Context

7. Return to the Form or Source tab and use the bindings you created as follows:

   a. In the Pre-conditions section, write condition statements (in the TIBCO BusinessEvents rule language).

      Adding pre-conditions is similar to adding conditions for a rule, except that you can add non-conditional statements in a pre-condition. See Pre-conditions Section on page 270.

   b. In the Action Context section, (in the TIBCO BusinessEvents rule language) add action statements.

      Adding Action Context statements is similar to adding actions for a rule, except that only those action context statements used in business rules are actually executed.

8. Save the resource.

Adding a Rule Template View

First add a rule template. One rule template can be associated with multiple views. See Adding a Rule Template on page 274.

To Add a Rule Template View

1. Right-click the folder where you want to store 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.

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 Chapter 3, Element Refactoring Operations, on page 37 for more details.

4. Click **Finish**. You see the Rule Template View editor.

Configuration

5. Add or edit the description.
6. As needed, browse to 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.

Presentation

8. In the Presentation section, 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 click the Preview tab to see how the view will appear in WebStudio.
9. Save the resource.

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 on page 248 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):
	`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 on page 313.
Expression	If you specify a primitive you can also specify an expression. See Declarations and Variables Section on page 270 for an overview.
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` 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 (see Order of Evaluation of Rule Conditions in *TIBCO BusinessEvents Architect's Guide*.) See Pre-conditions Section on page 270 for an overview.

Property	Description
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, however, are actually taken. When you add an action context statement, you first choose one of the action types, then select an entity (create or modify action types) or rule function (call action type). See Action Context Section on page 271 for an overview.

Chapter 18 Functions

This chapter explains how to work with functions in TIBCO BusinessEvents.

Topics

Overview of 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 and Eclipse versions of the documentation, but not in PDF. To access the HTML version of documentation, open this page:

BE_HOME/doc/standard/html/index.htm

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 most generally useful catalog, the standard function catalog include the following categories:

- **BusinessWorks** functions are used in ActiveMatrix BusinessWorks integration projects. See Chapter 38, ActiveMatrix BusinessWorks Integration, on page 609.

- **Channel** functions return information about destinations, and can resume and suspend a destination.
- **Cluster functions** help with multi-engine functionality
  - **DataGrid** functions are for use with Cache object management. See Cache Related Functions on page 292.
- **Date** functions allow you to compare two DateTime values using only the date portion of the value.
- **DateTime** functions allow you to perform these date/time related tasks and more: add units of time to a DateTime, compare, retrieve, and format dates and times.
- **Engine** functions allow you to retrieve information about the engine, for example, available memory or the number of rules fired.
- **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.
- The **Exception** function enables you to create an exception.
- **File** functions provide various useful functions used when working with files.
- **HTTP** functions are used with the HTTP channel
- **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.
- **Math** functions allow you to perform advanced mathematical operations.
- **Number** functions allow you to perform type conversions from and to numbers and return the maximum and minimum values for a numeric type.
- **SOAP** functions enable you to work with SOAP messages sent through an HTTP channel.
- **String** functions allow you to perform comparisons, searches, conversions, and other operations with strings.
- **System** functions allow you to send messages to a debug log, retrieve global variables, retrieve system properties, and write data to a file.
  - **IO** functions allow the writing and closing of specific files.
- **Temporal** functions allow you to examine and perform calculations on values stored in a property’s history. For information about using temporal functions, see Temporal Functions and Their Parameters on page 286.
- **Util** functions category has one sub-category for working with HashMaps.

- **VRF** functions (that is, Virtual Rule Function functions) allow you to work with decision tables. See Virtual Rule Functions and VRF Catalog Functions on page 288 and see *TIBCO BusinessEvents Decision Manager User's Guide* for details.
- **XPath** functions allow you to evaluate XPath expressions.

CEP Pattern Functions

Pattern functions are used with the pattern matcher language for identifying patterns in events. See TIBCO BusinessEvents Event Stream Processing add-on documentation 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 add-on documentation for details.

Communication Functions

Communication functions provide a set of catalog functions that enables TCP communication. You can create a local TCP server and a TCP client so that TIBCO BusinessEvents can communicate with data sources not otherwise available through channels, using TCP. See Communicating with Other Sources using TCP on page 64 for details.

RDBMS 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 Decision Manager add on, for authentication.

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 — Allow you to create a simple event or concept instance.
- Time events — Allow you to create and schedule a time event. See Chapter 10, Time Events and Scheduler Functions, on page 165.

- Rule functions — Allow you to invoke a rule function. See Chapter 16, Rules and Rule Functions, on page 241.

The Ontology Functions area uses the same folder structure as the project (or rather, a subset of that structure).

Custom Functions

You can also create custom functions. Custom functions appear in the Custom Function catalog. For information about custom functions, see Adding Custom Functions on page 301.

Ontology Functions

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.
- When TIBCO BusinessEvents Views is used additional ontology functions are available for each entity type. See TIBCO BusinessEvents Views documentatoin for details.

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 customer who wish to continue using them. To make them visible in the Catalog Functions view, do the following:

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

Tool Tips

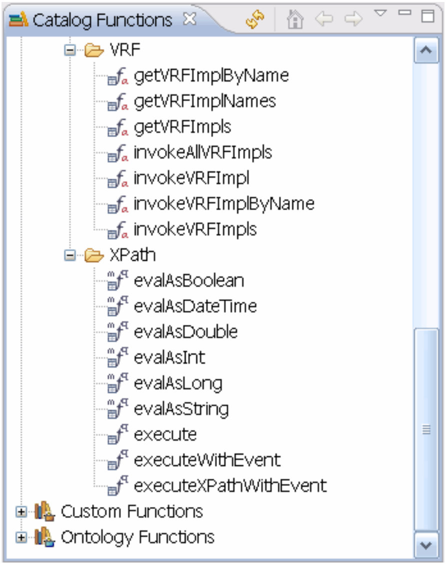
When you float the curser over a function in the registry, TIBCO BusinessEvents displays the description and syntax in a tool tip next to the curser. Tooltips also form the online reference to the function catalogs. See *TIBCO BusinessEvents Functions Reference* in the HTML documentation.

You can create your own tool tips for custom functions. See Adding Custom Functions on page 301.

You can turn off the tool tip 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 (available in TIBCO BusinessEvents Event Stream Processing add-on). Such functions are decorated with small symbols that indicate useful information about use of the function. The decorations 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:



Action-Only Functions

These functions are for use in rule actions only. 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 *f*. 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 *f*, for example: evalAsString . For more on using mapper functions, see Using the Function Argument Mapper on page 256.

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.

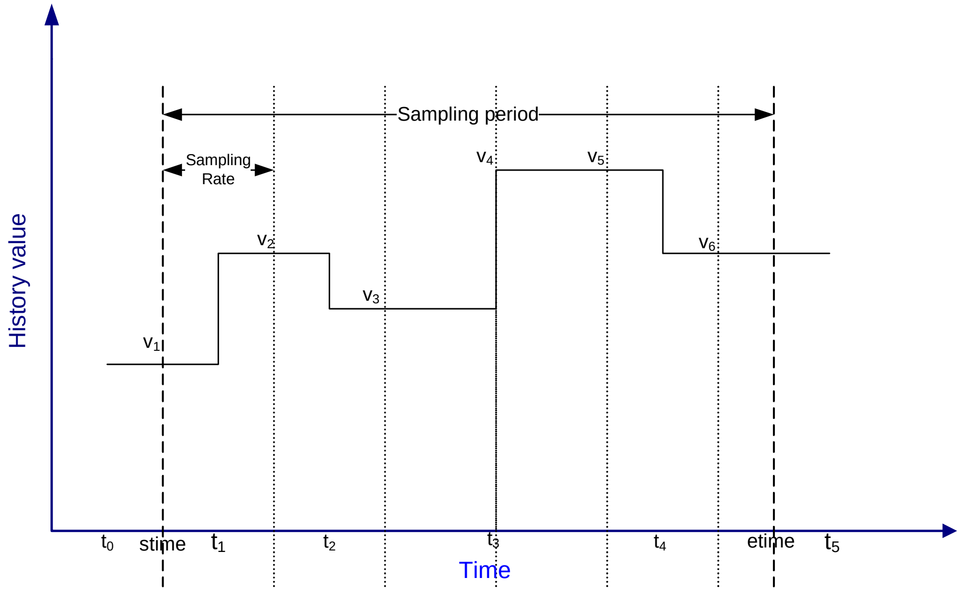
See documentation for TIBCO BusinessEvents Event Stream Processing add-on software for details on Pattern Matcher and query features.

Temporal Functions and Their Parameters

The set of Built-In > 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.

Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.

Figure 4 Temporal Functions Parameters



All temporal functions include these parameters, illustrated in Figure 4:

- **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 Built-in > 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.

Default Implementation 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()
```

Common arguments used in the above functions are described in Table 26, Common Arguments for VRF Functions, on page 291.

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.

To Add a Virtual Rule Function

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. Check the **Virtual** checkbox.

    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, check the Virtual check box.

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 do the following:

    a. Click **Add** to add resources that you will be 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.

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:

Table 26 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.
`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

TIBCO BusinessEvents Express This section relates to Cache OM functionality and is not relevant if you are using TIBCO BusinessEvents Express edition.

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.

The previous versions of these core functions (which began with C_) are deprecated. Migrate projects created in earlier releases to use the current names (see Migrating Core Coherence Functions on page 44).

Query functions are not available for the TIBCO BusinessEvents DataGrid provider. Similar functionality can be achieved using the TIBCO BusinessEvents Event Stream Processing add-on.

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's 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.

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's Guide* for details on use of locks.

Indexing Function (For Coherence Clusters Only)

`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. See Indexing for More Efficient Cache Queries on page 294 for both methods.

BQL Query-Related Functions

See To Query the Cache Using BQL Queries on page 296 for details

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

TIBCO BusinessEvents Express This section relates to Cache OM functionality and is not relevant if you are using TIBCO BusinessEvents Express edition.

When you use Oracle Coherence as the cache provider, you can index concept and event properties to make searches faster. You can index more than one of an entity type's properties.

The query optimization you set up is used by `C_Query*()` functions, and by snapshot queries, available in TIBCO BusinessEvents Event Stream Processing add-on.

You can create the indexes in the following ways.

To Create Indexes Using a Coherence Function

This method applies to the Coherence cache provider only. It is not the preferred method. It is recommended that you use the method explained in To Create an Index Using a Domain Object Override Setting on page 294.

You can create an ordered or unordered index using the following function in a startup rule function.

`C_Index(String` *cacheName*`, Object` *property*`, boolean` *isOrdered*`)`

where:

cacheName is a String returned by `C_CacheName()`.

property is the object returned by the appropriate C_*Datatype*`AtomGetter` functions, for example, `C_StringAtomGetter()`.

isOrdered is a Boolean: set to true to order the contents of the indexed information, and set to false if you want to use an unordered index.

For example:

```
String cacheName = Coherence.C_CacheName("/Customer");
Object getter = Coherence.Extractor.C_IntAtomGetter("age");
Coherence.C_Index(cacheName, getter, true);
```

To Create an Index Using a Domain Object Override Setting

This method applies both to the TIBCO and Coherence cache providers. You can create an unordered index in the project's Cluster Deployment Descriptor (CDD) using a domain object setting.

1. Open the project CDD in TIBCO BusinessEvents Studio and go to Cluster tab > 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, check the **Present in Index** checkbox for the property you want to index.
4. Save the CDD.

BQL Cache Queries

- **TIBCO BusinessEvents Express** This section relates to Cache OM functionality and is not relevant if you are using TIBCO BusinessEvents Express edition.
- This method of querying a cache requires use of the TIBCO BusinessEvents Event Stream Processing add-on product.
- The term BQL is an acronym indicating the TIBCO BusinessEvents Event Stream Processing query language.
- This is the only method that works with TIBCO BusinessEvents DataGrid.
- For the Coherence cache provider you can also use the methods explained in the section Oracle Coherence Cache Query Functions on page 299.

Enabling Explicit Tuple Format for TIBCO BusinessEvents DataGrid Fields

By default, entities and their properties in a TIBCO BusinessEvents DataGrid cluster are stored as blobs. 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.

You can still use query language queries (as explained in To Query the Cache Using BQL Queries on page 296) 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 TIBCO BusinessEvents DataGrid types:

  `integer, long, boolean, string, calendar, long, long`

- Concepts
  - — Concept property arrays are stored as blob columns
  - — Concept history is not supported
- Events
  - — Payload is a byte array or blob
- State Models
- Metrics (TIBCO BusinessEvents Views)

If an error occurs, for example due to an unsupported datatype, the item revers to its original blob format

To Enable Explicit Tuple Format

At the cluster level, set the following property to true in the CDD at the cluster level

```
be.engine.cluster.as.tuple.explicit=true
```

To Query the Cache Using BQL Queries

In order to execute the queries, you must first enable a dynamic query session.

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 preprocessor function, create a query string that selects entities from the cache. You can use any valid BQL query. See Example Rule Function for a TIBCO BusinessEvents DataGrid Cache on page 298 for an example.

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 Example Rule Function for a TIBCO BusinessEvents DataGrid Cache on page 298.

You can also do cache lookups using `@id` and `@extId` attributes. See Functions for Loading Entities to Rete from Cache and Backing Store on page 292.

Example Rule Function for a TIBCO BusinessEvents DataGrid Cache

The following example preprocessor rule function code shows the techniques used for querying a TIBCO BusinessEvents DataGrid 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");
    }
  }
}
```

Oracle Coherence Cache Query Functions

TIBCO BusinessEvents Express This section relates to Cache OM functionality and is not relevant if you are using TIBCO BusinessEvents Express edition.

These functions work only with a Coherence cluster. For information on using Oracle Coherence with TIBCO BusinessEvents, see Chapter 24, Cache OM and Cluster Configuration, on page 397.

You can also query a Coherence cache using the method explained in the section To Query the Cache Using BQL Queries on page 296.

Also see Indexing for More Efficient Cache Queries, page 294.

Constants, Extractors, and Filters Categories

The Coherence category has Constants and Extractors functions which are used in conjunction with functions in the Filters category.

Extractor functions return values for properties of different types.

Constants functions wrap constants so they can be used in filter functions. For example if a filter checks for `X = 10`, you would first wrap `10` using `C_IntConstant()`.

Functions in the Filters category enable you to use various criteria to identify a set of objects in the cache for a query. You can pass the filter to a query function.

The `C_RuleFunction()` function allows you to specify a rule function containing a custom filter condition.

Query Category

Query category functions operate only on the cache Unlike the `C_CacheLoad*()` functions, the query functions do not look in the backing store if objects are not found in the cache.

If a backing store is used Do not use query functions that delete or modify values if a backing store is used. Instead use a query to return the IDs of the entities you want to delete and use `Instance.deleteInstance()` or `Event.consumeEvent()` as needed.

```
C_CacheInvoke()
C_CacheOnlyMode_DeleteConcepts
C_CacheOnlyMode_DeleteEntities
C_CacheOnlyMode_QueryConcepts
```

```
C_CurrentContext
C_EntryHasNext
C_EntryIterator
C_EntryNextValue
C_KeyHasNext
C_KeyIterator
C_KeyNextValue
C_QueryAction
C_QueryAndLoadConcepts
C_QueryConcepts
C_QueryEvents
C_QueryEvents_Order
C_QueryIDs
```

Query functions take various actions for a specified entity or set of entities. For example, `C_CacheInvoke()` allows you to invoke a rule function for all matching entities in the cache.

For some functions you can specify the entities by passing a filter (from the Filters category).

Tool tips in the user interface (and reproduced in the *TIBCO BusinessEvents Functions Reference*) explain how to use the functions singly or in combination to achieve the desired results.

The `C_CacheOnlyMode*()` functions are for use with entities that use cache-only cache mode.

Adding Custom Functions

TIBCO BusinessEvents allows you to write your own custom functions in Java and add them to the function registry. When you have added the custom functions to TIBCO BusinessEvents Studio, their catalog appears automatically in the Catalog Functions view, along with the built-in function catalogs.

TIBCO BusinessEvents documentation does not contain detailed instructions for creating custom functions or programming in Java. It is assumed that you are already familiar with programming in Java, and have at a minimum already implemented a class and a static function.

See also:

- Restrictions on Use of Custom Functions on page 303
- Example Custom Function with Tooltips on page 304
- Structure of a Function Catalog on page 306

Task Summary

The steps below summarize the tasks required to integrate your own custom functions with TIBCO BusinessEvents Studio. An additional action is required to make the functions available at runtime.

1. Write a custom static function in Java and compile it. Add help content as a Java annotation, following the structure shown in Example Custom Function with Tooltips on page 304.
2. Create a file called `functions.catalog`, an XML file that makes it possible to access your custom functions from TIBCO BusinessEvents Studio. You can also include information for a tool tip for each function. See Structure of a Function Catalog on page 306 for details.
3. Create a `.jar` file that includes the static function class files for the implementation and the function catalog file.
4. Add the location of the JAR file or files (and any dependent JAR files) to the project properties. See Working with External Library and Custom Function Paths on page 8 for details.

The functions are then available for use in any rule editor. (It is not necessary to restart TIBCO BusinessEvents Studio before you can use your functions.)

The locations of the custom function jars are stored in the `.beproject` configuration file, which is located at the root level of the TIBCO BusinessEvents Studio project. You can check this configuration file into a version control system so that it can be shared with other project developers.

Note that the `.beproject` file contains other information as well, such as the location of project libraries and so on.

Restrictions on Use of Custom Functions

Note the following restrictions that pertain to using custom functions.

Static and Non-Static Functions

Custom functions must be written in Java and have public static modifiers.

As a workaround, encapsulate a non-static function in a static function and compile the encapsulating class to get the `.class` file.

Return Types

TIBCO BusinessEvents custom functions support the following return types:

- Java types supported are: `Object`, `String`, `Calendar` (which displays in TIBCO BusinessEvents as `DateTime`), `Integer`, `Long`, `Double`, `Boolean`, `int`, `long`, `double`, and `boolean` (but not `byte`, `short`, `float`, or `char`).
- Entity and its subtypes are supported, such as Concept or Event.
- Arrays are supported, with the exception of multidimensional arrays.

Name Overloading

The `functions.catalog` file makes functions available for use in TIBCO BusinessEvents Studio. The structure of the file requires each function within a class to have a unique name. Because of this structure, you cannot refer to an overloaded function in `functions.catalog`.

For example, the standard Java library has several `String.valueOf()` functions overloaded for each primitive type (`String.valueOf(int` *i*`)`, `String.valueOf(double` *d*`)` and so on). However, the TIBCO BusinessEvents standard function catalog has a separate function name for each data type: `valueOfBoolean()`, `valueOfDouble()`, `valueOfInt()`, and `valueOfLong()`.

See Structure of a Function Catalog on page 306, for more about the `functions.catalog` file.

Editing Custom Functions

If you need to edit custom functions, you must do so in a Java editor or project, and then re-export them to a custom function JAR. Add the JAR to the TIBCO BusinessEvents project, as explained in Adding Custom Functions on page 301.

Example Custom Function with Tooltips

The following very simple example of a custom function illustrates the required structure of a function called `add`, with the Java annotation that displays the tooltip for the function (beginning at `@FunctionJavadoc` in the example below).

```
package com.acme.functions.string;

import com.tibco.be.model.functions.FunctionJavaDoc;
import com.tibco.be.model.functions.FunctionParamDescriptor;
public class StringFunctions {

   public @FunctionJavaDoc (
      name="add",
      synopsis="Returns the concatenation of two strings.",
      signature="String add (String a, String b)",
      params={
         @FunctionParamDescriptor(
                     name="a",
                     type="String",
                     desc="param 1"),
         @FunctionParamDescriptor(
                     name="b",
                     type="String",
                     desc="param 2")},
      freturn = @FunctionParamDescriptor
               name="",
               type="String",
               desc="The concatenation of the parameter string"),

      version="5.1.0",
      see="",
      mapper=false,
      cautions="none",
      domain={"action","condition","query"},
      example="<br/>  " +
            "String  result = concatenate(\"a\",\"b\");" +
            "<br/><br/> Result is: result contains: \"ab\"."
   )
   static String add(String a, String b) {
      return a+b; // default impl.
   }
```

The sample above should provide enough guidance. A few additional notes follow.

synopsis Add the description of the function in the synopsis parameter.

domain Shows where the function can be used. This text is informational only. It does not appear in the tooltip. The code controlling where the function can be used (and the decorations that appear) is in the function catalog. See `<isActionOnly>`, `<isValidInQuery>`, `<isValidInBUI>` in Table 27, Function Catalog Elements, on page 307.

version Specify the version of TIBCO BusinessEvents that this function supports. Informational only.

see Provide a URL for additional information.

mapper Set to true if the function uses the XSLT Mapper feature. The Mapper decoration is added if the function catalog has a <mapper> element. In the function catalog file is more detailed code to support use of functions in the XSLT Mapper.

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 Table 27 to map properly to the schema.

- 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 a 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

Table 27 lists and describes the elements used in the function catalog.

Table 27 Function Catalog Elements

Element Name	Sub-Elements	Description
`<catalog name="`*name*`">`		The root element. Attribute: `name="`*name*`"` where *name* is a name you provide for this functions catalog. `Example: <catalog name="custom">`
`<category>`		This is a sub-element 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 executes asynchronously. Set to `false` if the function executes synchronously.
	`<helpurl>`	Not used in this release.

Table 27 Function Catalog Elements (Cont'd)

Element Name	Sub-Elements	Description
	`<args>`	A comma separated list of descriptive names for the 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 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. See Using the Reevaluate Element on page 308 for details. Valid values: `true`, `false`.

Using the Reevaluate Element

The 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 then:

- TIBCO BusinessEvents does not memorize the result of the evaluation of the condition that contains this function.
- If any of the conditions is re-evaluated, then this function is also re-evaluated.

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 re-evaluated, then `currentTimeMillis()` is also re-evaluated.

If set to false then:

- TIBCO BusinessEvents calls the function during the first evaluation and stores the result is stored and used for subsequent condition evaluations.
- TIBCO BusinessEvents Studio re-evaluates the condition only if another part of the same condition changes.

In the above stock price example, if `<reevaluate>` were set to false, then the condition would be re-evaluated only if `stock.time` changes.

Chapter 19 Rule Language Grammar

This chapter describes the grammar for TIBCO BusinessEvents rules.

Topics

Rule Language Basics

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

Comment rules as shown:

`/*` *text* `*/` The text from “/*” to “*/” is ignored

`//` *text* The text from “//” to the end of the line is ignored

Separators

The following tokens are used for separators:

; 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 (_).

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 on page 318 and Literals on page 315.

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.

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`

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

You can use local variables of the following types in rule actions and rule functions:

- Primitives
- Concepts
- Events

Primitive Arrays

You can also use primitive arrays, which are fixed length. 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

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: `"POQSTN3"` `"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

You can represent characters in literals using these escape sequences:

Table 28 Escape Sequences

Character	Escape Sequence
backspace	`\b`
horizontal tab	`\t`
linefeed	`\n`
form feed	`\f`
carriage return	`\r`
double quote	`\"`
single quote	`\'`
backslash	`\\`

Operators

The language defines the following operators. Operators in this list that are also used in the Java language work the same as the Java operators:

Table 29 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.)

Table 29 Operators in the TIBCO BusinessEvents Rule Language (Cont'd)

Operator	Notes
`&& || !`	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"`
`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

Keywords and Other Reserved Words

Do not 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
alias
all
and
as
asc
assert
attribute
avg
backwardChain
between
body
boolean
break
by
byte
case
catch
char
class
Concept
CONDITION
const
ContainedConcept
continue
count
date
DateTime
days
dead
declare
default
delete
desc
distinct
do
double
emit
else
entity
enum
Event
except
exists
extends
false
final
finally
first
float
for
forwardChain
from
goto
group
having
hours
if
implements
import
in
instanceof
int
interface
intersect
last
is_defined
is_undefined
key
last
latest
like
limit
lock
long
maintain
max
milliseconds
min
minutes
mod
moveto
native
new
not
null
object
offset
or
order
order
pending_count
package
policy
priority
private
protected
public
purge
QUERY
rank
requeue
return
rule
rulefunction
scope
seconds
select
short
SimpleEvent
sliding
static
strictfp
String
sum
super
switch
synchronized
then
this
throw
throws
time
TimeEvent
timestamp
transient
true
try
tumbling
undefined
union
unique
using
validity
virtual
void
volatile
when
where
while
```

Attributes

TIBCO BusinessEvents provides attributes that you can use in rules to access information of various kinds. The attributes are listed in Table 30. Use the @ operator to access attributes.

Table 30 Attributes

Entity	Attributes	Type	Returns
SimpleEvent	@id	long	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	long	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	long	The time event's unique internal ID.
	@closure	string	A string that was specified when 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.)

Table 30 Attributes (Cont'd)

Entity	Attributes	Type	Returns
Advisory Event	`id`	`long`	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	long	The concept instance's unique internal ID.
	@extId	string	The concept instance's unique external ID.
ContainedConcept	@id	long	The contained concept instance's unique internal ID.
	@extId	string	The contained concept instance's unique external ID.
	@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.

The internal ID is automatically generated by TIBCO BusinessEvents. You cannot set it.

Working with Concept and Event Properties

This section describes how to access concept properties and event properties using the TIBCO BusinessEvents language.

Accessing a Concept Property Atom

This is the syntax for accessing a concept property atom:

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;
```

If the history size is 0, TIBCO BusinessEvents does not record a timestamp.

Get and Set PropertyAtom Value With User-Specified Time

You can get and set PropertyAtom values as follows:

- You can 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 and *propertyAtomName* is the name of the PropertyAtom, and *time* is the time from which you want to retrieve the value.

- You can 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.

Working with Concept Property Arrays

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.

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'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.

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*`);`

When history is not tracked

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 reason is as follows. 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.

Event Property

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.

Exception Handling

The TIBCO BusinessEvents rule Language includes try/catch/finally blocks and has an Exception type. The `try/catch/finally` blocks behave like their same-name Java counterparts.

Advisory Events 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 resources. See Chapter 11, Advisory Events, on page 177.

This section describes the `try/catch/finally` commands.

Syntax

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

This section provides some examples to demonstrate use of exception handling.

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;
```

Flow Control

The TIBCO BusinessEvents rule Language includes commands to perform conditional branching and iteration loops. This section describes these commands.

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, executing a code block until the condition you specify is false.

Syntax:

```
for(initialization; continue condition; incrementor){
code_block;
[break;]
[continue;]
}
```

`break` allows you to break out of the loop.

`continue` allows you to stop executing 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` ten 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;]
}
```

break allows you to break out of the loop.

continue allows you to stop executing the code block but continue the loop.

Chapter 20 Rule Language Datatypes

This chapter provides datatype conversion tables, information about operators and types, and information about how TIBCO BusinessEvents handles inconstancy problems with datatypes.

Topics

Concept Properties to XML Datatype Conversions

Table 31 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 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.

- 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 located in the TIBCO Rendezvous and TIBCO Enterprise Message Service documentation.

Compatibility of Operators with Types

Table 32, Operator Matrix, defines the compatibility of operators with types.

Table 32 Operator Matrix

		Right Side of Operator							
		str	int	lon	dou	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, eq, cmp	=, math, eq, cmp	=, math, eq, cmp			=, math, 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. TIBCO BusinessEvents 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.

Chapter 21 Mapping and Transforming Data

This chapter describes how to map variables in the scope of a rule or rule function to arguments of a function used in that rule or rule function.

See also Chapter 22, XPath Formula Builder, on page 373 for related information.

See Chapter 16, Rules and Rule Functions, on page 241 for information about working with rules and rule functions.

Topics

Overview of Mapping and Transformation

The Function Argument Mapper allows you to supply the data that a function expects as input.

For instructions on accessing the Function Argument Mapper in the rule editor see Using the Function Argument Mapper on page 256

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

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 this chapter. 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.)

Mapping and Transforming Data to Function Input

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 (!) button.

Most options in the Edit Statement dialog are straightforward. However, there are some complex scenarios that require multiple steps. Many of these situations are described in the section Examples of Mappings on page 352. You may also wish to refer to XSLT Statements on page 368 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 Chapter 22, XPath Formula Builder, on page 373. 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 on page 337 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 Repairing Incorrect Mappings on page 347 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 Repairing Incorrect Mappings on page 347 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 elements in each panel. Table 33 describes the buttons and right-click menu items available in the panels of the Input tab.

Table 33 Input tab toolbar buttons

Button	Right-Click Menu	Description
Scope Variables Panel		
→=		Coercions. 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 datatype `any` can be coerced into a specific datatype.
?		Type Documentation. Allows you to specify or view documentation for schema elements.
	Expand	This menu item has two sub-menus: 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 sub-elements 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 sub-elements.
	Delete	Deletes the selected element.
	Copy	Copies the selected element. The element can be later pasted.
Function Panel		
☰		Shows or hides the mapping formulas for the input elements.

Table 33 Input tab toolbar buttons

Button	Right-Click Menu	Description
⇧		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 on page 368 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 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 on page 368 for more information about XSLT statements.
✚		Add Child. Adds a statement for a child element to the currently selected element.

Table 33 Input tab toolbar buttons

Button	Right-Click Menu	Description
✓		Mapper Check and Repair. 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. See Repairing Incorrect Mappings on page 347 for more information.
!		Edit Statement. Allows you to modify an XSLT statement for the element. See XSLT Statements on page 368 for more information about XSLT statements.
✎	Edit	XPath Formula Builder. Invokes the XPath formula builder. You can use this editor to create an XPath statement for this input element. See Chapter 22, XPath Formula Builder, on page 373 for more information about XPath and the XPath formula builder.
	Expand	This menu item has three sub-menus: Content, Errors, and All. Expand > Content expands the current element so that all sub-elements that have a mapping or expression are visible. Expand > Errors expands the current element so that all sub-elements that have an error in their expression are visible. Expand > All expands all sub-elements 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 sub-elements.
	Statement	This menu item contains shortcuts that allow you to easily add the desired XSLT statement(s) with one menu item instead of adding the statement(s) with the Insert button. See Statement Menu Options on page 348 for a description of the sub-items 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 be later pasted to a new location.

Table 33 Input tab toolbar buttons

Button	Right-Click Menu	Description
	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. Table 34 describes the icons used for schema items.

You can use the Type Documentation button to obtain any available documentation on any node in the Scope Variables or function input schema trees.

Table 34 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.

Table 34 Icons for schema items

Icon	Description
▦	Simple Date or Time. This can be any of the following datatypes: • Time • Date • Date & Time • Duration • Day • Month • Year • Month & Year • Day & Month
0|	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.

Table 35 describes the additional qualifiers that appear next to the name of schema items.

Table 35 Additional icons for hints

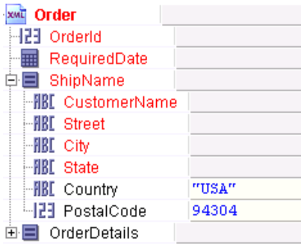
Qualifier	Scope Variables or Hint	Statement
	No qualifier indicates the element is required.	N/A

Table 35 Additional icons for hints

Qualifier	Scope Variables or Hint	Statement
?	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 or more times.	N/A
⧄	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.



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 Chapter 22, XPath Formula Builder, on page 373.

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 10th, 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 encouraged to always use 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.

Table 36 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.

Table 36 Datatype validation

Data Type	Validation
`boolean`	`xsd:boolean`
`double`	`xsd:double`
`string`	`xsd:string`
`dateTime`	`xsd:dateTime`
`long`	`xsd:long`
`int`	`xsd:int`

Repairing 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 will attempt 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.

If you want to return to the original expected Function input and remove all of the currently specified mappings, perform the following:

1. Delete the root element of the function's input by selecting it and clicking the Delete button.
2. Click the Mapper Check and Repair (☑) button.
3. In the Mapper Check and Repair dialog Fix column, select the checkboxes for all items.
4. Click OK.

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.

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

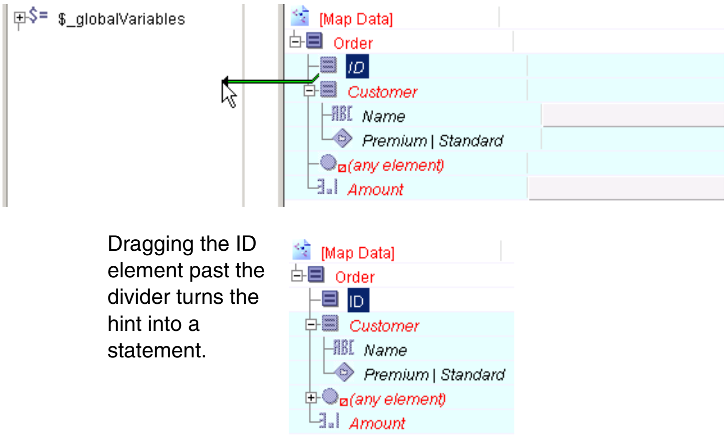
When you select an element in the function input schema and right-click, a menu appears. The Statement menu item contains several sub-items that are useful shortcuts for creating XSLT statements.

- 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 past the divider between the two areas of the Input tab changes a hint into an XSLT statement. Figure 5 illustrates dragging an element to the left.

Figure 5 Dragging to the left to change a hint to a statement



Before dragging, the ID element is a hint.

Dragging the ID element past the divider turns the hint into a statement.

This shortcut is useful in the following situations:

- When you have a complex element with no sub elements and no content.
- When you have a choice element, dragging to the left brings up the Mapping Wizard and allows you to choose a type for the element.
- When you have an element of type Any, dragging to the left brings up a dialog that allows you to 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

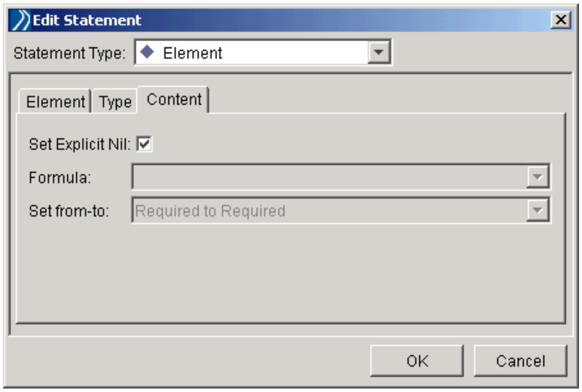
Some mappings require several steps to achieve the desired results. This section describes some complicated mapping scenarios and how to achieve the desired mappings using the tools available.

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. To set an input element explicitly to nil, perform the following:

1. Select the input element you wish to set to nil.
2. Click the Edit Statement (!)button on the Input tab toolbar.
3. Click the Content tab of the Edit Statement dialog.
4. Check the checkbox in the Set Explicit Nil field.



Edit Statement

Statement Type: ◆ Element

Set Explicit Nil: ☑

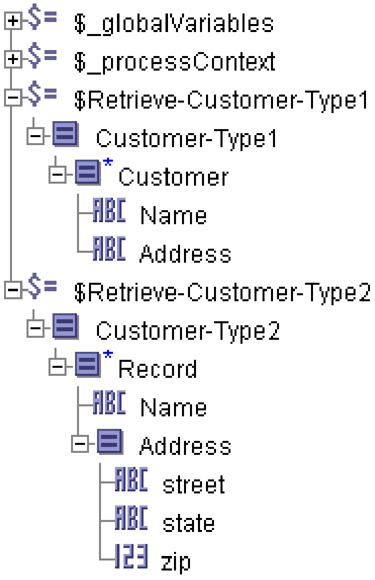
Formula:

Set from-to: Required to Required

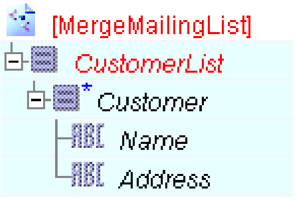
The element’s formula becomes blank and is not editable (because nil is the value of the element) and the explicit nil qualifier icon appears next to the statement ☒ .

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. In this example, the schemas are the following:



Multiple types of customers are retrieved, each having a different format for the address.



You wish to combine customers into a single repeating structure to create a mailing list.

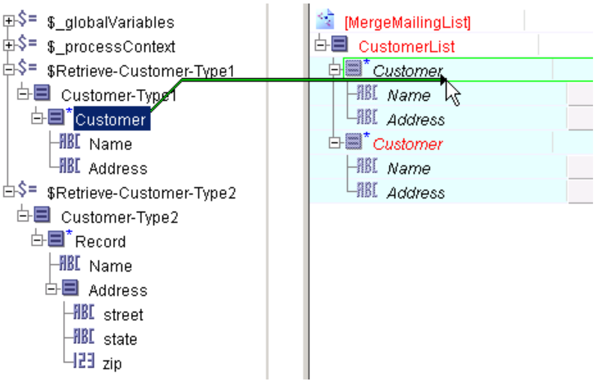
The following procedure describes how to map multiple elements into a single repeating element.

1. Select the repeating element in the Function panel, right-click, and select **Statement > Duplicate** from the menu.

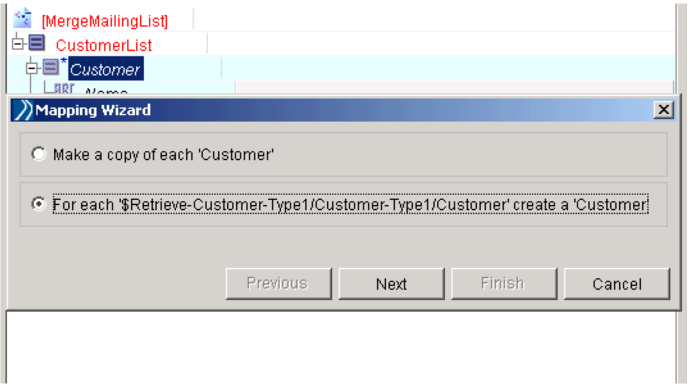
   Because you are creating two different formulas for mapping, you need two copies of the repeating element, one for each format. The resulting output contains only one repeating customer element, but the two copies in the Function panel make it simpler to perform two different mappings.

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**.



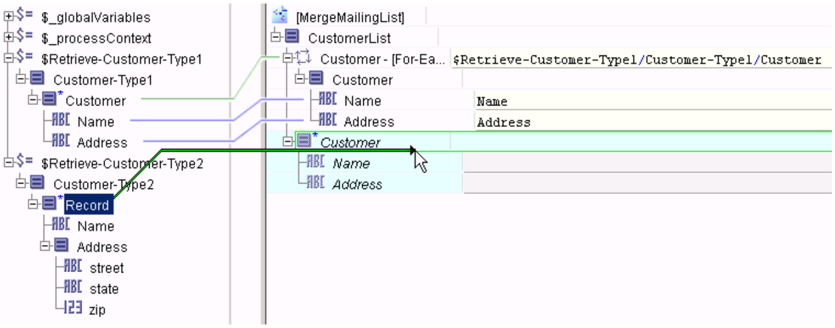
○ Make a copy of each 'Customer'

● 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.

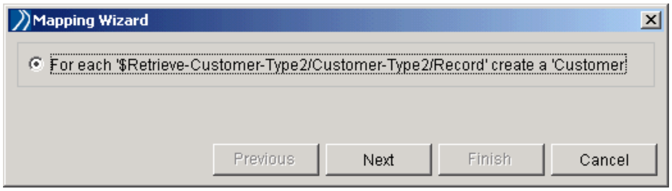
3. Map the other element from the Scope Variables to the second copy of the repeating element in the Function. For example, map

`$Retrieve-Customer-Type2/Record` to `MergeMailingList/CustomerList/Customer`.



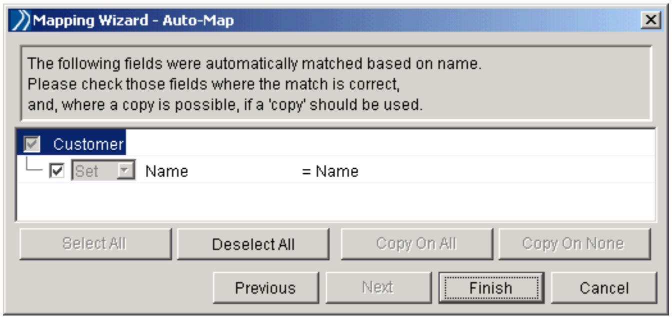
```
$Retrieve-Customer-Type1/Customer-Type1/Customer
Name
Address
```

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.

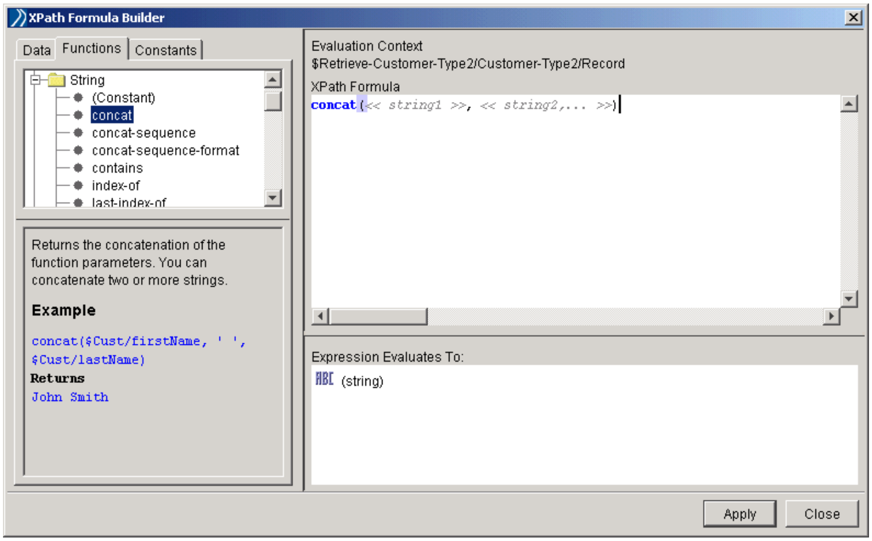


The following fields were automatically matched based on name.
Please check those fields where the match is correct,
and, where a copy is possible, if a 'copy' should be used.

☑ 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,... >>)
```

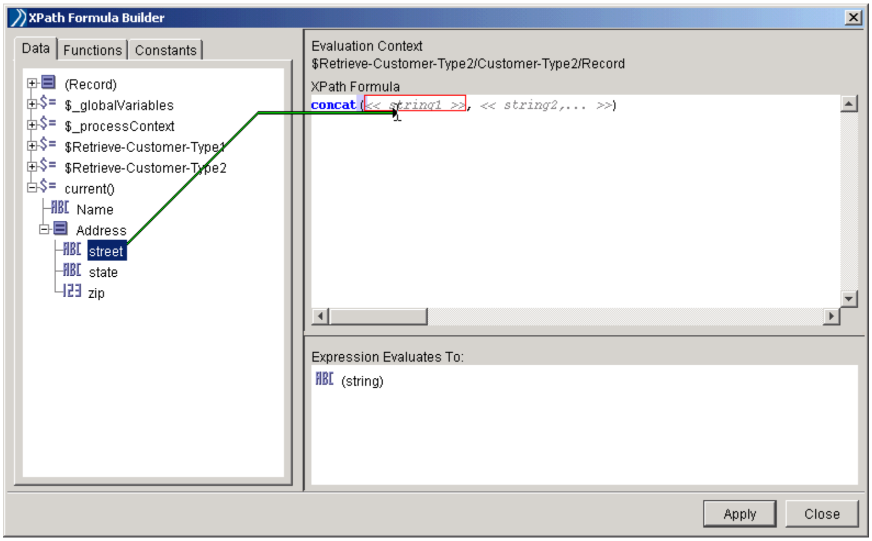
Returns the concatenation of the function parameters. You can concatenate two or more strings.

Example

```
concat($Cust/firstName, ' ',
$Cust/lastName)
Returns
John Smith
```

Expression Evaluates To:
ABC (string)

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

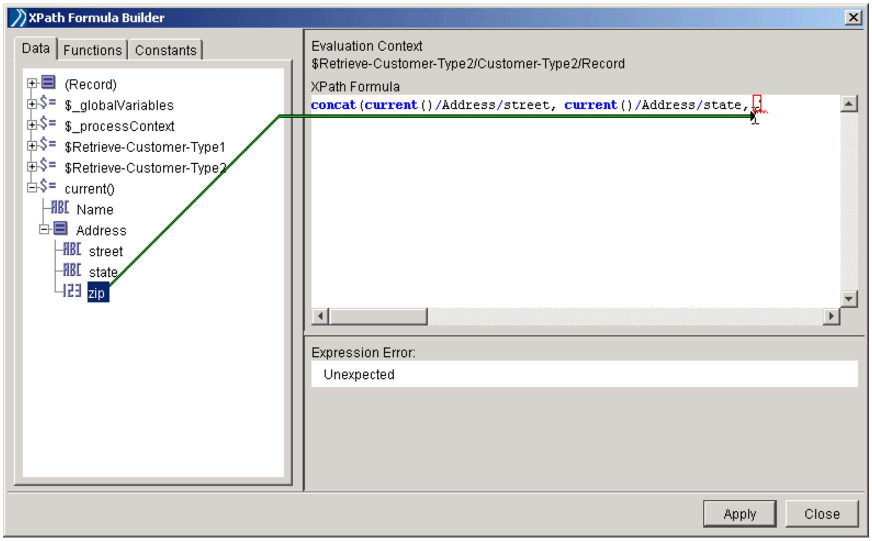
XPath Formula

```
concat(<< string1 >>, << string2,... >>)
```

Expression Evaluates To:
ABC (string)

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.



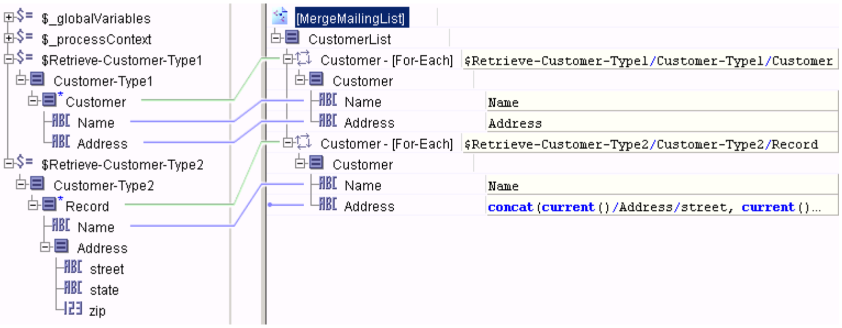
Evaluation Context
$Retrieve-Customer-Type2/Customer-Type2/Record

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:

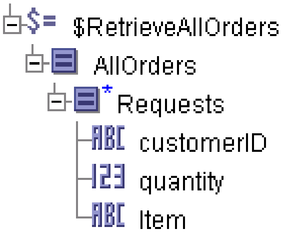


Customer - [For-Each]	`$Retrieve-Customer-Type1/Customer-Type1/Customer`
Name	`Name`
Address	`Address`
Customer - [For-Each]	`$Retrieve-Customer-Type2/Customer-Type2/Record`
Name	`Name`
Address	`concat(current()/Address/street, current()...`

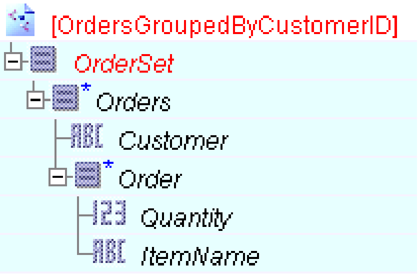
Converting a List Into a Grouped List

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 may want to organize that list so that you can group the orders placed by each customer.

In this example, the schemas are the following:



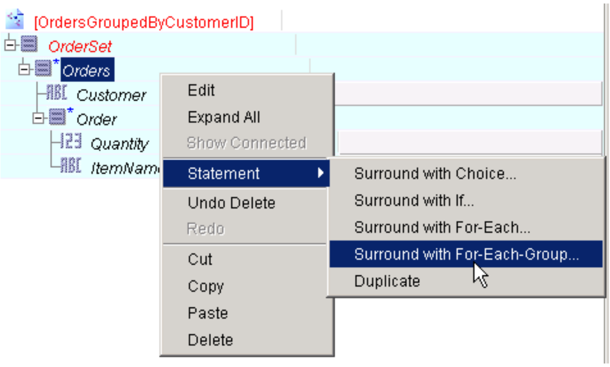
Flat list of orders placed by all customers. Each item in the repeating element Requests lists an order. The same customer can have mutliple orders in this list.



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.

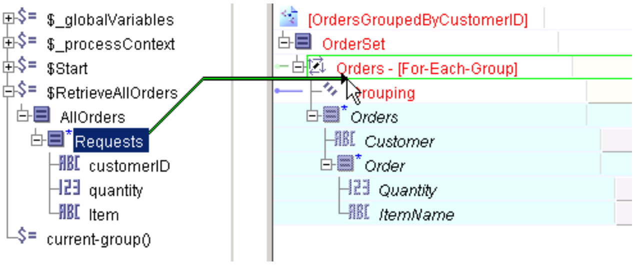
The following procedure describes how to map the flat list of orders into a list grouped by customer ID.

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.

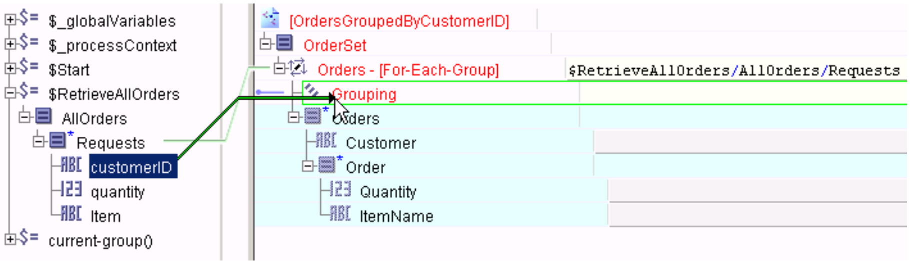


Adding the Grouping statement creates the `$=current-group()` element in the Scope Variables area. The Grouping statement creates the list grouped by the desired element, and the `current-group()` function allows you to access the items in the `Requests` repeating element that correspond to the group that is currently being processed.

2. Drag the repeating element from the Scope Variables area to the For-Each-Group statement.

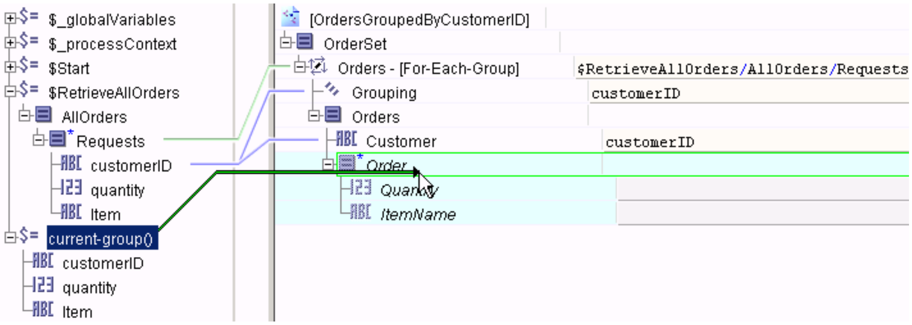


3. Drag the element you wish to group by from the Scope Variables area to the Grouping statement in the Function panel. In this example, customerID is the grouping element.



Orders - [For-Each-Group] `$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.

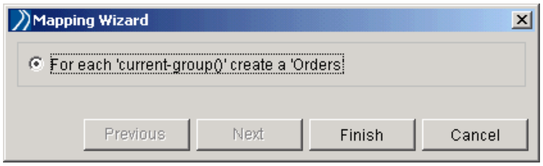


Orders - [For-Each-Group] `$RetrieveAllOrders/AllOrders/Requests`

Grouping `customerID`

Customer `customerID`

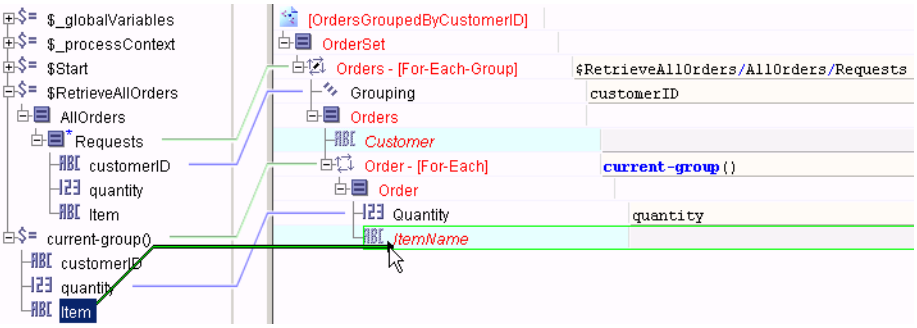
The default choice in the mapping wizard for this mapping is to create a For-Each. Choose this option in the mapping wizard.


Mapping Wizard

● For each 'current-group()' create a 'Orders'

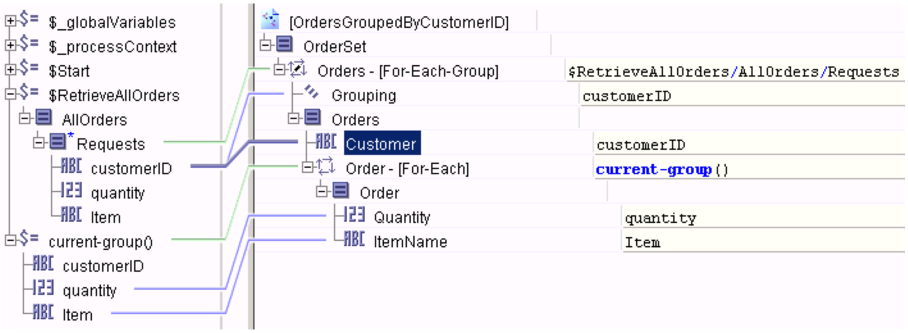
This creates an item in the `Order` list for each item in the current customer ID group that is being processed. The mapping wizard asks if you wish to map items with the same name in the current group and the orders group.

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`.



```
Orders - [For-Each-Group]   $RetrieveAllOrders/AllOrders/Requests
Grouping                    customerID
Order - [For-Each]          current-group()
Quantity                    quantity
```

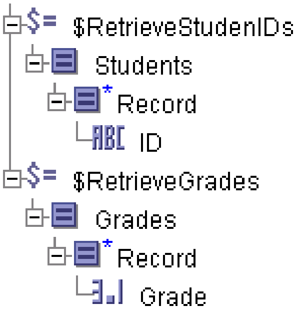
6. Map the customerID element in the Requests element into the Customer element in the Function panel.



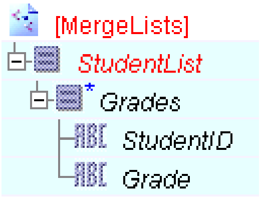
```
Orders - [For-Each-Group]   $RetrieveAllOrders/AllOrders/Requests
Grouping                    customerID
Customer                    customerID
Order - [For-Each]          current-group()
Quantity                    quantity
ItemName                    Item
```

Merging Two Corresponding Lists

You may need to merge two lists that have corresponding items. 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. In this example, the schemas are the following:



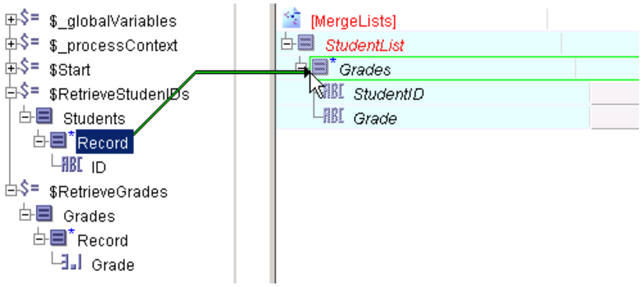
Two repeating lists contain corresponding student IDs and grades.



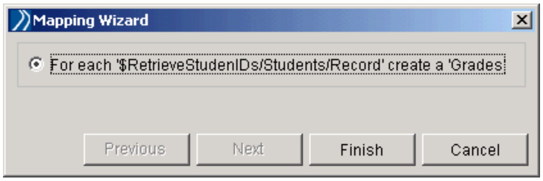
The resulting merged list will contain a both student ID and grade in a single repeating element.

The following procedure describes how to merge the two repeating elements containing corresponding data into one repeating element.

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.



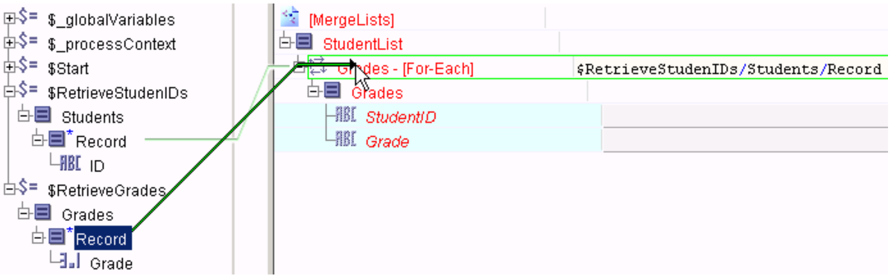
This brings up the mapping wizard with the default choice of creating a For-Each statement. Click **Finish** in the Mapping Wizard dialog to create the For-Each statement.



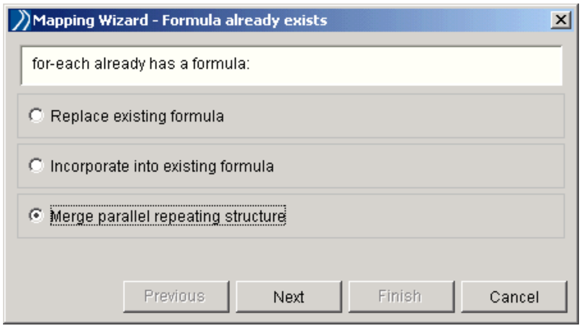
Mapping Wizard

● 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 chose an option. Choose the Merge parallel repeating structure option and click **Next**.



Mapping Wizard - Formula already exists

for-each already has a formula:

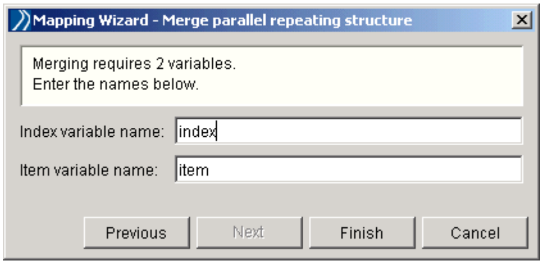
○ Replace existing formula

○ 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.



Mapping Wizard - Merge parallel repeating structure

Merging requires 2 variables.
Enter the names below.

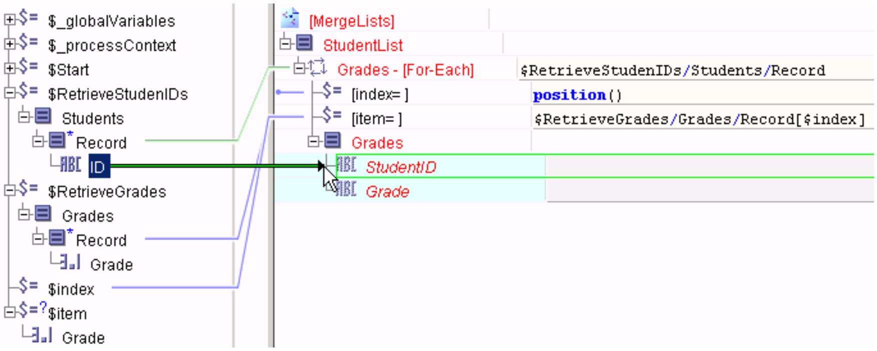
Index variable name: index

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.

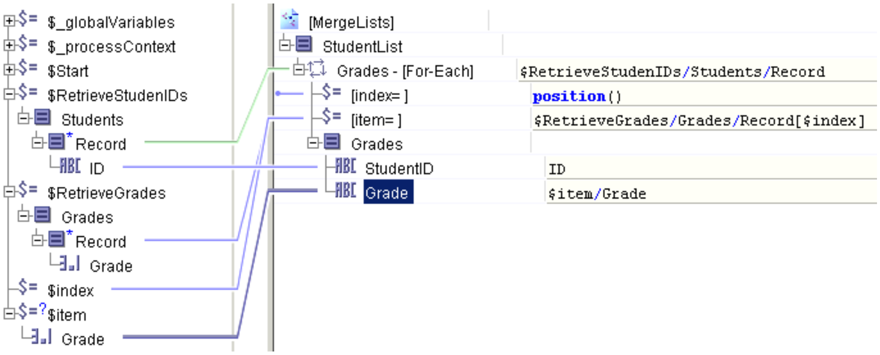
The `$=[index=]` element contains the XPath formula `position()` to set the element with the current position number of the list item being processed. The `$=[item=]` element contains a statement to retrieve the item in the second repeating element that corresponds to the position of the item in the first list that is currently being processed.

5. Map the `ID` element to the `StudentID` element in the function arguments.



```
Grades - [For-Each]    $RetrieveStudenIDs/Students/Record
[index=]               position()
[item=]                $RetrieveGrades/Grades/Record[$index]
```

6. Map the `$=item/Grade` element to the `Grade` element in the Function panel.



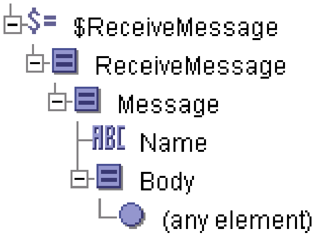
```
Grades - [For-Each]   $RetrieveStudenIDs/Students/Record
[index= ]             position()
[item= ]              $RetrieveGrades/Grades/Record[$index]
StudentID             ID
Grade                 $item/Grade
```

Coercions

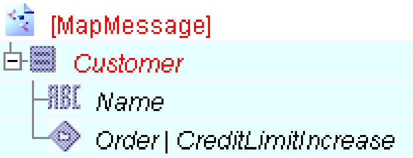
In some situations, the datatype of a Scope Variables element may be undefined. In these situations, you may know the datatype of the element, and you can coerce the element into a specific type. The Coercions button in the Input tab toolbar allows you to create and manage your coercions.

The following example illustrates a schema with an element defined as the "any element" datatype. 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.

In this example, the schemas are the following:



The incoming message can have a body of type any element.

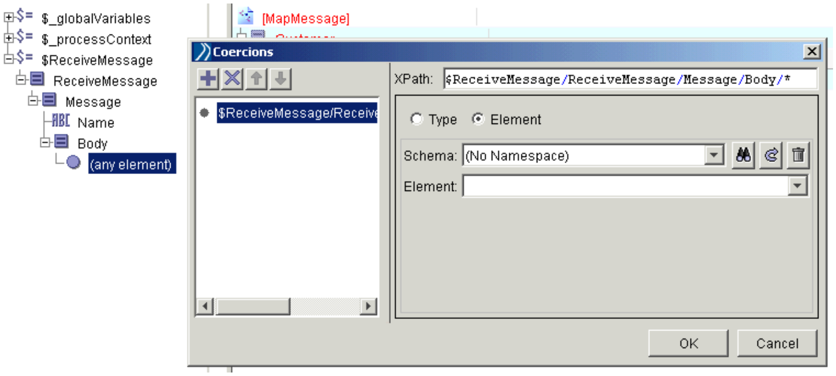


The function is expecting either an Order or a CreditLimitIncrease.

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.

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.

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.



XPath: $ReceiveMessage/ReceiveMessage/Message/Body/*

$ReceiveMessage/Receive

○ Type ● Element

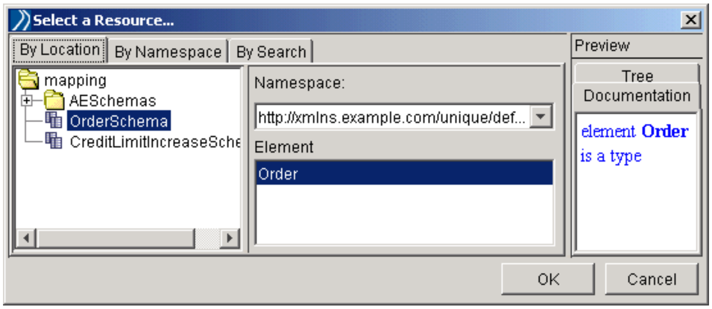
Schema: (No Namespace)

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

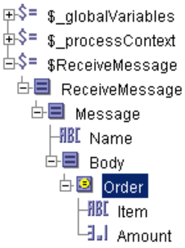


Namespace: http://xmlns.example.com/unique/def...

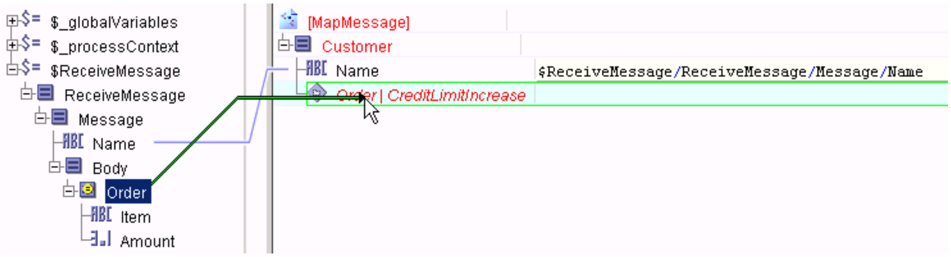
Element: Order

element **Order** is a type

Click OK to coerce the element into the datatype 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.

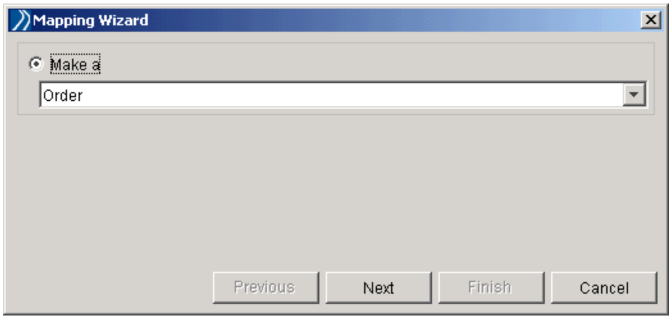


3. Map the Name element to the Name element in the Function panel. Then, map the coerced Order element to the choice element in the Function panel.



$ReceiveMessage/ReceiveMessage/Message/Name

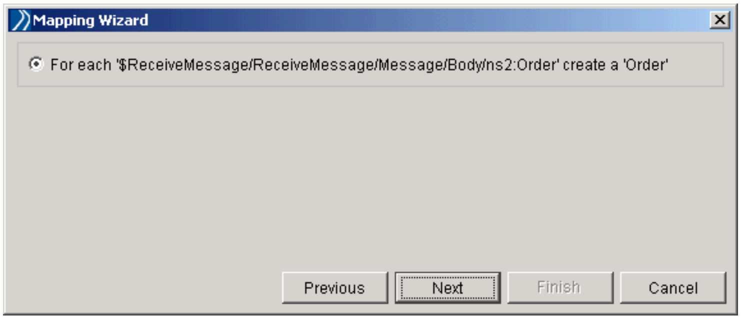
The Mapping Wizard dialog appears and asks if you wish to create an Order or a CreditLimitIncrease element. Select Order and click **Next**.



● Make a

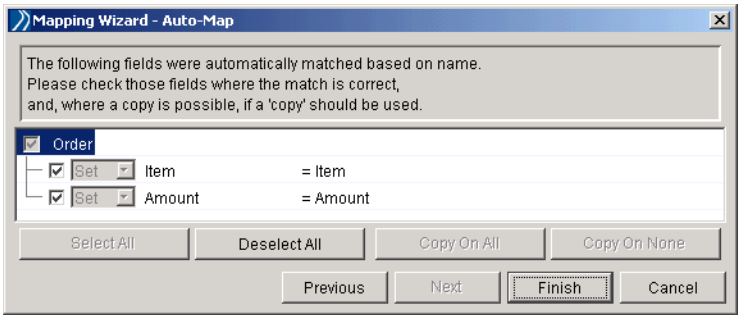
Order

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'

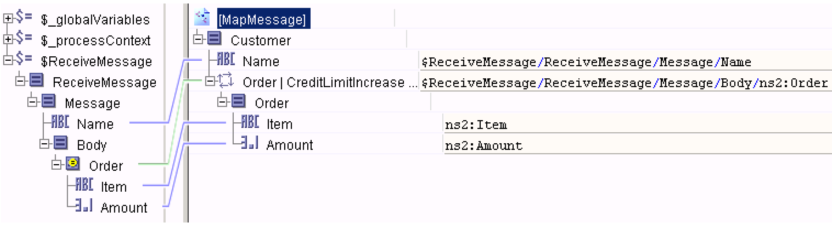
The Mapping Wizard then asks if you wish to automatically map elements with the same name. Click Finish to accept the default mappings.



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.



[MapMessage]	
Customer	
Name	$ReceiveMessage/ReceiveMessage/Message/Name
Order | CreditLimitIncrease ...	$ReceiveMessage/ReceiveMessage/Message/Body/ns2:Order
Order	
Item	ns2:Item
Amount	ns2:Amount

XSLT Statements

The following sections describe the XSLT statements you can add to your mapping. You can add or edit these statements by clicking the Edit Statement (!) button or 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 drop-down 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 will have 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 on page 357 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 will be 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"/>
```

Chapter 22 XPath Formula Builder

TIBCO BusinessEvents uses XPath in the XPath Formula Builder, available in the Function Argument Mapper tool (see Chapter 21, Mapping and Transforming Data, on page 335). You can use XPath, for example, when defining payloads for events TIBCO BusinessEvents also uses XPath as the language for defining conditions and transformations. This section covers the basics of XPath and its use in TIBCO BusinessEvents.

Topics

XPath Basics

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.

Table 37 describes the different areas of the XPath formula builder.

Table 37 XPath Formula Builder Reference

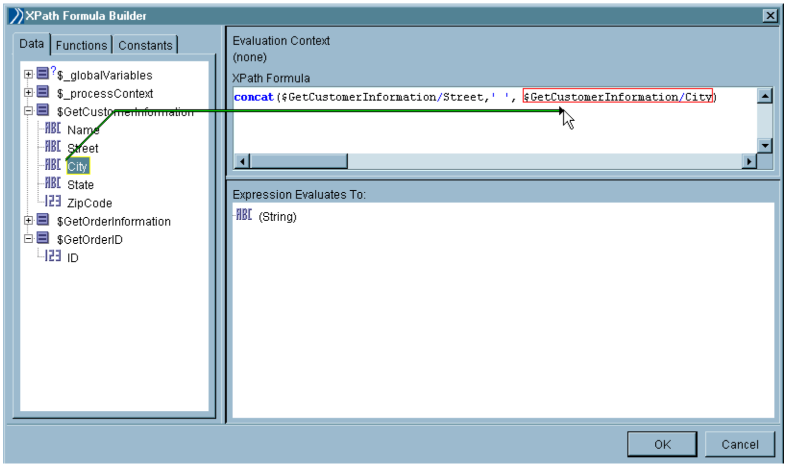
Element	Description
Data tab	Displays the Scope Variables schema tree. All elements in this tree are available to drag and drop into the XPath Formula field.
Functions tab	Displays the available XPath functions. These are categorized into groups and each function can be dragged from the function list 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.

Table 37 XPath Formula Builder Reference

Element	Description
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 Context field	Displays the evaluation context of the expression field that the editor was invoked from. See Evaluation Context on page 375 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.

Figure 6 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.

Figure 6 Creating an XPath formula



Evaluation Context
(none)

XPath Formula

```
concat($GetCustomerInformation/Street,' ', $GetCustomerInformation/City)
```

Expression Evaluates To:
(String)

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 that allow you to 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 `''`. Table 38 describes the alphabetic characters and their associated presentation in a date or time string.

Table 38 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
`W`	Week in month	3
`D`	Day in year	254
`d`	Day in month	28

Table 38 Formatting characters in date or time strings

Letter	Description	Example
`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
`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.

Table 39 illustrates some example date and time format patterns and the resulting string.

Table 39 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`
`"hh 'o''clock' a, zzzz"`	`9 o'clock AM, GMT-8:00`
`"K:mm a"`	`0:08 PM`
`"yyMMddHHmmssZ"`	`010704120856-700`

Chapter 23 Cluster Deployment Descriptor (CDD)

This short chapter provides general information about the Cluster Deployment Descriptor (CDD) and working with the CDD editor, which is used to configure CDD files.

Topics

Cluster Deployment Descriptor Overview

TIBCO BusinessEvents Express Content relating to Cache OM and backing store does not apply to TIBCO BusinessEvents Express edition.

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.

Because the deploy-time configuration settings for all processing units are in the CDD file, you do not have to rebuild the EAR file to make changes to deploy-time settings. However, as desired, you can use only the copy of the CDD file that is in the EAR, for tighter control and uniformity.

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

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

The CDD file you specify can be in the file system or in the EAR file. To specify a CDD located in an EAR file, provide its project path and name.

- For deployment using TIBCO Administrator, the CDD file in the EAR is used.
- The CDD file does not accept global variables as values.

You can configure multiple CDD files for a project for different purposes such as testing a design, trying out different object management options, dividing the work differently between agents and processing units (engines), and so on. However you use the same CDD file when deploying all the processing units for an application.

Maintaining the CDD File

The CDD is an XML file and you maintain it using the CDD editor.

Working with the CDD Editor

The CDD editor is organized into various tabs. Predefined fields make it easy to maintain common settings. Validation checks assist correct configuration. Settings for less common options are configured by adding properties to property sheets on the CDD editor’s tabs.

Editing the CDD After Deployment

Edit the CDD file in Studio and copy the output to the deployment location as needed.

Great care must be taken to ensure that the deployed CDD is edited correctly and that all CDD files are kept in sync for straightforward project management. If an emergency change is made to the deploy-time file, ensure that such changes are added back to the design-time version of the file and maintained in TIBCO BusinessEvents Studio (and put under source control), so that subsequent redeployment can be done without the need for manual changes.

Direct Editing is Discouraged

Although the CDD file can be edited directly, doing so is strongly discouraged. The file is complex, and must adhere to the schema. The schema itself may change in a release, and migration is handled using TIBCO BusinessEvents Studio.

Defining and Configuring the Cluster and Object Management Type

Of the object management (OM) types available, the most commonly used type for production systems is Cache OM. Cache OM is generally used with a backing store for persistence of data generated in the engine. See *TIBCO BusinessEvents Architect's Guide* for an overview of clusters and object management types.

When you configure Cache OM, you must first choose a cache provider, TIBCO or Oracle. The internal TIBCO cache provider, TIBCO BusinessEvents DataGrid, is the default choice. Or you can use a supported version of Oracle Coherence, for which you have a license that is appropriate for your usage. See Chapter 24, Cache OM and Cluster Configuration, on page 397.

For In Memory OM, no object management configuration is required because all objects are in memory and do not persist.

Configuring Management of Domain (Entity) Object Instances

If you choose Cache OM, you must also configure how to manage the domain (that is, entity) object instances. For example you can determine whether the instances are flushed from the Rete network after each RTC (as is generally recommended) or are kept in the cache.

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

See Chapter 28, Domain Objects Configuration, on page 455 and Chapter 26, Backing Store Configuration, on page 433.

Defining and Configuring Agent Classes and Processing Units

Cache OM uses multiple processing units, each running one or more agents. You define the agent classes and processing units in the CDD.

Agent Classes

Agent classes define the different sorts of agents you can deploy.

In Memory OM uses only inference agents, and each agent operates independently. With Cache OM, inference agents share the cache data.

For inference agent classes, you distribute a project’s resources among the agent classes to define the specific work each agent will do.

The following types of agent classes require the use of Cache OM:

- Cache
- Query, used in TIBCO BusinessEvents Event Stream Processing
- Process, used in TIBCO BusinessEvents Process Orchestration
- Dashboard, used in TIBCO BusinessEvents Views
- Monitoring & Management, used by MM server only. (See Chapter 3, Basic MM Configuration of *TIBCO BusinessEvents Administration*.)

Collections Tab

At the Collections tab, you can (optionally) group rules, rule functions, and destinations into collections so that they can be easily assigned to agent classes, to simplify agent class configuration.

The Collections tab is also where you define log configurations, which are assigned to processing units (engines).

Startup and Shutdown Rule Functions At the Collections tab, put rule functions for use at start up (startup rule functions) into different groups from those used at shut down (shutdown rule functions) and arrange them in the order in which they are to be executed. Then you can select them appropriately at the agent classes tab.

See Configuring Agent Classes (All OM Types), page 474 and Configuring Collections of Rules, Rule Functions, and Destinations, page 470.

Processing Units

Processing units deploy as engines within which the agents run. In the Processing Units tab, you define which agents to include in the processing unit, and which logging configuration to use. Depending on the OM, you also configure some additional settings.

See Configuring Processing Units (All OM Types), page 492.

For deployment using the TIBCO BusinessEvents Monitoring and Management component, processing units configured in the CDD editor are referenced in the Site Topology editor, where they are associated with physical resources. (See Chapter 3, Basic MM Configuration in *TIBCO BusinessEvents Administration*.)

How CDD Settings Apply at Runtime

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

Using Properties at Different Levels

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

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

Cluster level Applies to all *AgentClassName* agents in the cluster.

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

Agent class level Applies to any *AgentClassName* agent, used in any processing unit.

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

Only one value for a property is used when a processing unit is deployed. The order of precedence is explained in the section, Order of Precedence at Runtime.

Order of Precedence at Runtime

This section also appears in TIBCO Administrator and is reproduced here for your information, because order of precedence at run time can affect decisions made at design time.

The order of precedence at run time, from highest priority to lowest, is as follows:

1. Command-line arguments at engine startup.
2. Properties set in property files specified at the command line.
3. Properties in the deployed TRA file.
4. CDD file, processing unit level (for the current PU).
   a. Properties
   b. Settings

5. CDD file, agent class level (for agents listed in the current PU settings, prioritized in reverse order of that list):

   a. Properties

   b. Settings

6. CDD file, cluster level:

   a. Properties

   b. Message encoding

   c. Settings

EAR file properties (such as global variable overrides)

- Global variables set in the CDD file are ignored if you deploy using TIBCO Administrator. They are overridden by variables set in TIBCO BusinessEvents Monitoring and Management.
- Note that the TRA files should be used only for system-level settings that must be read before the JVM starts. All other properties should be in the CDD.

Setting Global Variables in the CDD File (for Command Line Startup)

Global variables are added to a project and their value is set using the TIBCO BusinessEvents Studio Global Variables editor, as explained in Working with Global Variables on page 14.

You can also set global variable values in the CDD file. This can sometimes be useful if you will start the engine at the command line. It is not done for other purposes

Global variables set in TIBCO Administrator or TIBCO BusinessEvents Monitoring and Management (BEMM) override those set in the CDD If you will deploy using TIBCO Administrator or BEMM, set global variable values using the features provided in TIBCO Administrator or MM (whichever you will use), not in the CDD.

To SetGlobal Variables in the CDD

Add properties using this format:

`tibco.clientVar.`*GVName = value*

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

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

Add such properties at the appropriate level in the CDD, depending on the desired scope: cluster, processing unit, or agent class.

Adding a Cluster Deployment Descriptor (CDD)

When you add a CDD, you select an object management type (known as the OM type). The OM type determines many of the configuration options available in the CDD editor. For an introduction to this topic, see Chapter 6, Object Management Types in *TIBCO BusinessEvents Architect's Guide*.

Specifying and Changing the Object Management Type If you choose the OM type in the wizard, a template for that OM type is used. For example, if you choose Cache, then a cache agent with default values is created for you. If you choose the OM type in the editor (after you finish the wizard), you add the necessary settings yourself and defaults are not provided. You can switch back and forth between object types without losing any configuration values, but only until you save. For example, if you change from cache to memory and save, the cache agent and its configuration are lost.

For all OM types, however, the General settings are configured in the same way, as documented in this section.

Names in the CDD 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/

To Add a CDD

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

For Deployment using TIBCO Administrator By default, TIBCO Administrator looks for a processing unit named default and a CDD file named default.

3. At the Object Management Selection page, select an object management type for the deployment, then click **Finish**. You see the CDD editor.

Check the Object Management node on the left to be sure you selected the correct OM type. Depending on the OM type you chose, you see one of the following:

- **Object Management: [Cache]**
- **Object Management: [In Memory]**

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

4. Select Cluster tab > **General** on the left. On the right, specify the following:

   — The cluster name and message encoding.

   — As desired, an author name, and any comment you wish to record. (Version and date are not editable.)

   See CDD Cluster Tab General Settings Reference on page 395 for details.

For deployment with TIBCO Administrator The message encoding specified in the CDD file General settings must match the TIBCO Administrator domain's message encoding.

5. Save the resource.

The next steps for each object management type are provided in the following sections:

- In Memory OM: (No cluster configuration.) Chapter 29, Agent and Processing Unit Configuration, on page 467.
- Cache OM: Chapter 24, Cache OM and Cluster Configuration, on page 397

Database Concepts Configuration

If you use database concepts, available in the TIBCO BusinessEvents Data Modeling add-on product, see *TIBCO BusinessEvents Data Modeling Developer's Guide* for details about configuring the Database Concepts node in the CDD.

Dashboard Agent Configuration

If you use the TIBCO BusinessEvents Views add-on, see *TIBCO BusinessEvents Views Developer's Guide* for details about configuring dashboard agents.

CDD Cluster Tab General Settings Reference

For the related procedure, see Adding a Cluster Deployment Descriptor (CDD) on page 393.

Table 40 CDD Cluster Tab General Settings

Property	Notes
Cluster Name	Specifies the name of the cache cluster. If TIBCO BusinessEvents DataGrid is the cluster provider, any spaces in the name are converted to underbar characters internally. **Note** Do not use the name `$cluster`. It is a reserved name. Initially set to the CDD name.
Message Encoding	The encoding used in Rendezvous messages exchanged between TIBCO applications. **For deployment with TIBCO Administrator** The message encoding specified here must match the TIBCO Administrator domain's message encoding. Default is ISO8859-1 (which is also the default for TIBCO Administrator)
Author	The name of the author of this CDD, as desired. Initially set to the currently logged-on user name.
Comment	Any comments as desired. Comments persist across versions.
Version	View-only field to record the version of the CDD, for information only. You could, for example, check whether deployed CDDs are all using the same version.
Date	View-only field to record creation time of this version.

Chapter 24 Cache OM and Cluster Configuration

TIBCO BusinessEvents Express This chapter relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

Use this chapter if you picked Cache OM in the second page of the New Cluster Configuration wizard. This chapter explains how to configure cluster discovery and internal communications, and how to configure cache-related object management settings.

Topics

Cache Object Management Configuration

When you configure object management for Cache OM, you select the cache provider, configure a cache agent quorum, number of backup copies. If you have a backing store, you can also configure a limited cache and specify the cache size for each entity.

In addition to the basic reference tables (as noted in the procedure below) the following sections have additional specific guidelines:

- Configuring a Limited (or Unlimited) Cache on page 399
- Synchronous and Asynchronous Replication of Cache Objects on page 403
- Property for Cache Based Object Management on AIX on page 404
- Configuring Cluster Discovery and Internal Communication on page 405 and sections following

For backing store details, see Chapter 26, Backing Store Configuration, on page 433 for project configuration details and see Chapter 30, JDBC Backing Store Setup, on page 501 for details about setting up the backing store itself.

To Configure Cache OM Options

1. Add a CDD file or open the CDD file you added, as explained in Adding a Cluster Deployment Descriptor (CDD) on page 393.
2. Select the Cluster tab > Object Management node on the left and on the right, configure settings as explained in CDD Cluster Tab Cache OM Settings Reference on page 401.
3. Save the resource.

Configuring a Limited (or Unlimited) Cache

This procedure explains how to use the options available to configure a cache of a limited size. You can also use this procedure to set options for an unlimited cache. Also see Limited and Unlimited Cache Size in *TIBCO BusinessEvents Architect's Guide*.

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

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

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

To Configure Limited Cache and Object Table Cache Options

1. Select the CDD editor Cluster tab > Object Management node.
2. In the Entity Cache Size setting, enter the desired number of objects per entity type.
3. In the Object Table Cache Size, enter the desired number of objects (handles) in the object table cache. You cannot set this value differently for different object types. See The Role of the Object Table, in *TIBCO BusinessEvents Architect's Guide*, for more details about the object table.
4. Select the Cluster tab > Domain Objects > Defaults node. Check the Is Cache Limited checkbox to enable limited cache globally. (Or uncheck the checkbox to use an unlimited cache globally.)
5. Select the Domain Objects > Overrides node. Select an override entry (or add one as needed).

6. Set the Is Cache Limited checkbox for the selected object type in one of the following ways:

    — If limited cache is set at the default level, uncheck the overrides Is Cache Limited checkbox to use an unlimited cache for objects of this type.

    — If unlimited cache is set at the default level, check the Is Cache Limited checkbox to use a limited cache for objects of this type.

    See Configuring a Limited (or Unlimited) Cache on page 399 for important requirements. For reference documentation on the Is Cache Limited checkbox see Table 49, CDD Cluster Tab Domain Object Default Settings, on page 459.

Ensuring Multiple Clusters Do Not Conflict

With TIBCO BusinessEvents DataGrid clusters, use a different value for Cluster Name (in the Cluster tab, General node) and also use different discovery values.

With Coherence clusters, use a different value for cluster name.

CDD Cluster Tab Cache OM Settings Reference

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

For the related procedure, see Configuring Cluster Discovery and Internal Communication on page 405.

Table 41 CDD Cluster Tab Cache OM Settings (Sheet 1 of 2)

Property	Notes
Provider	Select the cache provider: TIBCO: Uses the provided TIBCO BusinessEvents DataGrid component. See Configuring the TIBCO BusinessEvents DataGrid Discover URL on page 407 and Configuring the TIBCO BusinessEvents DataGrid Listen URL on page 410 for more details. ORACLE: Uses a supported version of Oracle Coherence, for which you have a license that is appropriate for your usage. Requires additional configuration. See Enabling Use of Oracle Coherence as the Cache Provider on page 414 and Configuring Oracle Coherence Cluster Discovery on page 415 for details.
Cache Agent Quorum	Specifies a minimum number (quorum) of storage-enabled nodes that must be active in the cluster when the system starts up before the following occur: • Data is preloaded from the backing store, if a backing store is configured and preloading is configured (see Chapter 26, Backing Store Configuration, on page 433). • The other agents in the cluster become fully active. The property does not affect the running of the deployed application after startup (though a message is written to the log file if the number of cache agents running falls below the number specified in this property). As a guideline, set to the number of cache agents configured. Default is 1.

Table 41 CDD Cluster Tab Cache OM Settings (Sheet 2 of 2)

Property	Notes
Number of Backup Copies	The number of backup copies (also known as the backup count) specifies the number of members of the distributed cache service that hold the backup data for each unit of storage in the cache. Recommended values are 0, 1, or 2. Value of 0 means that in the case of abnormal termination, some portion of the data in the cache will be lost. Value of N means that if up to N cluster nodes terminate at once, the cache data will be preserved. A backup count of 1 means one server plus one backup are needed, that is, two cache agents (or storage enabled nodes if cache agents are not used). To maintain the partitioned cache of size M, the total memory usage in the cluster does not depend on the number of cluster nodes and will be in the order of M*(N+1). See Synchronous and Asynchronous Replication of Cache Objects on page 403 for details on replication behavior and options for TIBCO BusinessEvents DataGrid. Default is 1.
Entity Cache Size	Specifies the size of the limited cache, in number of cache entries for each object type. The setting is per processing unit. See Configuring a Limited (or Unlimited) Cache on page 399. Default is 10000 (entries per object type)
Object Table Cache Size	Specifies the maximum size of the object table cache, in number of entries. Used with limited cache only. See The Role of the Object Table in *TIBCO BusinessEvents Architect's Guide* for more details about the object table. Also see Configuring a Limited (or Unlimited) Cache on page 399. Default is 100000 entries

Synchronous and Asynchronous Replication of Cache Objects

When you set a backup count, you define the number of backup object copies to make, in addition to the primary cache object. Backup cache writes can be done synchronously or asynchronously. There is a difference between the behavior of TIBCO BusinessEvents DataGrid and for Oracle Coherence as regards replication.

TIBCO BusinessEvents DataGrid Uses Asynchronous Replication

TIBCO BusinessEvents DataGrid is set up to use asynchronous replication. There is no option to use synchronous replication. Asynchronous replication allows you to run tests using a single cache agent.

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

Oracle Coherence Uses Synchronous Replication

With Oracle Coherence as the cache provider, TIBCO BusinessEvents is set up to use synchronous replication. There is no option to use asynchronous replication.

With synchronous replication, when the inference agent writes to the cache, the cache provider makes a network call to another cache agent and makes a replica (or replicas) and then the call from the inference agent returns. This means making two cache puts for each cache operation. Therefore, synchronous replication is slower than asynchronous replication.

You must also ensure that the required number of cache agents is always up and running (depending on the backup count).

Property for Cache Based Object Management on AIX

When TIBCO BusinessEvents is installed on AIX and uses cache-based object management, you must add this property to all TRA files, and set the value to true.

```
java.net.preferIPv4Stack=true
```

Remember to set this property on all internal TIBCO BusinessEvents engines' TRA files too, such as in `be-mm.tra` for the TIBCO BusinessEvents Monitoring and Management (MM) server and the MM broker properties set in the MM CDD file. Add-on products also have engine TRA files you must update.

If you do not add this property, you see the following exception:

```
java.net.SocketException: The socket name is not available on this
system
```

Configuring Cluster Discovery and Internal Communication

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

This section has summary steps for both cache providers, and pointers to sections with more details.

To Configure a TIBCO BusinessEvents DataGrid Cluster (Metaspace)

An active LAN connection (device enabled and network cable plugged in) is required for TIBCO BusinessEvents DataGrid to work.

1. Add a CDD file or open the CDD file you added, as explained in Adding a Cluster Deployment Descriptor (CDD) on page 393.
2. Select the Cluster tab > Properties node on the left and on the right, add the following two properties, as needed:

   ```
   be.engine.cluster.as.discover.url
   be.engine.cluster.as.listen.url
   ```

   The properties can be omitted if you use PGM multicast with default values. See the following sections for details on configuring these properties:

   — Configuring the TIBCO BusinessEvents DataGrid Discover URL on page 407

   — Unicast (Well-Known Address) Cluster Member Discovery on page 408

   — Configuring the TIBCO BusinessEvents DataGrid Listen URL on page 410
3. If you use unicast (well-known address) discovery, and you use TIBCO BusinessEvents Monitoring and Management for monitoring and management, you must also do the following (in the to-be-monitored project CDD):

   a. Add the following property to the cluster properties sheet.

      `be.mm.cluster.as.listen.url` *MMHostIP: Port*

      Specify the IP of the computer hosting the MM server, and an unused port.

   b. Add the value of the `be.mm.cluster.as.listen.url` property to the list of addresses in the `be.engine.cluster.as.discover.url` property, which should be present at the cluster level (so the value is identical for all potential cluster members).

The discover URL for well-known address configuration uses the following format:

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

To Configure an Oracle Coherence Cluster

1. Before you begin ensure that you have met all prerequisite steps. See Enabling Use of Oracle Coherence as the Cache Provider on page 414
2. Add a CDD file or open the CDD file you added, as explained in Adding a Cluster Deployment Descriptor (CDD) on page 393.
3. Select the Cluster tab > Properties node on the left and on the right, add properties as explained in Configuring Oracle Coherence Cluster Discovery on page 415.
4. Add any other cluster level properties as needed. See Other Coherence Properties on page 423.

To Configure Cache-Related Object Management Settings

1. Open the CDD file you added, as explained in Adding a Cluster Deployment Descriptor (CDD) on page 393.
2. Select the Cluster tab > Object Management node on the left, and on the right configure settings as explained in CDD Cluster Tab Cache OM Settings Reference on page 401. Also see Synchronous and Asynchronous Replication of Cache Objects on page 403.

Configuring the TIBCO BusinessEvents DataGrid Discover URL

When a cluster starts up, and also when new members join a cluster, a discovery process enables the members to discover each other. The discover URL specifies how an engine (node) listens for discovery requests from nodes attempting to join the cluster.

After the discovery is complete, the members communicate internally using a listen URL (explained in Configuring the TIBCO BusinessEvents DataGrid Listen URL on page 410).

Two types of discovery are available:

- Multicast discovery (PGM)
- Unicast discovery (TCP), also known as "well-known address" discovery

Configuration for both discovery methods is explained below.

A TIBCO BusinessEvents DataGrid cluster is also known as a *metaspace*.

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

If No Other Cluster Members are Started

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

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

Multicast (PGM) Cluster Member Discovery

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

The discovery property is `be.engine.cluster.as.discover.url`. For multicast discovery, the value is a URL with the following format:

`tibpgm://`*`destinationPort`*`/`*`network`*`/`

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

Specify the parameters as follows.

Parameter	Notes
destinationPort	Specifies the destination port used by the PGM transport. Must be the same value on all machines in the cluster. Default value is `7888`.
network	Specifies the IP address of the interface to be used for sending multicast packets, and the multicast group address to be used. The format is as follows: *interface;multicast group address* The value for *interface* is unique to a node. It must also be the same in both the discovery and the listen URLs for a node. If there are multiple interfaces on one machine, specify the interface you want to use and do not rely on the default value. The value for *multicast group address* must be the same on all machines in the cluster. The default value for *interface* is the first available interface provided by the operating system hosts file for the machine. **Note** If the desired interface is not listed in the hosts file then PGM picks the first available interface in the file. (On most operating systems, this file is called the `/etc/hosts` file.) If the first interface is the loopback interface (127.0.0.1) then PGM fails to start. In this case you would see a stacktrace exception in the log file such as the following: `SYS_ERROR (multicast_error - (8) grp_iface not a valid multicast interface)` To resolve this issue, either modify the hosts file, or provide the desired interface explicitly in the *network* argument. The default value for *multicast group address* is the multicast group address `239.8.8.9`.

Unicast (Well-Known Address) Cluster Member Discovery

If you cannot or do not wish to use multicast discovery in your environment, then configure unicast discovery, also known as "well-known address" or WKA discovery. These "well-known addresses" enable a newly started node to discover existing membersUnicast discovery uses the TCP protocol.

The discovery property is `be.engine.cluster.as.discover.url`. For unicast discovery, the value is a semicolon-separated list comprising a sub-set of all the listen URLs (which are different for each PU), using this format:

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

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

For WKA discovery, make discover URL a cluster-level property and listen URL a PU-level property The discover URL property (`be.engine.cluster.as.discover.url`) must be present and configured identically for all potential cluster members. Therefore add this property at the cluster level of the CDD file. The listen URL property (`be.engine.cluster.as.listen.url`) must be present and configured differently for each possible cluster member. Therefore add this property at the PU level.

Configuring the TIBCO BusinessEvents DataGrid Listen URL

The listen URL is used for direct communication between the members of the metaspace. It is configured the same way for multicast and for unicast discovery (see Configuring the TIBCO BusinessEvents DataGrid Discover URL on page 407). The listen URL value must be different for each cluster member, so configure it at the PU level.

The listen URL uses this format:

tcp://*interface*:*port*[-*EndPort* |*]/

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

Parameter	Notes
interface	To specify a value, use the desired IP address. The value for *interface* must be the same in both the discovery and the listen URLs for a node. If there are multiple interfaces on one machine, specify the interface you want to use and do not rely on the default value. The default value for *interface* is the first available interface provided by the operating system for the machine.
port	To specify a single port use the port number in the listen URL, as shown in this example: tcp://*interface*:6000/ You can use an auto-incrementing feature, as explained in Auto-incrementing Within a Range of Ports, page 411. The default value is the first available port in the 50000+ range.

Multiple Nodes on One Machine

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

Auto-incrementing Within a Range of Ports

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

`tcp://`*interface*`:`*Port-EndPort*`/`

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

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

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

`tcp://`*interface*`:`*Port*`-*/`

CDD Cluster Tab TIBCO BusinessEvents DataGrid Properties Reference

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

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

Table 42 CDD Cluster Tab TIBCO BusinessEvents DataGrid Properties

Property	Notes
`be.engine.cluster.as.codec.explicit`	
	The codec explicit property specifies how a concept is stored in the TIBCO BusinessEvents DataGrid cache. The default value for the property is set to false.
`be.engine.cluster.as.discover.url`	
	The discover URL specifies how an engine (node) listens for discovery requests from nodes attempting to join the cluster. PGM protocol is supported for multicast discovery. TCP protocol is supported for unicast (well-known address) discovery. Configuration is different for multicast and unicast discovery. See Configuring the TIBCO BusinessEvents DataGrid Discover URL on page 407 for details. The default value for multicast equates to: `tibpgm://7888/;239.8.8.9/`

Table 42 CDD Cluster Tab TIBCO BusinessEvents DataGrid Properties (Cont'd)

Property	Notes

`be.engine.cluster.as.listen.url`	The listen URL is used for direct communication between the members of the metaspace after the discovery process. The listen URL uses this format:
	`tcp://`*interface*`:`*port*`/`
	You can also use an auto-incrementing feature by specifying a range as follows:
	`tcp://`*interface*`:`*Port*`-[` *toPort* | * `]/`
	The default value for *interface* is the first available interface provided by the operating system for the machine.
	See Configuring the TIBCO BusinessEvents DataGrid Listen URL on page 410 for details.
	The default value for port is the first available port in the 50000+ range.

`be.mm.cluster.as.listen.url`	If you use well-known-address discovery and TIBCO BusinessEvents Monitoring and Management (MM), you must also add this property at the cluster level of the to-be-monitored project's CDD file. Specify the value as the IP of the computer hosting the MM server, and an unused port. See Configuring Cluster Discovery and Internal Communication on page 405 for the procedure, including an additional step you must take.

Enabling Use of Oracle Coherence as the Cache Provider

If you want to use the Oracle Coherence cache provider, you must provide a fully licensed, supported version of the software. See the product readme file for supported version information. You must also enable Oracle Coherence as the cache provider, as explained below.

> You cannot use the `coherence.jar` file from an earlier TIBCO BusinessEvents release.

To Enable Oracle Coherence as the Cache Provider

1. Copy the `coherence.jar` file from your Oracle Coherence installation to *BE_HOME*`/lib/ext/tcpl`.

   (This location is preconfigured in the `studio/eclipse/configuration/studio.tra` classpath, as shipped. If you use a different location, update the classpath.)

2. Enable the Coherence category functions as follows:

   a. Open the *BE_HOME*`/studio/eclipse/configuration/studio.tra` file for editing.

   b. Change the setting of the following property to true (it is false as shipped):

      ```
      TIBCO.BE.function.catalog.coherence=true
      ```

   c. Save the file and restart TIBCO BusinessEvents Studio.

   For your information, log messages printed to the console at startup show the location of the files in use. See *BE_HOME*`/bin/logs/cep-engine.log`.

Configuring Oracle Coherence Cluster Discovery

See Enabling Use of Oracle Coherence as the Cache Provider on page 414 for prerequisite actions you must take before you can use Oracle Coherence as the cache provider.

You can use either multicast or well-known-address (WKA) discovery, as appropriate. See Cache Cluster Member Discovery in *TIBCO BusinessEvents Architect's Guide* for basic guidelines. Procedures for both methods of discovery are explained below.

Guidelines for Managing Coherence Clusters

Provided Files

The following files are located in *BE_HOME*`/lib/cep-datagrid-oracle.jar`:

- `coherence-cache-config-jdbc.xml` is an example cache configuration descriptor file.
- `tangosol-coherence-override-tibco-be.xml` is an example operational descriptor override file. To reference such a file, use the property `tangosol.coherence.override` in the CDD file.

To understand when and how to use them, read the Coherence documentation.

The following links to Coherence documentation provide helpful information for use of Coherence as the cache provider.

Checklist and Guidelines Before Architecting a New Project

http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/deploy_checklist.htm
http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/deploy_plat_consid.htm
http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/tune_perftune.htm

Coherence Network Protocol

http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/cluster_tcmp.htm
http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/tune_datagramtest.htm

Coherence Metrics

http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/appendix_mbean.htm
http://download.oracle.com/docs/cd/E15357_01/coh.360/e15723/appendix_operational.htm

Configuring Multicast Cluster Discovery for Coherence Clusters

Multicast is the default option. If default values for multicast properties work in your environment, you can omit this procedure.

To Configure Multicast Cluster Discovery

1. If the default values for the following properties are not appropriate for your environment, add them to the properties sheet of the CDD Cluster tab and specify values as needed:

   ```
   tangosol.coherence.clusteraddress
   tangosol.coherence.clusterport
   tangosol.coherence.ttl
   ```

   See Multicast Discovery Properties on page 419 for details

2. For multicast discovery you may also need to set these properties as explained in Localhost and Localport Properties on page 420:

   ```
   tangosol.coherence.localhost
   tangosol.coherence.localport
   ```

Specifying one or more well-known addresses disables all multicast communication. Remove any well-known address properties, if any.

Configuring Well-Known Address Cluster Discovery for Coherence Clusters

You can add up to nine well-known addresses. If you need to add more refer to Coherence documentation for instructions about using override files.

To configure well-known address discovery, you must configure machine-specific settings at the cluster level and at the processing unit level.

One processing unit deployed to a WKA machine must have the additional WKA configuration. Additional processing units can be deployed to a WKA machine, configured in the usual way, and they will discover and join the cluster in the usual way at runtime.

To Configure Well-Known Address Cluster Member Discovery

For details about the properties, see Well-Known Address Properties on page 422.

1. In the CDD file editor Cluster tab properties sheet, add a pair of WKA properties for each machine you want to configure as a well-known address machine:

   ```
   tangosol.coherence.wkan  HostIP
   tangosol.coherence.wkan.port  Hostport
   ```

For example at the cluster level you might have these two WKA machines:

Property: `tangosol.coherence.wka1`	Value: `10.97.118.151`
Property: `tangosol.coherence.wka1.port`	Value: `8098`
Property: `tangosol.coherence.wka2`	Value: `10.97.118.152`
Property: `tangosol.coherence.wka2.port`	Value: `8098`

If two engines run on one machine (with a single IP), ensure that each engine uses a different port. For example:

Property: `tangosol.coherence.wka1`	Value: `10.97.118.151`
Property: `tangosol.coherence.wka1.port`	Value: `8098`
Property: `tangosol.coherence.wka2`	Value: `10.97.118.151`
Property: `tangosol.coherence.wka2.port`	Value: `8099`

2. In the Processing Units tab properties sheet for a processing unit (PU), configure one set of WKA properties to match one of the cluster level set of WKA properties:

   `tangosol.coherence.localhost` *HostIP*
   `tangosol.coherence.localport` *Hostport*

   If you will deploy using TIBCO BusinessEvents Monitoring and Management (MM) also add the following property:

   `be.engine.hostaddress` *HostIP*

   Use the same the value as the value of the `tangosol.coherence.localhost` property. See Configuring for Coherence WKA Cluster Discovery in Chapter 3, Basic MM Configuration in *TIBCO BusinessEvents Administration* for more steps required when configuring MM.

   Set the `localhost` property to the IP of the host where you will deploy the PU, and set the `localport` property to the port defined in the cluster properties `localport` property.

3. Repeat step 2 until you have configured a PU with matching PU-level properties for each cluster-level (that is, machine-level) set of WKA properties.

   For example, if you configured the two well-known addresses shown in step 2, then at the processing unit level you would configure two processing units as follows:

   `tangosol.coherence.localhost`. Value: `10.97.118.151`
   `tangosol.coherence.localport`. Value: `8098`

   `tangosol.coherence.localhost`. Value: `10.97.118.152`
   `tangosol.coherence.localport`. Value: `8098`

At deploy time you must deploy the processing units on the appropriate (matching) WKA machine. It can be helpful if the name of the PU contains the machine name as a reminder that this PU must be deployed to that machine.

See Localhost and Localport Properties on page 420.

CDD Cluster Tab Coherence Properties Reference

Oracle Coherence Cache Provider This section is relevant only if you use Oracle Coherence as the cache provider.

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

Add properties as needed to configure multicast cluster member discovery, or well-known address cluster member discovery. See Configuring Oracle Coherence Cluster Discovery on page 415 for details.

Table 43 CDD Cluster Tab Coherence Properties (Sheet 1 of 6)

Property	Notes
Multicast Discovery Properties If you will define cluster members using multicast discovery properties, use the properties in this section, instead of those in the section Well-Known Address Properties on page 422.	
`tangosol.coherence.clusteraddress`	Use this setting if multicast discovery is used and if you need a non-default value. Specifies the multicast IP address that the socket will listen to or publish on. Possible values are addresses between (and including) 224.0.0.0 and 239.255.255.255. Default value is `224.3.3.1`
`tangosol.coherence.clusterport`	Use this setting if multicast discovery is used and if you need a non-default value. Specifies the port that the socket will listen to or publish on. Possible values are integers between 1 and 65535. Default value is 35463

Table 43 CDD Cluster Tab Coherence Properties (Sheet 2 of 6)

Property	Notes
`tangosol.coherence.ttl`	Specifies the time-to-live setting for the multicast, that is, the maximum number of "hops" a packet can traverse. A hop is defined as a traversal from one network segment to another via a router. For production use, set this value to the lowest integer value that works. Setting the value too high can use unnecessary bandwidth on other LAN segments and can even cause the operating system or network devices to disable multicast traffic. On a single-host cluster, set to zero (0). On a simple switched backbone, set to 1. On an advanced backbone with intelligent switching, it may require a value of 2 or more. **Note**: A value of 0 is intended to keep packets from leaving the originating machine. However, some operating systems do not implement this correctly, and the packets may in fact be transmitted on the network. Required for multicast configuration. Possible values are integers between 0 and 255. Default value is 4
Localhost and Localport Properties These properties are used in these cases: • When a host has multiple network cards. • For multicast discovery when more than one cluster is running on the same subnet (localhost is required but not localport in this case.) • Add these properties at the PU level when well-known address discovery is use as explained in Configuring Well-Known Address Cluster Discovery for Coherence Clusters on page 416. Default values are provided at the cluster level. However if you need specify these properties at the PU level, add them as Processing Units tab properties and provide the values as needed.	
`tangosol.coherence.localhost`	Specifies the IP address that the socket will listen to or publish on. As needed, you can set the value of the `localhost` property to the value `localhost`. However, if `localhost` is used as the loop back address (127.0.0.1) you must enter a machine name or IP address. Default value is `localhost`.

Table 43 CDD Cluster Tab Coherence Properties (Sheet 3 of 6)

Property	Notes
`tangosol.coherence.localport`	
	Specifies the port that the socket will listen to or publish on. Possible values are 1 to 65535. Default value is 8088. **Note** If a specified port is not available, the object management layer (by default) increments the port number until it finds an available port. Avoid potential conflicts by choosing a number that is not close to a port used by other software in your environment. **Tip** To turn off auto-incrementing, add this property: `tangosol.coherence.localport.adjust=false`

Table 43 CDD Cluster Tab Coherence Properties (Sheet 4 of 6)

Property	Notes
Well-Known Address Properties See Configuring Well-Known Address Cluster Discovery for Coherence Clusters on page 416. **Note** If you will discover cluster members using well-known addresses, use the properties in this section, and remove the multicast discovery properties shown in the section Multicast Discovery Properties on page 419.	

`tangosol.coherence.wkan`
`tangosol.coherence.wkan.port`
	The addresses and ports for machines used by the well-known address cluster discovery protocol.
	At least one of these machines must be running at any time so that others can join the cluster.
	For `tangosol.coherence.wkan`, enter the IP address.
	For `tangosol.coherence.wkan.port`, enter a value between 1 and 65535.
	For example (in the UI the properties are not entered quite this way):
	`tangosol.coherence.wka1        10.97.118.151`
	`tangosol.coherence.wka1.port   8088`
	`tangosol.coherence.wka2        10.97.118.152`
	`tangosol.coherence.wka2.port   8088`
	Also at the Processing Units tab, configure `localhost` and `localport` properties for one processing unit that will be deployed to the WKA machine. Set the `localhost` value to the value of the `wkan` property. Set the `localport` value to the value of the `wkan.port` property. (See Localhost and Localport Properties on page 420)
	Note You can configure two well-known addresses for the same machine, and use a different port number for each. In this case you would also configure two processing units, each of which matches one set of WKA properties.
	Tip To turn off auto-incrementing, add this property:
	`tangosol.coherence.localport.adjust=false`

Table 43 CDD Cluster Tab Coherence Properties (Sheet 5 of 6)

Property	Notes
Other Coherence Properties These properties are used in various situations.	
`tangosol.coherence.distributed.threads`	Specifies the number of Coherence daemon threads used by the distributed cache service when Oracle Coherence is used as the cache provider. Mainly used with write-behind database strategy. When used with cache aside strategy, this setting is used for handling cache operations only (gets and puts). In this release the property must be set to the same value across the cluster. Value is per processing unit. A value of zero (0) means all relevant tasks are performed on the service thread. See Write Behind Options in *TIBCO BusinessEvents Architect's Guide*. Default value is 0
`tangosol.coherence.override`	Specifies the location of an Operational Descriptor Override File. A sample value is: `file:/c:/tmp/my_tangosol-coherence-override.xml` An example override file is provided in *BE_HOME*`/lib/cep-datagrid-oracle.jar`. Use of an operational descriptor override is not generally required. For details on override files, see Coherence documentation.

Table 43 CDD Cluster Tab Coherence Properties (Sheet 6 of 6)

Property	Notes
`tangosol.coherence.localport.adjust`	
	An auto-incrementing feature ensures that a different port is used if one specified is already in use. However in various situations you may want to turn off this behavior. For example, if you use TIBCO BusinessEvents Monitoring and Management, and the MM server runs on the same machine as any of the monitored cluster engines, you must explicitly ensure that all ports used by MM *and* the monitored cluster are unique. Therefore the auto-incrementing feature may not be appropriate. To turn off auto-incrementing, add this property and set the value to false. Default is true.
`tangosol.coherence.guard.timeout`	
	When infrastructure latency occurs such as remote databases, you can improve the performance of inference engines by setting the property to 0 in be-engine.tra. `tangosol.coherence.guard.timeout=0`

Chapter 25 Load Balancer Configuration

TIBCO BusinessEvents Express This chapter relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

This chapter explains how to configure agents to work cooperatively as routers and receivers to ensure that related messages arriving from queue sources are handled by the same agent, so that related information is locally available.

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

Topics

Understanding Load Balancing Options

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

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

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

Basic Load Balancing

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

Certain aspects of the design have to be managed by the application. See Designing for Concurrency on page 96 for related information.

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

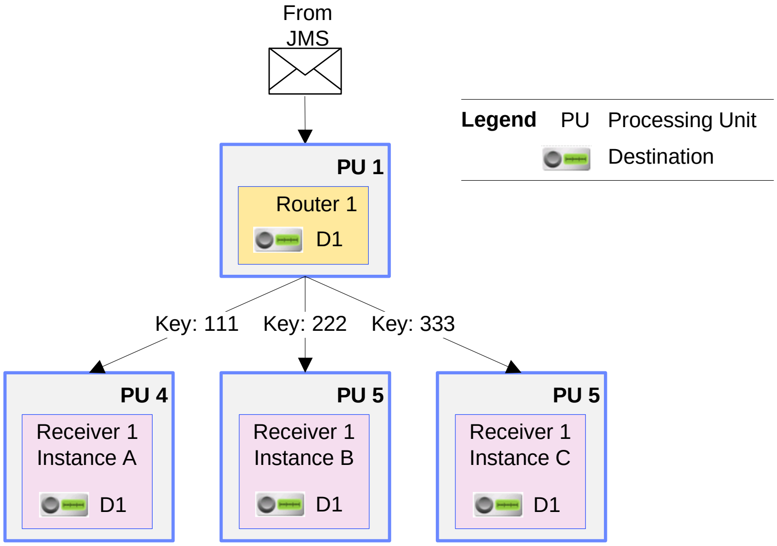
Content-aware Load Balancing

With content-aware load balancing, all related events arriving from queues are routed to the same agent using a routing key. The key uses the value of a selected event property. For example, if the event property is ZipCode then a routing key is a specific zip code. All messages relating to one zip code are routed (over TCP) to the same agent, providing "session stickiness."

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

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

The following diagram shows the CDD based router and receiver configuration.



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

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

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

The router redirects events over TCP to a receiver, based on the destination and the routing key values. The router transparently distributes the load across the available receivers. If a receiver agent fails, its messages (that is, messages with the key that the router was sending to that agent) are routed to another receiver and continue to be handled by that other receiver.

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

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

Content-Aware Load Balancer Configuration

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

Pair Configuration

Preconfigured using the Load Balancer tab of the CDD.

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

See CDD Load Balancer Tab Properties Reference on page 431 for details.

Adhoc Configuration

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

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

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

Adhoc Load Balancer Configuration

This section provides guidelines to configure the router and receivers in an adhoc load balancer configuration.

Router Configuration

`LoadBalancer.Router.*` functions are used for the router side.

Create the Load Balancer

Use this rule function as a startup rule function. It creates and returns a load balancer that can be used to send messages to load balanced remote destinations. You create the router TCP connection in a startup rule function.

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

Send Event to Receiver

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

```
void send(Object loadBalancer, SimpleEvent event, String routingKey)
```

Discard the Load Balancer

Put this rule function in a shutdown rule function.

```
Object loadBalancer = Util.HashMap.remove(String mapID, String key);
Util.HashMap.deleteMap(Object LoadBalancerReceiver );
```

This rulefunction discards the load balancer.

```
void discardLoadBalancer(Object loadBalancer)
```

Receiver Configuration

`LoadBalancer.Receiver.*` functions are used for the receiver side.

Local Channel

Receiver needs a local channel. The port information is obtained like this:

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

Create a Receiver

Creates and returns a receiver object that receives messages from a router. Messages will be received from the router on the local channel and destination specified. Create the receiver TCP connection in a startup rule function.

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

Discard a Receiver

You discard the receiver in a shutdown rule function.

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

Receiver Membership Functions in Catalog

`LoadBalancer.Receiver.Membership.isInFlux`
Returns true if the loadbalancer node membership is in a state of change such as nodes joining and/or leaving currently or in the recent past.

`LoadBalancer.Receiver.Membership.getRecentChangeAt`
Returns the timestamp (1970 epoch milliseconds) at which the most recent membership change occurred.

CDD Load Balancer Tab Properties Reference

Configure the load balancer by specifying the receiver and router agents, and setting the destination.

Table 44 CDD Load Balancer Tab Properties

Property	Notes
Pair Configuration	
Name	Name of the pair configured load balancer.
JMS Destination	Destination used by the router and receiver agents. Ensure that the destination has a default event configured.
Key	Routing key used by the pair configuration.
Router	Router agent class for the load balancer configuration.
Receiver	Receiver agent class for the load balancer configuration.
Adhoc Configuration	

Name	Name of the adhoc load balancer.
	When creating a load balancer using the catalog function `createLoadBalancerTo`, the name of the load balancer must be specified. For example,
	`Object loadBalancer = `
	`LoadBalancer.Router.createLoadBalancerTo("adhocLoadBalancerName");`

Table 44 CDD Load Balancer Tab Properties

Property	Notes
JMS Destination	Destination used by the router and receiver agents. In an adhoc configuration, a local channel with a local destination is used to communicate between the router and receiver agents.
Properties	
Transport	Transport to enable communication between the receiver and the router. Typically, this happens through an internal TCP connection. The default value is tcp.
Hostname	Hostname where the load balancer is configured. The default value is localhost.
Port	Port number used by the specified transport.

Chapter 26 Backing Store Configuration

This chapter explains how to configure CDD settings and properties that relate to backing store behavior.

Instructions for creating a backing store are in Chapter 30, JDBC Backing Store Setup, on page 501.

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

Topics

Configuring Backing Store Settings and Properties

This section provides a high level procedure and pointers to the reference tables and other sections that contain the details.

Oracle Database Strategy If the Strategy field for an Oracle database with a Shared All persistence (set under the Cluster tab > Backing Store) is set to `oracle`, then Oracle Database pooling strategy settings are used and various CDD properties act on the corresponding Oracle property, as noted in the reference table.

In some circumstances, it is necessary to configure some backing store settings before you set up the backing store. See Cases That May Need Additional Setup on page 505 in Chapter 30, JDBC Backing Store Setup.

To Configure the Cluster Tab Backing Store Settings and Properties

1. Open the CDD file you added, as explained in Adding a Cluster Deployment Descriptor (CDD) on page 393.
2. Select the Cluster tab > Backing Store node on the left and on the right, configure settings as explained in CDD Cluster Tab Backing Store Settings Reference on page 435.
3. Select the Cluster tab > Connection node on the left and on the right configure database connection settings. See Backing Store > Connection Settings on page 437.
4. Select the Cluster tab > Properties node on the left and configure properties on the right, following guidelines in CDD Cluster Tab Backing Store Properties Reference on page 439.
5. You can select domain objects (entities) to be included in or excluded from the backing store. In addition preloading options are available for loading domain objects from backing store to cache at system startup. See Chapter 28, Domain Objects Configuration, on page 455.
   - — See Configuring Preloading Options on page 457 for object settings
   - — See `be.engine.cluster.recovery.threads` in CDD Cluster Tab Backing Store Properties Reference on page 439.
6. Save the resource.

CDD Cluster Tab Backing Store Settings Reference

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

For the related procedure, see Configuring Backing Store Settings and Properties on page 434.

Table 45 CDD Cluster Tab Backing Store Settings (Sheet 1 of 4)

Property	Notes
Persistence Option: None	
Specifies that the cluster does not have a backing store or the backing store is temporarily disabled, for example during testing phases of a project. **Note**: Individual entities can be set to not use the backing store. See Has Backing Store in CDD Cluster Tab Domain Object Override Settings Reference on page 463. Default is set to None.	
Persistence Option: Shared All	
Database Type	Select which of the supported DBMS products to use: `Oracle, SQL Server, or Berkeley DB.` Default is `Oracle`.
Strategy	Used only if the Database Type is set to Oracle. If you use Oracle Database, you have the option of using either the TIBCO BusinessEvents internal pooling implementation, or Oracle Database's implementation. Possible values are as follows: **`jdbc`** Use the internal pooling mechanism. **`oracle`** Use Oracle's pooling mechanism (see the class `OracleConnectionCacheManager` in the package `oracle.jdbc.pool`). When set to `oracle` then the TIBCO BusinessEvents pooling property values are used to set their corresponding to Oracle Database properties. Default is `jdbc`.

Table 45 CDD Cluster Tab Backing Store Settings (Sheet 2 of 4)

Property	Notes
Cache Aside	Available only for the backing store database types Oracle and SQL Server. Choose between these two options: • **Checkbox unchecked means Write Behind** Writes data to the cache and then to the backing store. One write-behind thread is used for each entity type. If write-behind strategy is used with Oracle Coherence cache provider, you can also set `tangosol.coherence.distributed.threads` in the cluster level properties. • **Checkbox checked means Cache Aside** Writes data to the cache and at the same time to the backing store. User controls are available for threads and queue size, and other options such as using parallel or sequential operations in the post-RTC phase. See Chapter 10, Threading Models and Tuning in *TIBCO BusinessEvents Architect's Guide* for more information.
Enforce Pools	Available only for the backing store database types Oracle and SQL Server. Check this property if you want to enforce connection pool properties. See Database Connection Properties on page 439 for the property details. Default is unchecked.
Berkeley DB Data Store Path	Available only for the backing store database type Berkeley DB. Enter the absolute path of the data store to be used. The data store needs to be located on a network drive that can be accessed by all the cache nodes.

Table 45 CDD Cluster Tab Backing Store Settings (Sheet 3 of 4)

Property	Notes
Persistence Option: Shared Nothing See Setting up Shared Nothing Persistence, page 444 for details on setting up Shared Nothing persistence.	
Persistence Path	Specifies the absolute path to the directory where the data is to be stored. For example, /tmp/datastore/. Set this value for each individual processing units.
Persistence Policy	Specifies the type of communication to be used to maintain persistence: asynchronous (ASYNC) or synchronous (SYNC). • **ASYNC** This policy is recommended if you want to avoid frequent IO operations, which can slow inference agents. • **SYNC** Solid State Drives (SSD) are recommended when using this policy. Default is async.
Backing Store > Connection Settings Try running with default pool values and monitor the behavior. Using more connections improves runtime performance and can also speed up recovery in the event of a failure. Pool settings are used only if Enforce Pools is checked.	
URI	Specifies the project path, that is, the path from the project root to the JDBC Connection resource, to define the connection to the backing store. For example: `/SharedResources/JDBC Connection.sharedjdbc` You can also use a global variable to specify the connection. Default value is `%%DbUri1%%`.

Table 45 CDD Cluster Tab Backing Store Settings (Sheet 4 of 4)

Property	Notes
Min Size	Minimum number of JDBC connections in the JDBC connection pool used for the backing store. **Oracle Database Strategy** If the Cluster tab > Backing Store > Strategy field is set to `oracle`, then Oracle Database strategy settings are used, and this property corresponds to the `OracleConnectionCacheManager` class property `MinLimit`. Default is 10.
Max Size	Maximum number of JDBC connections in the JDBC connection pool used for the backing store. Connections do not exceed the maximum. The value of this property overrides the value of the Maximum Connections setting in the JDBC Connection resource. Although the limit is seldom reached, you can guarantee a connection is always available for a `dbwriter` thread as follows. Set the this field to the same value as the Agent.*AgentClassName*`.dbthreadcount` setting. Similarly (and also seldom needed), with Coherence cache provider, you can guarantee a connection is available by setting this field to the same value as the property `tangosol.coherence.distributed.threads`. **Oracle Database Strategy** If the Cluster tab > Backing Store > Strategy field is set to `oracle`, then Oracle Database strategy settings are used, and this property corresponds to the `OracleConnectionCacheManager` class property `MaxLimit`. Default is 10.
Initial Size	Specifies the initial size of the JDBC connection pool used for the backing store, when it is created on startup. For example: **Oracle Database Strategy** If the Cluster tab > Backing Store > Strategy field is set to `oracle`, then Oracle Database strategy settings are used, and this property corresponds to the `OracleConnectionCacheManager` class property `InitialLimit`. Default is 10.

CDD Cluster Tab Backing Store Properties Reference

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

For the related procedure, see Configuring Backing Store Settings and Properties on page 434.

Table 46 CDD Cluster Tab Backing Store Properties

Property	Notes
Database Connection Properties Used only if Enforce Pools is checked (see CDD Cluster Tab Backing Store Settings Reference on page 435).	
`be.backingstore.dburi.pool.waitTimeout.0`	Used only if the Strategy setting (see CDD Cluster Tab Backing Store Settings Reference on page 435) is set to `oracle`. **Oracle Database Strategy** If the Cluster tab > Backing Store > Strategy field is set to `oracle`, then Oracle Database strategy settings are used, and this property corresponds to the `OracleConnectionCacheManager` class property `WaitTimeout`. Specifies behavior when a connection is requested and there are already Cluster tab > Backing Store > Connection > Max Size connections active. If the `be.backingstore.dburi.pool.waitTimeout.0` value is greater than zero (0), each connection request waits for up to the specified number of seconds. If no connection is returned to the pool before the timeout elapses, a `No Database Connection available` exception is thrown. The `waitTimeout` and `inactivityTimeout` properties specify wait periods to minimize the creation and destruction of connections (an expensive operation). Default is 1 second.

Table 46 CDD Cluster Tab Backing Store Properties

Property	Notes
`be.backingstore.dburi.pool.inactivityTimeout.0`	
	Oracle Database Strategy If the Cluster tab > Backing Store > Strategy field is set to `oracle`, then Oracle Database strategy settings are used, and this property corresponds to the `OracleConnectionCacheManager` class property `InactivityTimeout`. Specifies the number of seconds an unused connection remains available (so that other threads can use it). After this period, the connection is closed and removed from the pool. Default value is 900 seconds.
`be.backingstore.readtimeout`	
	Oracle Database Strategy If the Cluster tab > Backing Store > Strategy field is set to `oracle`, then Oracle Database strategy settings are used, and this property corresponds to the `OracleConnectionCacheManager` class property `ReadTimeout`. Use this property to handle situations where engines running inference agents hang when the JDBC connection to the backing store is slow or intermittent. If no response is received from the database within the specified period, a call is aborted. Time unit is milliseconds. A value of 0 (zero) means that no timeout is set. Default value is 0
`be.backingstore.recreateOnRecovery`	
	Set this property to true if the database pool size does not recover to the initial or minimum connection size, as defined by Min Size and Max Size properties (in CDD Cluster Tab Backing Store Settings Reference on page 435). Default value is false.

Table 46 CDD Cluster Tab Backing Store Properties

Property	Notes
Other properties	
`be.backingstore.commitSize`	Used with the Coherence cache provider and write-behind strategy only. Sets the maximum number of RTC transactions that a distributed cache service thread takes from the distributed cache service queue and processes in one batch. When threads are idle, they take jobs from the queue in smaller batches. Set this property to the desired number of transactions to suit your needs. See Write Behind Options in *TIBCO BusinessEvents Architect's Guide*. Default is 10.
`be.backingstore.optimize.reads`	Used with Microsoft SQL Server only. Set the property to true to improve the runtime performance. Use NOLOCK for SELECT statements to avoid locks on the database or table when SELECT statements are issued. An example syntax is: `select * from dbo.D_MailerIndex with (nolock) where ...`
`be.backingstore.optimize.writes`	Used with Microsoft SQL Server only. Set the property to true to improve the runtime performance. Use ROWLOCK with UPDATE or DELETE statements to avoid lock contentions. When you use rowlock in the T-SQL statement, the SQL Server locks only the rows that match the 'where' condition and not the entire table. An example syntax is: `DELETE FROM dbo.D_Mailed WITH (ROWLOCK) where mailernumber = '12345678895' and time_created$ = '`*`somedate`*`'`
`be.backingstore.timestamp.useDataTimeZone`	Used when the backing store is enabled. Set the property to true to ensure that the correct datetime properties are retrieved when an agent's timezone is changed and the agent is restarted.

Table 46 CDD Cluster Tab Backing Store Properties

Property	Notes
`be.engine.cluster.as.useSharedNothing`	
	Used to enable Shared Nothing persistence option. The data is stored at individual node levels and is not shared across the cluster, thus enabling faster processing. Default is false.
`be.engine.cluster.as.dataStorePath`	
	Used to specify the location where the data is to be stored. The location is set at the processing units (PU) level and needs to be set for each individual processing unit. Specify the value as absolute path of the directory where the data is to be stored. For example, `/tmp/datastore/`.
`be.engine.cluster.cleanup`	
	Used with write-behind strategy only. By default, deleted entities are removed from the backing store automatically at system startup. Set this property to `false` to disable that behavior. See Handling Entities Deleted from a Write-Behind Backing Store on page 529 for more details. Default is `true`.
`be.engine.cluster.recovery.threads`	
	Recovery threads are used when preloading the cache during startup. For an explanation of preloading and other preloading controls, see Configuring Preloading Options on page 457. Default is 4.

Table 46 CDD Cluster Tab Backing Store Properties

Property	Notes
`be.engine.cluster.useDBBatching`	
	Note For use with cache aside and only when the parallel operations feature is used. This property has no effect if `Agent.`*AgentClassName*`.dbOpsBatchSize` is set to 1 (see Table 53, CDD Agent Classes Tab Inference Agent and Query Agent Properties, on page 479). This property affects how all RTC transactions that a database writer thread takes from the database operations queue are written to the backing store: • When set to true, the RTC transactions are handled as one job. • When set to false, each RTC transaction is handled as a separate job. For a guide to usage of this and other related properties, see the section Database Write Tuning Options for Cache Aside in Chapter 10, Threading Models and Tuning in *TIBCO BusinessEvents Architect's Guide*. Default is false.

Setting up Shared Nothing Persistence

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

Following steps explain how to configure a TIBCO BusinessEvents project to use Shared Nothing persistence:

1. In BusinessEvents Studio Explorer, edit the project CDD file to set the Object Management to **Cache** and Provider to **TIBCO**.
2. For the Backing Store, select the Persistence Option as **Shared Nothing**.
3. Enter the value for persistence path as the absolute path to the directory where data is to be stored.
4. Select the appropriate persistence policy.
5. Save the CDD file.

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

Runtime Configuration to Specify the Engine Name Property

The TIBCO BusinessEvents engine name is used to name the Shared Nothing files for that node. To enable the engine name to be used, you need to explicitly set the engine name property when running clusters with Shared Nothing. The `-n` <*name*> must be specified for all the nodes (inference and cache) in the cluster.

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

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

Nodes with duplicate engine names will not be initialized.

Features Supporting the Shared Nothing Persistence Option

You can use the Shared Nothing persistence option with the following features:

- Concepts
- Events, with and without payloads
- Schedulers

- Scorecards
- State machines

> When querying by extid, you must use the functions `Cluster.DataGrid.CacheLoadConceptByExtIdByUri` and `Cluster.DataGrid.CacheLoadEventByExtIdByUri`.

Note that the following do not support the use of Shared Nothing persistence option:

- TIBCO BusinessEvents Views
- History feature in Concepts.
- Catalog functions - Temporal.History.* and Temporal.Numeric.addAllHistory*
- Functions Cluster.DataGrid.CacheLoadConceptByExtId and Cluster.DataGrid.CacheLoadEventByExtId.

Chapter 27 Berkeley DB Shared All Persistence

This chapter explains how to configure the Berkeley DB Shared All Persistence option for persisting cache data to disk.

Topics

Berkeley DB Shared All Persistence Overview

This form of data persistence uses TIBCO BusinessEvents DataGrid as the cache provider and Oracle Berkeley DB Java Edition software as the data store (not bundled with the TIBCO BusinessEvents software).

You must obtain a separate license for the Oracle Berkeley DB Java Edition software that is appropriate for your usage if you wish to use the Shared All with Berkeley DB feature.

With this form of persistence, cache servers act as persisters, and interact with the persistence layer. All cache nodes must have access to a reliable shared file system. The Berkeley DB data store is created within this shared file system.

The Berkeley DB Shared All Persistence option uses stores on a shared file system, such as NFS. This can result in better performance than use of a traditional DBMS product accessed over the network.

Reading from the Data Store

Persisters load data in bulk from the data store at startup. The following functions are used to load individual concepts into cache:

`CacheLoadConceptByExtIdByURI()`

`CacheLoadConceptById()`

Writing to the Data Store

Data is flushed to the operating system buffers for every write operation. TIBCO BusinessEvents uses `WRITE_NO_SYNC` durability by default, which means that Berkeley DB will flush every write to the operating system's buffers immediately but not call `fsync`. In case of an application crash, there will be no data loss as long as the underlying operating system synchronizes its buffers to disk.

Notes

- TIBCO BusinessEvents internally sets durability to `WRITE_NO_SYNC` durability. This behavior can be overridden by setting the `je.txn.durability` property in the `je.properties` file. See Configuring the Berkeley DB Shared All Persistence Option on page 450.
- The `CacheLoadConceptByExtId()` function is not supported for use with this feature.
- Scorecards are not persisted (and so cannot be recovered).
- Scheduler events are not persisted (and so cannot be recovered)

Configuring the Berkeley DB Shared All Persistence Option

This section explains how to configure Berkeley DB Shared All Persistence. Configuration is done in the CDD file and optionally in a properties file called the `je.properties` file.

1. Download the supported version of the Oracle Berkeley DB Java Edition software from the Oracle web site. See the Release Notes for details on supported versions. Place the `je-`*versionNumber* JAR file in the following directory:

   BE_HOME`/hotfix/lib/ext/tpcl/`

2. Open the project CDD or add one if the project does not yet have a CDD.

   — If you add a new CDD select Cache object management type in the wizard.

   — If you edit an existing CDD, select the Cluster tab. If the Object Management node is set to In Memory, right-click Object Management and select Cache.

3. Select the Cluster tab. In the Cluster tab navigation tree, select Object Management. In the Configuration panel, set the following:

   — Provider: TIBCO

   — Cache Agent Quorum: Generally set to the number of cache agents (see Table 41, CDD Cluster Tab Cache OM Settings, on page 401 for details)

   — The following items are not relevant: Number of Backup Copies, Entity Cache Size, Object Table Cache Size.

4. In the Cluster tab navigation tree, select Backing Store. In the Configuration panel set the following values:

   — Persistence Option: **Shared All**

   — Database Type: **Berkeley DB**

   — Berkeley DB Data Store Path: The name and file location of the data store and (if used) the `je.properties` file.

5. If the CDD is using overrides for concepts or events, select the checkbox "Has Backing Store" to indicate that the entity will be stored in Berkeley DB. If the checkbox is not selected, then the entity will not be stored.

6. In the Cluster Tab navigation tree, select Properties. In the configuration tab, add the following properties (see Table 47, Berkeley DB Shared All Persistence CDD Configuration Properties, on page 452 for details):

7. Select the Agent Classes tab. Configure cache agents with the properties shown in Cache Agent Properties on page 452.

8. In the Agent Classes tab, configure inference agents with the properties shown in Inference Agent Properties on page 452.

9. Save the CDD file and build project EAR files for deployment.

10. If you will use any properties shown in Reference To Berkeley DB (JE) Properties on page 454, add the properties as name-value pairs in a file called `je.properties`. Place the file in the same location as the data store (as specified in the Berkeley DB Data Store Path setting).

Reference to Berkeley DB Shared All Persistence CDD Properties

This section provide a reference to the CDD properties

Table 47 Berkeley DB Shared All Persistence CDD Configuration Properties

Property	Notes
Cluster Settings	
Berkeley DB Data Store Path	Specifies the name and file location of the data store. If this setting is not specified, a directory named `datastore` is created under the working directory. If you use a `je.properties`, file place it in the same location as the data store. See Reference To Berkeley DB (JE) Properties on page 454.
Cache Agent Properties	
`be.backingstore.useobjecttable`	Required. Required value is `false`.
`be.engine.cluster.as.lock.ttl`	Required. Value is specified in milliseconds. Controls the time after which TIBCO BusinessEvents DataGrid will forcibly unlock a space key. Thirty seconds is a reasonable value when the system not operating under heavy load (More than 2000 transactions per second may required a longer timeout period). Default value is 30000.
Inference Agent Properties	
`be.backingstore.useobjecttable`	Required. Required value is `false`.
`be.engine.cluster.as.lock.ttl`	Required. Value is specified in milliseconds. See notes for this property in the Cache Agent Properties section of the table. Default value is 30000.

Table 47 Berkeley DB Shared All Persistence CDD Configuration Properties (Cont'd)

Property	Notes
`be.engine.cluster.as.node.retry.times`	
	Required. Specifies the number of times TIBCO BusinessEvents retries a `put` or `putAll` call on the TIBCO BusinessEvents DataGrid cache. Each retry is done after five seconds. The number of retries depends on the `be.engine.cluster.as.lock.ttl` property. To calculate the value for retry times, use the following formula: `retry.times=lock.ttl/5+1` For example, if you set `be.engine.cluster.as.lock.ttl` to `30000`, then you would set the `be.engine.cluster.as.node.retry.times` value to 7.

Reference To Berkeley DB (JE) Properties

This section provides a reference to the Berkeley DB JE properties you can use to override the default configuration. For more details on the Berkeley DB JE properties, refer to the Berkeley DB product documentation:

http://www.oracle.com/technetwork/database/berkeleydb/je-faq-096044.html

http://docs.oracle.com/cd/E17277_02/html/GettingStartedGuide/administration.html#propertyfile

The properties are added to a file called `je.properties`. See Configuring the Berkeley DB Shared All Persistence Option on page 450 for the required location.

Table 48 Berkeley DB JE Properties

Property	Notes
`je.txn.durability`	Indicates the durability of a transaction. Possible values are `WRITE_NO_SYNC`, `NO_SYNC`, and `SYNC`. TIBCO BusinessEvents internally sets durability to `WRITE_NO_SYNC`. You can override that setting using this JE property.
`je.log.numBuffers`	The number of write buffers that the Berkeley DB uses internally. Default value is `3`.
`je.log.bufferSize`	The size of each write buffer. Default value is `1048576`.
`je.log.totalBufferBytes`	The sum of sizes of all the write buffers. Default value is `3145728`.
`je.maxMemoryPercent`	The percentage of the JVM maximum memory to which the cache is limited. Default value is 60%

Chapter 28 Domain Objects Configuration

This chapter explains options that determine how entity object instances are stored and options for preloading them from backing store to cache at system startup.

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

Topics

Configuring Domain Object Settings

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

If a backing store is already set up, and you enable any Use Backing Store settings, you must update the backing store setup. The backing store will not operate correctly unless you do so. See Updating an Existing Backing Store Schema on page 526.

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

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

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

- The mode: Cache plus Memory, Cache Only, or Memory Only. See Chapter 8, Cache Modes and Project Design in *TIBCO BusinessEvents Architect's Guide* to understand the effect of the different modes. Cache plus Memory (Cache + Memory in the UI) is a deprecated feature.
- Whether the objects or handles or both are preloaded into the backing store at startup.
- A preload fetch size (one setting for both objects and handles).
- Whether the cache is limited or unlimited. If limited, you can specify the size of the cache. See Configuring a Limited (or Unlimited) Cache on page 399.

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

- Whether the object is stored in the backing store or not.
- A subscription preprocessor (used for Cache Plus Memory mode only).
- A backing store table name (used when setting up a backing store. See Chapter 30, JDBC Backing Store Setup, on page 501.

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

Configuring Preloading Options

Preloading refers to the loading of the cache with objects from the backing store, at system startup. If you do not preload, then the objects are loaded as needed, which can affect performance the first time each object is requested, depending on the size of the objects.

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

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

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

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

One tuning property for preloading is available:

```
be.engine.cluster.recovery.threads
```

To Configure the Cluster Tab Domain Object Settings and Properties

1. Add a CDD as explained in Adding a Cluster Deployment Descriptor (CDD) on page 393, or open an existing CDD, configured for Cache OM.
2. Select the Cluster tab > Domain Objects > Default node on the left and on the right, configure settings as explained in CDD Cluster Tab Domain Objects Default Settings Reference on page 459.
3. To add object-level overrides and other object-level settings do the following:
    a. Select the Cluster tab > Domain Objects > Overrides node on the left and click **Add**.
    b. At the Entity Selection dialog, select the ontology object type or types you want to configure and click **OK**. The first entity in the list is selected and the configuration section for the entity displays on the right.
    c. Select the `/uri` node for each selected entity in turn and configure the settings on the right as needed. You can also edit existing override entries,

and remove entries not needed (by clicking Remove). See CDD Cluster Tab Domain Object Override Settings Reference on page 463.

When a project is migrated from 4.x to 5.1, domain object override entries are automatically added for all entities. See Migration from 4.x to 5.1 in *TIBCO BusinessEvents Installation* for details.

4. Save the resource.

CDD Cluster Tab Domain Objects Default Settings Reference

The Domain Object Default settings apply to all objects except those for which you explicitly configure overrides, using the Domain Object Overrides section (see CDD Cluster Tab Domain Object Override Settings Reference on page 463).

Table 49 CDD Cluster Tab Domain Object Default Settings (Sheet 1 of 4)

Property	Notes
Mode	With Cache OM, you can keep memory objects in the cache or Rete network using the following cache modes. • **Memory Only**: Objects are not persisted in the cache. They are kept in the Rete network (working memory) only. • **Cache Only**: Objects are persisted in the cache. They must be loaded into working memory as needed. This is the most common choice for a cache cluster. • **Cache+Memory** (also written as Cache Plus Memory): Deprecated feature. Due to issues with concurrency, in clusters with multiple active inference agents, use Cache Plus Memory only for constants and objects that change infrequently. **Note** If you set the mode to Memory Only, the rest of the properties in this section are not relevant and are ignored. See Chapter 8, Cache Modes and Project Design in *TIBCO BusinessEvents Architect's Guide* to understand the effect of this setting. Default is Cache Only.
Preload Entities	Specifies whether objects are loaded into the cache from the backing store on system startup (both normal startup and recovery). **Checked**: All objects are preloaded into the cache from the backing store. Lower level settings can override this setting by excluding specified objects. **Unchecked**: No objects are preloaded. into the cache from the backing store. Lower level settings can override this setting by including specified objects. Default is unchecked.

Table 49 CDD Cluster Tab Domain Object Default Settings (Sheet 2 of 4)

Property	Notes
Preload Handles	Specifies whether object handles are loaded into the `ObjectTable` cache. The `ObjectTable` cache holds references (handles) to the objects themselves. Handles are used in the object table. See The Role of the Object Table in *TIBCO BusinessEvents Architect's Guide* for more details. **Checked**: All object handles are preloaded. Lower level settings can override this setting by excluding handles for specified objects. **Unchecked**: No object handles are preloaded into the cache from the backing store. Lower level settings can override this setting by including handles for specified objects. Default is unchecked.
Preload Fetch Size	If Preload Entities or Preload Handles or both are checked, this setting specifies the number of entity objects or handles (or both) to preload for each entity type whose objects or handles (or both) are configured to be preloaded. This setting applies to both objects and handles and cannot be set differently for each. Objects and handles are fetched in a non-deterministic manner. This setting can be overridden at the entity level. Set to 0 to preload all. Set to a number to load that number of objects or handles (or both). Default is 0. Ignored unless Preload Entities or Preload Handles or both are checked.

Table 49 CDD Cluster Tab Domain Object Default Settings (Sheet 3 of 4)

Property	Notes
Check for Version	This field applies to concepts that use cache-only mode or cache+memory mode. An inference agent uses its L1 cache, a local cache of limited size, to improve access time to the concepts stored in the cluster cache. When an agent finds a concept instance in this local cache, the Check for Version setting determines whether the agent will use the instance directly, or instead check in the cluster cache for more recent version. **If Checked** (default value) The agent checks in the cluster cache for a more recent version. If a more recent version exists, it will be used, and will replace the one found in the local cache. **If Not Checked** The agent uses the instance found locally. When content-aware load balancing is used, the local instance can be used without checking for version, improving performance. Default is checked.
Constant	This field applies to entities that use the cache-only mode or the cache+memory mode. In multi-engine deployments, use only for entities that do not change after creation. The processing unit has a special local cache used only for entities marked as Constant. Entities placed in this cache are only removed when they are explicitly deleted. If the processing unit finds an entity in the constant cache, it will use it without checking in the cluster. **If Checked** The entity is marked `Constant`, and uses the constant cache. **If Not Checked** The entity does not use the constant cache. Default is unchecked.

Table 49 CDD Cluster Tab Domain Object Default Settings (Sheet 4 of 4)

Property	Notes
Evict from Cache on Update	Used only if both of the following are the case: • Cache-aside database write strategy is used • The property `Agent.`*AgentClassName*`.cacheTxn.updateCache` is set to false (see CDD Agent Classes Tab Properties Reference on page 479) If checked: When a rule action changes the value of any of an entity's properties, then the entity instance is evicted from the cache (updates are saved in the backing store) Use as needed to improve performance and cache memory management. For example, if an entity is not accessed frequently, it may save memory in the cache if the entity is evicted from cache after it is updated. Possible values are checked (true) and unchecked (false). Default is unchecked.
Is Cache Limited	If checked, the cache size is limited. Limited cache requires use of a backing store. See Configuring a Limited (or Unlimited) Cache on page 399. The size of the entity cache and the size of the object table cache are set in the Object Management section of the Cluster tab. If not checked, the cache size is unlimited. You can override this default setting in entity overrides. Default is unchecked.
Subscribe Cluster	For objects that use Cache+Memory mode, check this checkbox to subscribe to subscription RTCs, so that changes to this object in one Rete Network are also changed in the Rete networks across all inference agents. See Using Locks to Ensure Data Integrity Within and Across Agents in *TIBCO BusinessEvents Architect's Guide* for details. Default is checked.

CDD Cluster Tab Domain Object Override Settings Reference

Many settings simply override the value of default settings. See Table 49, CDD Cluster Tab Domain Object Default Settings, on page 459 for general details about the use of each setting. The table below only provides details that are specific to overrides.

Table 50 CDD Cluster Tab Domain Object Override Settings (Sheet 1 of 4)

Property	Notes
Entity URI	Specifies the project path to the entity for which overrides are being set. Defaults to the selected entity's URI. For example: `/Concepts/MyConcept`.
Mode	Overrides the Default level setting for this object type. **Memory Only Mode** If you set the mode for an entity to Memory Only, the rest of the properties in this section are not relevant and are ignored. Backing store is disabled for entities that use Memory Only mode. **Caution** If you change from Memory Only mode to a cache mode after the backing store has been set up, you must update the backing store schema. See Updating an Existing Backing Store Schema on page 526.
Preload Entities	Specifies whether objects of the specified type are loaded into the cache from the backing store on system startup (both normal startup and recovery). Overrides the Preload Entities setting at the Default level. **default**: Use the Preload Entities setting specified at the default level. **true**: Objects of the specified type are preloaded into the cache from the backing store. If the default level setting is not to preload entities, you can use this override to preload selected entities. **false**: No objects of the specified type are preloaded into the cache from the backing store. If the default level setting is to preload entities, you can use this override to not preload selected entities. Default is default.

Table 50 CDD Cluster Tab Domain Object Override Settings (Sheet 2 of 4)

Property	Notes
Preload Handles	
	Specifies whether object handles for the specified type are loaded into the cache from the backing store on system startup (both normal startup and recovery). Overrides the Preload Handles setting at the Default level. **default**: Use the Preload Handles setting specified at the default level. **true**: Handles for the specified type are preloaded into the cache from the backing store. If the default level setting is not to preload handles, you can use this override to preload selected entities' handles. **false**: No handles for the specified type are preloaded into the cache from the backing store. If the default level setting is to preload handles, you can use this override to prevent preloading the selected entities' handles. Default is default.
Preload Fetch Size	
	Overrides the Preload Fetch Size setting in the Default settings.
Check for Version	
	Overrides the value of the same-named setting in the Default settings.
Constant	
	Overrides the value of the same-named setting in the Default settings.
Evict from Cache on Update	
	Overrides the value of the same-named setting in the Default settings.
Is Cache Limited	
	Overrides the value of the same-named setting in the Default settings.
Subscribe Cluster	
	If this object uses Cache+Memory mode, check this checkbox to subscribe to subscription RTCs, so that changes to this object in one Rete Network are also changed in the Rete networks across all inference agents. See Using Locks to Ensure Data Integrity Within and Across Agents in *TIBCO BusinessEvents Architect's Guide* for details.

Table 50 CDD Cluster Tab Domain Object Override Settings (Sheet 3 of 4)

Property	Notes
Subscription Preprocessor	If this object uses Cache+Memory mode, and Subscribe Cluster is checked, specify a subscription preprocessor. This preprocessor is generally used to provide locking to ensure data consistency. The required signature for a subscription preprocessor is as follows: *FunctionName*`(long id, String extId, int typeId, int version, boolean isDeleted)` See Using Locks to Ensure Data Integrity Within and Across Agents in *TIBCO BusinessEvents Architect's Guide* for details. Cache plus memory is a deprecated feature. See *TIBCO BusinessEvents Release Notes* for details.
Backing Store Section	
Has Backing Store	Used only if the Backing Store > Enabled checkbox is checked. To exclude an entity from the backing store, uncheck the Has Backing Store checkbox. **Caution** If you enable this override setting after the backing store has been set up, you must update the backing store schema. See Updating an Existing Backing Store Schema on page 526. **Concepts Related by Containment or Inheritance** All concepts related by containment or inheritance must either be included in the backing store or excluded from the backing store. That is, they must share the same value for the Has Backing Store setting. Default value is checked.
Table Name	Specifies a table name to be used in the backing store. Typically used if the entity name is long. See Ontology Identifiers that Exceed the DBMS Maximum Column Length on page 505 for details.
Properties Metadata Section	
Property	Displays the property name. Read-only.

Table 50 CDD Cluster Tab Domain Object Override Settings (Sheet 4 of 4)

Property	Notes
Present in Index	If checked, sets an unordered index on the property. Used only if Oracle Coherence is the cache provider. Also used only if the query agent has enabled indexing: see `be.agent.query.enable.filter.optimizer` in CDD Agent Classes Tab Properties Reference on page 479. Also see Indexing for More Efficient Cache Queries on page 294 for more details and options.
Max Length	Used with backing store to specify the length of string properties that exceed 255 characters (that is the actual contents stored in the column is more than 255 characters). Specifies the expected maximum length for the property. See String Properties that Exceed the DBMS Maximum Column Length on page 506 for details.

Reverse References	This setting is for use only with ConceptReference type concept properties.
	With a backing store, database updates related to a referring concept in a referenced concept can cause decreased performance. This happens when there are very many reverse references in a shared instance (referenced by many other instances).
	To address this issue, set the value to `false` for ConceptReference type properties.
	If you set the value to `false`, you must explicitly remove ConceptReference properties for deleted referenced concepts in the referring concept in your code.
	For example, if employee is a ConceptReference type property in a concept acme, and smith is an instance of a concept type employee, then you would set Reverse References to true for the employee ConceptReference property, and you would add something like this to rules:
	`acme.employee = null;`
	`Instance.deleteInstance(smith);`
	Or, for array properties:
	`Instance.PropertyArray.removeConceptReference(acme.employee, smith);`
	`Instance.deleteInstance(smith);`
	Default is `true`.

Chapter 29 Agent and Processing Unit Configuration

This chapter explains how to configure agent classes and processing units in the CDD file.

Topics

Collections, Agent Classes, and Processing Units

Destinations require additional configuration, which can be done in this tab. (Destinations that are added to agent classes individually can be configured at the Agent Classes tab.)

Collections

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

See Configuring Collections of Rules, Rule Functions, and Destinations on page 470.

Agent Classes

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

See Using Properties at Different Levels on page 390 to understand the effect of using agent class properties at the cluster level and at the processing unit level to widen the scope of the property.

See Configuring Agent Classes (All OM Types) on page 474

How Collections and Individual Resources are Used to Configure Agents

Different agent types use different types of resources.

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

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

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

Startup Functions and Shutdown Functions Select function collections and individual functions as needed, to define which functions execute at engine startup and shutdown respectively.

How Startup Rulefunctions and Shutdown Rule Functions are Executed

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

Log Configurations

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

See Understanding and Configuring Log Configurations on page 484

Configuring Collections of Rules, Rule Functions, and Destinations

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

Using References

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

To Configure Collections

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

For the log configurations procedure, see To Add a Log Configuration on page 485

1. In the Collections tab do any of the following:

   — To add a new collection, select the parent for the collection type, Rules, Destinations, Functions, or Log Configurations as needed, and click **Add**.

   — In the *Item* Collection field that appears on the right, enter a name for the group and click **Add** again.

   — To add *items* and *item* group references to a collection select the item collection and then click **Add**.

   You see the Select *Items* dialog.

2. In the Select *Items* dialog do any of the following:

   — To add *items*, in the ***Items*** tree click the checkboxes of *items* you want to add to the group you are defining.

   — To add collection references, in the **Collection References** tree click the checkboxes of collections you want to add (by reference) to the collection you are defining.

   When you select a collection on the left, you see details on the right: For example, the path to item you selected, and the names of collections you selected.

3. For function collections only, reorder the functions as needed, so that they execute in the correct order at runtime (that is, at startup or shutdown). Highlight a rule function in the tree on the left, and then click Move Up or Move Down as needed.
4. For destination collections only, configure each destination in turn. Select the destination on the left and complete the settings on the right to define characteristics such as the threading model to use, and the event preprocessor. See CDD Collections Tab Input Destination Settings Reference on page 472 for information about each setting.
5. Save the CDD.

To Update Collections

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

CDD Collections Tab Input Destination Settings Reference

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

Table 51 CDD Collections Tab Input Destination Settings (Sheet 1 of 2)

Property	Notes
Input Destination ID	Uniquely identifies this destination configuration at runtime. Edit as needed to ensure that each destination in the cluster has a unique deployment name. Default value is destination name.
URI	Project path to the destination (that is path to the destination in the design-time project).
Preprocessor	Specifying an event preprocessor for a destination is optional. **Tip** If you specify a preprocessor, you generally also also specify worker thread settings, because event preprocessors are multithreaded (unless Caller's Thread threading model is used, which is single threaded). Select the rule function that has been configured as this destination's event preprocessor. Event preprocessors are rule functions with one argument of type simple event. See Event Preprocessors on page 265.

Table 51 CDD Collections Tab Input Destination Settings (Sheet 2 of 2)

Property	Notes
Threading Model	
	If you specified a preprocessor, also specify thread settings. Select one model: **Shared Queue** Uses the TIBCO BusinessEvents system-wide shared queue and threads. For queue size and number of threads settings, see CDD Agent Classes Tab Settings Reference on page 476. **Caller's Thread** Uses the thread (and queue size) provided by the channel resource client. There is one thread per destination. **Note** If it is important to ensure that acknowledgements are sent in the expected order with Caller's Thread threading model, do not use the parallel operations option. See `Agent.`*agentClassName*`.enableParallelOps` in Table 53, CDD Agent Classes Tab Inference Agent and Query Agent Properties, on page 479. **Destination Queue** TIBCO BusinessEvents creates a dedicated thread pool and set of worker threads in each destination. See Thread Count and Queue Size below. For more information on threading models see Chapter 10, Threading Models and Tuning, in *TIBCO BusinessEvents Architect's Guide*.
Thread Count	
	If you specified Destination Queue in the Threading Model setting, specify the number of threads for this destination here.
Queue Size	
	If you specified Destination Queue in the Threading Model setting, specify the queue size for this destination here.

Configuring Agent Classes (All OM Types)

TIBCO BusinessEvents Express Content relating to Cache OM and backing store does not apply to TIBCO BusinessEvents Express edition.

Agent class types are as follows:

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

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

Query Agent Used with Cache OM only and available only if TIBCO BusinessEvents Event Stream Processing software is used.

Dashboard Agent Used with Cache OM only and available only if TIBCO BusinessEvents Views software is used. See *TIBCO BusinessEvents Views Developer's Guide* for details.

Monitoring & Management Agent Used internally by the Monitoring and Management component. Do not add any agents of this class. Configuration of this agent type is explained in Chapter 3, Basic MM Configuration of *TIBCO BusinessEvents Administration*.

Process Agent Used with Cache OM only and available only if TIBCO BusinessEvents Process Orchestration software is used.

To Add an Agent Class

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

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

3. The new agent class name appears on the left. Select the agent class name. Appropriate settings for that agent type appear in the Configuration section.

   — Complete the settings as explained in CDD Agent Classes Tab Settings Reference on page 476

   — Add any properties as needed. The available properties are explained in CDD Agent Classes Tab Properties Reference on page 479.

4. For inference, dashboard, and query agent class types, add the resources you want to use. In the agent class tree on the left, click each type of resource collection in turn and configure as explained next.

   (In the instructions below, the word *item* stands in for destination, function, and rule depending on the collection category.)

   a. Highlight a category of collections (for example Input Destination Collections).

   b. Click **Add**. You see the Select *items* dialog.

   c. In the upper section of the dialog, select individual project *item* resources, as desired.

   d. In the lower section of the dialog (the Reference Groups section), select *item* collections you defined earlier, as desired.

   e. Click **OK**. A list of *item* IDs appears in the box on the right.

5. If you added any individual destinations to the Input Destinations Collections category, highlight their name on the left and configure their settings on the right. See CDD Collections Tab Input Destination Settings Reference on page 472 for details.

   (Destinations within input destination collections are configured at the Collections tab.)

6. Do any of the following as needed:

   — Click a collection category on the left to see a list of collections and *items* you selected from that category on the right.

   — Expand a category on the left and click a collection reference within it. You see a list of its item IDs and paths, and any collection references within that collection, on the right.

   — Edit the project paths for individual items you add here. You would do this only if the project location of that item changed.

7. Save the CDD.

CDD Agent Classes Tab Settings Reference

TIBCO BusinessEvents Express Content relating to Cache OM and backing store does not apply to TIBCO BusinessEvents Express edition.

The tables below explain settings used with inference agents and query agents. Query agents are used with the TIBCO BusinessEvents Event Stream Processing add-on.

Dashboard agents are used with the TIBCO BusinessEvents Views add-on and are documented in *TIBCO BusinessEvents Views Developer's Guide*.

Table 52 CDD Agent Classes Tab Inference Agent and Query Agent Settings (Sheet 1 of 3)

Setting	Notes
Inference Agent and Query Agent Settings	
Max Size (Local Cache)	
	Specifies the maximum number of objects (entities) in each agent's L1Cache (inference agent) or local cache (query agent). The L1 cache is a local cache used by the inference agent for local access to recently used objects. It is used to optimize access to objects. The query local cache is used in a way similar to the inference agent L1Cache. The query agent's local cache stores cache data locally for efficient reuse. The local cache listens to and synchronizes the locally stored entity instances with those in the main cache, so that the local cache stays up-to-date. When the threshold is reached, oldest entities are removed first. Default is 1024 (unit is objects).
Eviction Time (Local Cache)	
	Specifies an age limit on the cached entities in seconds. After this period, they are removed from the local cache. **Note** Age resets each time an entity is accessed by a query engine. Default is 900.

Table 52 CDD Agent Classes Tab Inference Agent and Query Agent Settings (Sheet 2 of 3)

Setting	Notes
Queue Size (Shared Queue)	Used for destinations whose threading model is Shared Queue (see Threading Model in Table 51, CDD Collections Tab Input Destination Settings, on page 472). Specifies the queue size for the processing unit-wide shared queue **Note**: In this release, set the same value for all agents configured to deploy in the same processing unit. If set to 0 (zero), the queue size is unlimited. Default is 1024
Thread Count (Shared Queue)	Used for destinations whose threading model is Shared Queue (see Threading Model in Table 51, CDD Collections Tab Input Destination Settings, on page 472). Specifies the number of processing unit-wide shared threads. **Note**: In this release, set the same value for all agents configured to deploy in the same processing unit. As a guideline, set the value to the number of processors available to the JVM. In MM Console, this thread appears with the name `$default.be.mt$`. Default value is 10.
Max Active	Specifies the maximum number of active agents of this class. This value is used for fault tolerance. Deployed agents that are acting as standbys can take over from active instances that fail. In many cases, there is no need to keep standby instances. A value of 0 indicates an unlimited number of active instances. See Fault Tolerance of Agents in *TIBCO BusinessEvents Architect's Guide* for more details. Default is 0.

Table 52 CDD Agent Classes Tab Inference Agent and Query Agent Settings (Sheet 3 of 3)

Setting	Notes
Inference Agent Settings	
BusinessWorks Repo URL	
	If this project will integrate with a TIBCO ActiveMatrix BusinessWorks project, enter the Repo URL for the ActiveMatrix BusinessWorks project repo URL here. Use forward slashes. See Task A, Add the Repo URL for BusinessWorks to the CDD, on page 619 for more details about the value to use.
Concurrent RTC	
	If checked, enables concurrent run to completion cycles, generally shortened to RTC cycles. (Also known in prior releases as concurrent Rete and `concurrentwm`). Concurrent RTC does not require cache OM but does require local locking. The number of concurrent cycles is determined by the number of available threads. See Chapter 29, Agent and Processing Unit Configuration for details. Also see Chapter 9, Concurrency and Project Design in *TIBCO BusinessEvents Architect's Guide* for important information on using concurrency features.
Check for Duplicates	
	By default, TIBCO BusinessEvents checks if the external IDs (`@extId`) of entities are unique within the current agent. If you want to check for uniqueness of external IDs across the cluster, check this check box. Performing this check affects performance. Default is unchecked.

CDD Agent Classes Tab Properties Reference

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

Table 53 CDD Agent Classes Tab Inference Agent and Query Agent Properties (Sheet 1 of 5)

Setting	Notes
Inference Agent and Query Agent Properties	
`com.tibco.cep.runtime.channel.payload.validation`	
	XML event payloads are validated when this property is set to true. There may be some loss of performance due to the extra processing. Default is `false`.
Inference Agent Properties	
`Agent.`*AgentClassName*`.recoveryPageSize`	
	Specifies the number of entries per page to be used while recovering objects from the cache. For example, if you set the value to 10,000, then the engine loads handles in blocks of 10,000, instead of trying to load them in a single batch. Smaller batch sizes result in slower recovery. Experiment with batch size to establish the best batch size to use for your environment. A value of 0 means that the objects are recovered in one iteration. Default is 0.

Table 53 CDD Agent Classes Tab Inference Agent and Query Agent Properties (Sheet 2 of 5)

Setting	Notes
Agent.*AgentClassName*.cacheTxn.updateCache	Used only if cache-aside database write strategy is used. If set to false: When a rule action changes the value of an entity’s properties, then the entity instance is evicted from the cache instead of updating it. Updates are saved in the backing store. Use this setting and Agent.*AgentClassName*.threadcount as needed to improve performance and cache memory management. This property interacts with the Cluster > Domain Objects setting, Evict From Cache on Update (and its override settings if any): • When this CDD property is set to true, the domain objects Evict From Cache on Update setting is ignored, in the agent for which the property is set. • When this CDD property is set to false, the domain objects Evict From Cache on Update setting overrides this CDD property. See CDD Cluster Tab Domain Objects Default Settings Reference on page 459 for details on the Evict From Cache on Update setting. Possible values are true and false. Default is true.
Agent.*AgentClassName*.threadcount	For use with cache aside and only when parallel operations feature is used (see Agent.*agentClassName*.enableParallelOps). Defines the number of $CacheWriter threads performing cache writing jobs. See Chapter 10, Threading Models and Tuning in *TIBCO BusinessEvents Architect’s Guide* for usage guidelines. **Tip** This property is also used to define the number of Recovery threads (used for recovering Cache Plus Memory entity handles at inference engine startup). Default value is 2.

Table 53 CDD Agent Classes Tab Inference Agent and Query Agent Properties (Sheet 3 of 5)

Setting	Notes
`Agent.`*AgentClassName*`.checkDuplicates`	Affects how TIBCO BusinessEvents checks uniqueness of entity external IDs (@extId). If set to false, checks for uniqueness of external IDs within the agent If set to true, checks for uniqueness of external IDs across the cluster. Performing this check affects performance so use it with care. Default is false.
Inference Agent Database Writer Thread Tuning Properties **Note** For use with cache aside and only when the parallel operations feature is used. For a guide to usage, see the section Database Write Tuning Options for Cache Aside in Chapter 10, Threading Models and Tuning in *TIBCO BusinessEvents Architect's Guide*.	
`Agent.`*agentClassName*`.enableParallelOps`	**If true, parallel operations are used** Post-RTC phase operations are done in parallel: • Writes to the cache • Writes to the database (relevant only cache aside strategy is used) • Executes the actions list, for example, sends messages (events) and acknowledges events, as needed. Use of parallel operations generally requires use of locking to ensure data integrity. **If false, sequential operations are used** All post-RTC phase operations are done on a single thread in the order shown above. This property is set to false for specific needs such as when Caller's Thread threading option is used. For an example, see Using CLIENT_ACKNOWLEDGE Mode with Websphere MQ and Cache-Aside on page 87. Another reason to set the value to false is to ensure that the system waits to send a reply event confirming that some work has been done, until the result of the work can be seen in the cache. Defaults to true only if cache-aside write strategy and concurrent RTC are both used. Otherwise defaults to false.

Table 53 CDD Agent Classes Tab Inference Agent and Query Agent Properties (Sheet 4 of 5)

Setting	Notes
`Agent.`*AgentClassName*`.dbthreadcount`	Defines the number of database write threads available to process the RTC transactions from the queue, that is, the number of threads performing database writing jobs (`$DBWriter` thread pool). Writes include applying entity inserts, updates, and deletes to the database. Although the limit is seldom reached, you can guarantee that a connection is always available for a `dbwriter` thread as follows. Set this field to the same value as CDD Cluster tab > Backing Store > Connection > Max Size field. Default is 2.
`Agent.`*AgentClassName*`.dbOpsQueueSize`	The size of the queue (a Java blocking queue) for database writing jobs. Zero (0) or a negative value means the queue size is unlimited. **Note** When the queue is full, all engine operations are blocked. Default is 8.
`Agent.`*AgentClassName*`.dbOpsBatchSize`	Used in the post-RTC phase. Sets the maximum number of RTC transactions that a database writer thread takes from the database operations queue and processes in one batch. Database write threads process the RTC transactions from the queue. The number of threads is defined by `dbthreadcount`. A database write thread takes *up to* the `dbOpsBatchSize` number of RTC transactions, processes them and commits them to the database. (When database write threads are idle, they take available jobs from the database operations queue, even if there are less jobs than `dbOpsBatchSize`.) Default is 10.

Table 53 CDD Agent Classes Tab Inference Agent and Query Agent Properties (Sheet 5 of 5)

Setting	Notes
Query Agent Properties	
`be.agent.query.localcache.prefetchaggressive`	
	If set to true, then objects required for a query are prefetched while the query is executing. The prefetch feature improves performance, but CPU and memory usage increases as a result of the aggressive prefetching. You may have to try different values till you find the optimal settings for your environment. Ensure that the cache size is large enough to accommodate objects that are prefetched. Default is `false`.
`be.agent.query.enable.filter.optimizer`	
	Used only with Oracle Coherence as the cache provider. If set to true, the query agent attempts to use indexing that is enabled and defined as explained in the section Indexing for More Efficient Cache Queries on page 294. If set to false, indexes are ignored by this agent. See also Present in Index, in CDD Cluster Tab Domain Objects Default Settings Reference on page 459. Default is false.
Cache Agent Properties	
`be.engine.cacheServer.channel.disable`	
	By default cache agents connect to channels. In most cases, however, cache agents do not need to connect to channels. To prevent cache agents from connecting to channels, set this property to true. Default is false.

Understanding and Configuring Log Configurations

Each processing unit references a log configuration. The log configurations are defined in the Collections tab. See Configuring Collections of Rules, Rule Functions, and Destinations on page 470 for related procedures. This section explains more about log configurations and how to configure them.

You can also replace the default line layout implementation with your own.

Log File Settings

For a reference to the settings, see the Files Section in Table 54, CDD Collections Tab Log Configurations Settings, on page 487.

Log File Name and Location

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

Number and Size of Log files

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

Understanding Log Configuration Levels and Syntax

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

Levels

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

Level	Description
Off	Highest possible rank. Filters out all logging messages (turns logging off for the specified module).

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

Syntax

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

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

module1:*level* *module2*:*level* . . .

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

```
*:info
```

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

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

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

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

Configuring Log Configurations

See Understanding and Configuring Log Configurations on page 484 for an explanation of the logging levels, modules, and syntax details

To Add a Log Configuration

1. In the Collections tab select Log Configurations and click **Add**.

2. In the Configuration section, give the log configuration a name.
3. Add the log levels you want to enable in this configuration. See CDD Collections Tab Log Configurations Settings Reference on page 487 for details on the fields.
4. If you want to redirect the STDERR or STDOUT streams, check the Enable checkbox and follow guidelines in the Send to Terminal Section on page 488.
5. Save.

To Specify a Custom Line Layout Class

1. In the Collections tab select Log Configurations and click **Add**.
2. In the Configuration section, give the log configuration a name.
3. In the Custom Line Layout section, check the Enable checkbox, and complete the Class and Argument fields as shown in Table 54, CDD Collections Tab Log Configurations Settings, on page 487.
4. Save.

CDD Collections Tab Log Configurations Settings Reference

Additional log settings can be added to the processing Unit tab Properties. See CDD Processing Units Tab Settings Reference on page 494.

Table 54 CDD Collections Tab Log Configurations Settings (Sheet 1 of 3)

Property	Notes
Name	Name of this log configuration.
Enable	Check the Enable checkbox to enable this log configuration. All other Enable settings are ignored if this checkbox is unchecked.
Levels	Space-separated list of levels and modules used in this log configuration. See Syntax on page 485 and other sections in introduction to Understanding and Configuring Log Configurations on page 484 Default is info
Files Section	
Enable	Check the Enable checkbox to enable log files to be written. Configure the settings in this section to specify details. If this checkbox is unchecked, all other properties in this section are ignored.
Directory	Enter the absolute path to the directory in which you want to store the files. If you do not enter a leading slash, the files are stored relative to the working directory (the directory in which you start the `be-engine.exe` executable).
Name	Name of the log file. The default value is the engine name. If no engine name is set, then the default value is `cep-engine.log`

Table 54 CDD Collections Tab Log Configurations Settings (Sheet 2 of 3)

Property	Notes
Max number	Number of log files to keep. When the Max size setting value is reached, a new log file is created for the next log entries. Files are created up to the Max number setting size. The oldest file is deleted when a new file is added after this value is reached. Default is 10.
Max size	Maximum size of one log file. Default is 10000000.
Append	If checked then new entries are added to the end of the file. If not checked, the contents of the file are flushed each time the engine starts.
Send to Terminal Section	
Enable	Check the Enable checkbox to enable the redirections specified in this section. If this checkbox is unchecked, all other properties in this section are ignored.
Output redirection	If true, the `STDOUT` stream is written to the terminal. If false, it is not.
Error redirection	If true, the `STDERR` stream is written to the terminal. If false, it is not.
Custom Line Layout Section	
Enable	Check the Enable checkbox to enable the custom line layout entries to take effect. Configure the settings in this section to specify details of a custom layout. If this checkbox is unchecked, all other properties in this section are ignored. If this checkbox is checked all properties in the other sections are ignored (except Name, and Enable in the upper section).

Table 54 CDD Collections Tab Log Configurations Settings (Sheet 3 of 3)

Property	Notes
Class	The custom line layout class. This class must implement `org.apache.log4j.Layout` and must be available in the runtime classpath. The class needs 2 constructors: • One with no argument • One with a single String argument, which receives the value of the Arguments field.
Arguments	A String parameter used for the custom line layout class, if required: • To use the constructor that requires an argument, specify the argument • To use the constructor that does not expect an argument,leave the field empty.

Logging for TIBCO BusinessEvents DataGrid

You must configure logging for TIBCO BusinessEvents DataGrid in TIBCO BusinessEvents separately. To do so, set the following properties in the CDD file:

Property	Description
be.engine.cluster.as.log.dir	The directory to which the TIBCO BusinessEvents DataGrid log files will be written. If unspecified, the logs will be written to the same directory as the TIBCO BusinessEvents logs.
be.engine.cluster.as.log.filename	File name of the log file. By default, the file name is as-<*pid*>.log
be.engine.cluster.a.log.level	The log level specifying how much logging is to be filtered out. See Log Configuration Levels for TIBCO BusinessEvents DataGrid Logging for details. By default, the log level is set to INFO.

Log Configuration Levels for TIBCO BusinessEvents DataGrid Logging

Following levels can be used to specify the level of logging for TIBCO BusinessEvents DataGrid logs. Note that the logging levels are case insensitive.

Level	Description
None	Highest possible rank. Filters out all logging messages (turns logging off for the specified module).
Fatal	Logs only severe runtime errors that cause the application to stop running.
Error	Also logs runtime errors that might not cause the application to stop running.
Warn	Also logs potentially harmful runtime events or situations.
Info	Also logs runtime informational events of general interest.
Fine	Also logs detailed runtime informational events, for use in identifying issues.
Finer	Also logs even more detailed runtime information.

Level	Description
Finest	Lowest possible rank. Turns on all logging including any custom logging levels.

Configuring Processing Units (All OM Types)

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

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

Processing units are referenced in the site topology file used by TIBCO BusinessEvents Monitoring and Management component (see Chapter 3, Basic MM Configuration in *TIBCO BusinessEvents Administration*).

PUs with Unique Agent Instance Properties

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

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

Agent Priority The agent priority determines which agents of a given class are active, when fault tolerance is used. Each deployed agent of an agent class must have a different priority.

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

To Add a Processing Unit

See CDD Processing Units Tab Settings Reference on page 494 for guidelines on the settings and properties.

1. At the Processing Units tab do the following:
   - Select the default processing unit and configure it. You can rename it as needed.
   - Click **Add** to add more processing units as needed.

2. In the Name field, enter the name for the processing unit as needed.

   For deployment, TIBCO Administrator by default looks for a processing unit called default and a CDD file called default.

3. In the Log Configuration field, browse to and select one log configuration. See Understanding and Configuring Log Configurations on page 484 for more details on log configurations.

4. Check the Hot Deploy checkbox if you want to enable hot deployment. See *TIBCO BusinessEvents Administration* for details about hot deployment.

5. If you use the TIBCO BusinessEvents Data Modeling add-on, check the Enable DB Concepts checkbox to enable database concept functionality on this processing unit as desired.

6. In the Agents section, click **Add** and select an agent class.

7. If needed, assign to each agent a key and a priority. See PUs with Unique Agent Instance Properties on page 492 for details.

8. In the Properties section, add any additional configuration properties as required. For example see Localhost and Localport Properties on page 420 for one use case.

CDD Processing Units Tab Settings Reference

Table 55 CDD Processing Units Tab Settings (Sheet 1 of 2)

Property	Notes
Name	Enter a name that is unique across the cluster.
Log Configuration	Browse to and select a log configuration, configured at Collections tab. See Understanding and Configuring Log Configurations on page 484 for more details.
Hot Deploy	Check the checkbox to enable hot deployment for this processing unit. See *TIBCO BusinessEvents Administration* for details about hot deployment.
Enable Cache Storage	Check the checkbox to enable cache storage on this processing unit (PU). Settings available depend on the types of cache agents in the PU. • In PUs used to host cache agents: The checkbox is checked and cannot be unchecked. • In PUs used only for dashboard agents (available in TIBCO BusinessEvents Views only): the checkbox is unchecked and cannot be checked. Dashboard agents cannot function as cache servers. • in PUs that host inference agents or query agents (or both): — If checked, the PU is used for storing cache data. — If unchecked, the PU is not used for storing cache data. **Note**: Enable cache storage in PUs running inference and query agents for test deployments only. Not recommended in production. Default value for PUs containing inference agents or query agents (or both) is unchecked.
Enable DB Concepts	Check the checkbox to enable database concepts functionality for this processing unit. Available only with TIBCO BusinessEvents Data Modeling add-on software.

Table 55 CDD Processing Units Tab Settings (Sheet 2 of 2)

Property	Notes
Agents Section	
Agent	Name of the agent class you selected. Agent classes are defined at the Agent Classes tab.
Key	Specifies a value that uniquely identifies an instance of an agent of this class at deploy time. Required for recovery of scorecards. Recommended in all cases, for situations that require an agent instance to be uniquely identified. The value for Key must uniquely identify the agent. **Note**: In certain TIBCO BusinessEvents Monitoring and Management methods, you may be prompted for a session name. For session name you generally put the agent class name. However, if the agent class also has a key, you must instead use the key value. Because of this, the key value must uniquely identify the agent. No default value.
Priority	Specifies the priority of the agent for load balancing purposes. (Not available in TIBCO BusinessEvents Express.) The priority indicates the order in which standby agents become active, and conversely, the order in which active agents become standbys, when new agents join the cluster. The *lower* the number, the higher the agent is in the activation priority list. For example, an agent with priority 2 has a higher priority than an agent with a priority of 6. Ensure that inference agents of the same class in different PUs have different values. This value determines the order of each instance of an agent class for startup, as well as failover and failback in fault tolerance situations. No default value.

CDD Processing Units Tab Coherence Log Properties Reference

The properties in the Coherence Log Properties section are used to configure the Coherence log. This log is used only if Oracle Coherence is used as the cache provider.

Standard logging settings are configured in the Log Configuration tab. See CDD Collections Tab Log Configurations Settings Reference on page 487.

Table 56 CDD Processing Units Tab Coherence Properties (Sheet 1 of 3)

Property	Notes
Coherence Log Properties	
`tangosol.coherence.log`	Specifies the output device used by the logging system. Optional. Possible values are: • `stdout` • `stderr` • `jdk` (Requires JDK 1.4 or later) • `log4j` (Requires log4j libraries to be in the classpath) • A file path and file name If you specify `jdk` or `log4j` you must also perform appropriate configuration of the JDK or Apache `log4J` logging libraries. Default is `stdout`.

Table 56 CDD Processing Units Tab Coherence Properties (Sheet 2 of 3)

Property	Notes
`tangosol.coherence.log.level`	
	Specifies which logged messages are output to the log destination. Optional. Possible values are: • 0: Only output without a logging severity level specified will be logged • 1: All the above plus errors • 2: All the above plus warnings • 3: All the above plus informational messages • 4-9: All the above plus internal debugging messages (the higher the number, the more the messages) • -1: No messages Default is 5.
`tangosol.coherence.log.limit`	
	Specifies the maximum number of characters that the logger daemon processes from the message queue before discarding all remaining messages in the queue. The message that causes the total number of characters to exceed the maximum is not truncated. All discarded messages are summarized by the logging system with a single log entry detailing the number of discarded messages and their total size. When the queue empties, the logger is reset and subsequent messages are again logged. The purpose of this setting is to avoid a situation where logging can itself prevent recovery from a failing condition, for example by contributing to timing issues. Logging occurs on a dedicated low-priority thread to further reduce its impact on the critical portions of the system. Optional. Possible values are positive integers or zero (0). Zero implies no limit. Default is 0

Table 56 CDD Processing Units Tab Coherence Properties (Sheet 3 of 3)

Property	Notes
JMX Management properties You must specify the following JMX-related properties to enable cluster statistics to appear in the monitored objects table in MM. (MM is not available in TIBCO BusinessEvents Express.)	
`tangosol.coherence.management`	
	Default is all
`tangosol.coherence.management.remote`	
	Default is true

CDD Processing Units Tab JMS Server Connection Properties

The following properties relate to JMS channels. Most enable you to define how TIBCO BusinessEvents attempts to reconnect to a JMS server in the event of a disconnection. Two relate to disabling connections. See Chapter 5, JMS Channels, on page 79 for details about configuring a JMS channel.

You can add these properties at the cluster level if they apply to JMS channels in all processing units in the cluster.

Table 57 CDD Processing Units Tab Properties for Reconnecting to a JMS Server

Property	Notes
`com.tibco.tibjms.connect.attempts`	Specifies the number of reconnection attempts, and the interval between each attempt to connect to the JMS server. The value must use the format: *attempts*, *retry interval.* For example: `10,500` means 10 attempts, with a 500 millisecond interval between each retry attempt. This property is used only for channels that have a TIBCO Enterprise Message Service provider. **Note**: Use *either* `be.jms.reconnect.timeout` or `com.tibco.tibjms.connect.attempts`. If you set both the properties, then `com.tibco.tibjms.connect.attempts` takes precedence. Default is 2, 500
`be.jms.reconnect.timeout`	Specifies the retry interval (in seconds) for reconnecting to the JMS server when the connection is broken. A value of zero (0) means do not retry. Any other value means keep retrying (with no limit to number of retries), and use the specified interval between each attempt. **Note**: Unacknowledged messages (Events) are resent to the TIBCO BusinessEvents engine, which may result in duplicate events. **Note**: Use *either* `be.jms.reconnect.timeout` or `com.tibco.tibjms.connect.attempts`. If you set both the properties, then `com.tibco.tibjms.connect.attempts` takes precedence. Default is 0 (zero)

Table 57 CDD Processing Units Tab Properties for Reconnecting to a JMS Server (Cont'd)

Property	Notes
`be.jms.reconnect.msgCodes`	Specifies a case-insensitive character pattern that matches all error messages or error codes that will cause a reconnect attempt. This property is used for JMS channels with providers other than TIBCO Enterprise Message Service. Default is * (that is, the wildcard matched by any characters.)

`be.channel.tibjms.queue.disabled`
`be.channel.tibjms.topic.disabled`
	By default, be-engine connects to all defined channels on startup, including those not mentioned in the CDD file. This is because such channels can be used as output channels. However this is not always desired.
	To disable queue or topic connections for specific JMS channels, add the following properties as appropriate. Enter the project path to the JMS channel as the individual value. Use commas or spaces as the delimiter. Use forward slashes in the project path. For example:
	`be.channel.tibjms.queue.disabled=channels/1jmschannel, channels/3jmschannel`
	`be.channel.tibjms.topic.disabled=channels/2jmschannel channels/4jmschannel`

Chapter 30 JDBC Backing Store Setup

A backing 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.

For instructions on migrating from the legacy Oracle Types (Oracle only) backing store to the JDBC backing store, see the migration chapters in *TIBCO BusinessEvents Installation*.

TIBCO BusinessEvents Express This chapter relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

Topics

JDBC Backing Store Setup and Configuration Overview

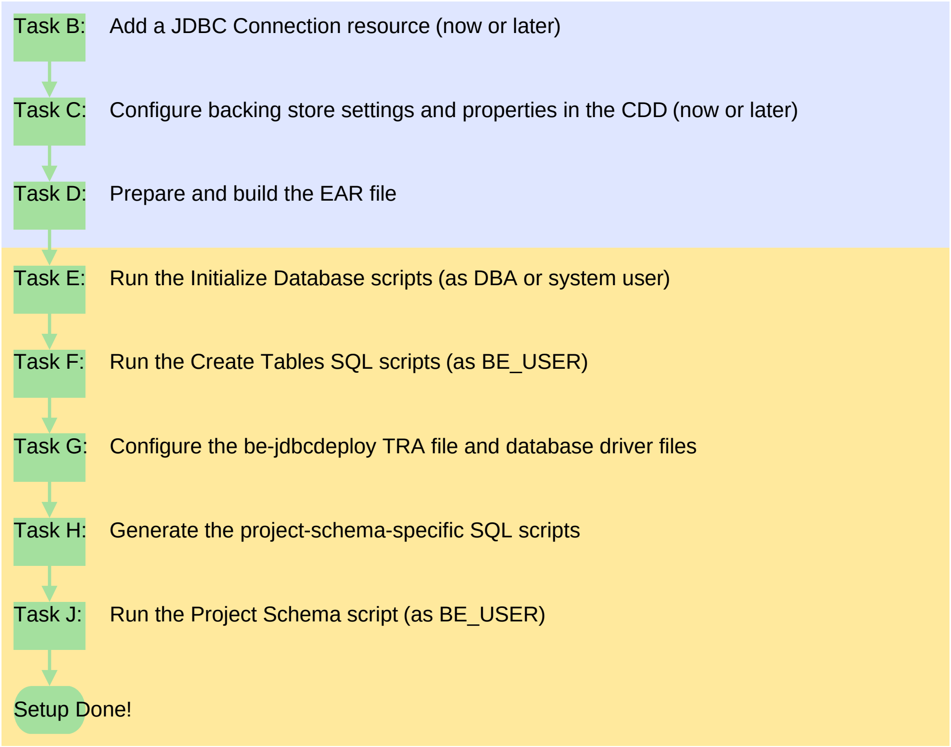
The backing store feature requires use of Cache object management. Before you add a backing store, develop your caching solution and test it. Also ensure that your project ontology is completely configured.

- 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 an Existing Backing Store Schema on page 526.

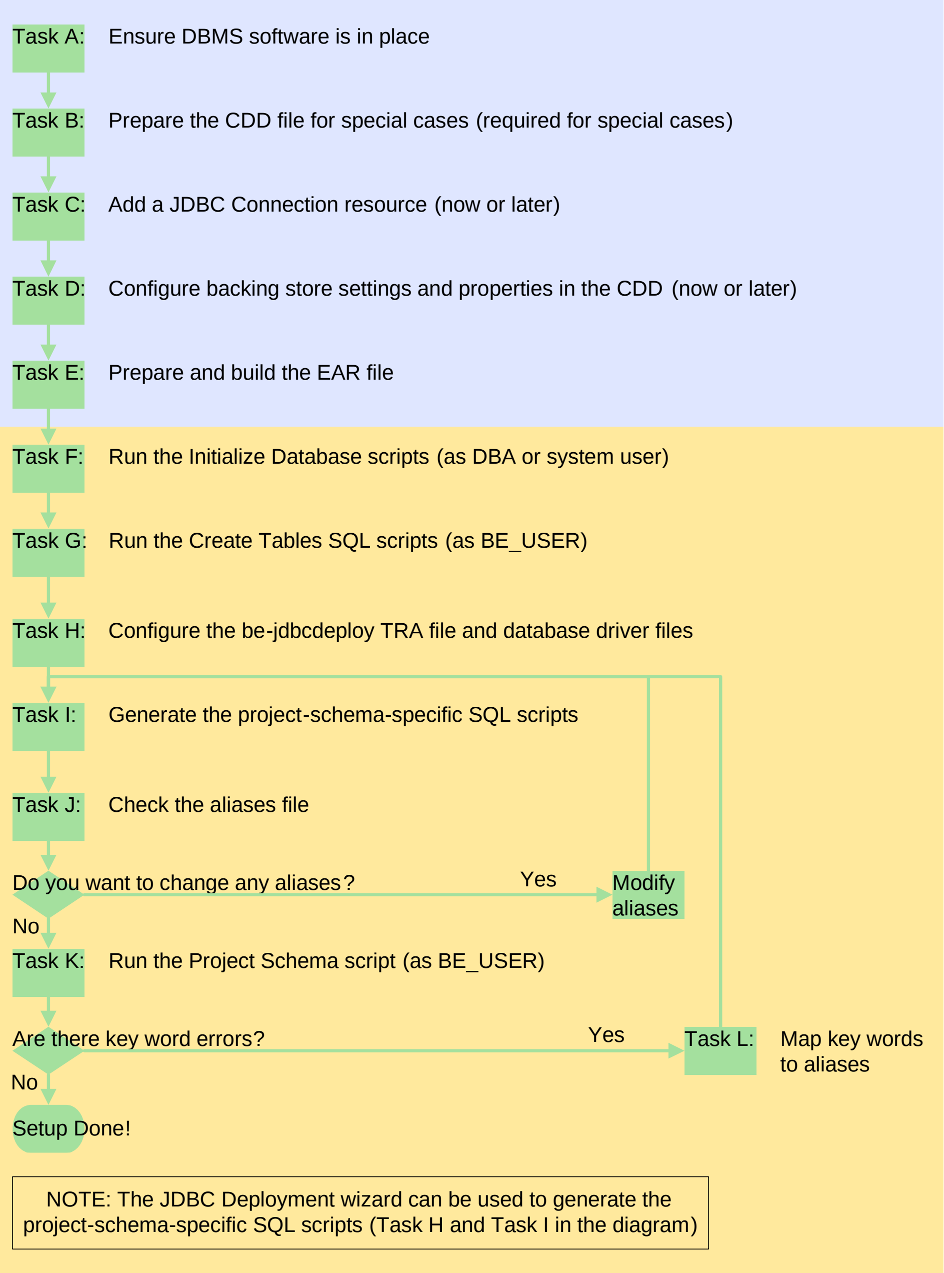
The upper (blue) area in the following diagrams shows TIBCO BusinessEvents Studio configuration tasks. The lower (yellow) area shows database setup utility tasks. The tasks shown map to task sections in this chapter.

Simplest Case

For the simplest case where no additional project configuration is required (see Cases That May Need Additional Setup on page 505), the main tasks in setting up a backing store are as shown next.


Task B: Add a JDBC Connection resource (now or later)
Task C: Configure backing store settings and properties in the CDD (now or later)
Task D: Prepare and build the EAR file
Task E: Run the Initialize Database scripts (as DBA or system user)
Task F: Run the Create Tables SQL scripts (as BE_USER)
Task G: Configure the be-jdbcdeploy TRA file and database driver files
Task H: Generate the project-schema-specific SQL scripts
Task J: Run the Project Schema script (as BE_USER)
Setup Done!

Most Complex Case The complete flow of tasks for all possible situations is shown here. For more information see Cases That May Need Additional Setup on page 505.


Task A: Ensure DBMS software is in place
Task B: Prepare the CDD file for special cases (required for special cases)
Task C: Add a JDBC Connection resource (now or later)
Task D: Configure backing store settings and properties in the CDD (now or later)
Task E: Prepare and build the EAR file
Task F: Run the Initialize Database scripts (as DBA or system user)
Task G: Run the Create Tables SQL scripts (as BE_USER)
Task H: Configure the be-jdbcdeploy TRA file and database driver files
Task I: Generate the project-schema-specific SQL scripts
Task J: Check the aliases file
Do you want to change any aliases?
Yes
Modify aliases
No
Task K: Run the Project Schema script (as BE_USER)
Are there key word errors?
Yes
Task L: Map key words to aliases
No
Setup Done!
NOTE: The JDBC Deployment wizard can be used to generate the project-schema-specific SQL scripts (Task H and Task I in the diagram)

Backing Store Setup

Setup refers to using the provided scripts to create the backing store schema for your project. See Resources Required for Setting Up the Database on page 507 for the DBMS-related requirements.

For the basic setup tasks, see Initialize the Database and Generate Non-Project Tables on page 517. However it is important that you first read the section Cases That May Need Additional Setup on page 505 to understand if you need to do any special project configuration before running the scripts. (Other project configuration can be done before or after you do the setup tasks.)

Backing Store Configuration

In addition to setting up the backing store (and doing related project configuration for special cases), you also configure the TIBCO BusinessEvents Studio project to use the backing store according to your needs. You can do the following to configure backing store behavior:

- Use the Shared All persistence option, with Database Type Oracle or SQL Server. (Formerly known as JDBC backing store.)
- You can use either write-behind or cache-aside database write strategy.
- Tune the database connection pool properties.
- You can use Use a limited or unlimited cache. You can define a global setting and configure object level overrides.
- Control how the cache is preloaded from the backing store at startup. You can define a global setting and configure object level overrides.

See Configure Backing Store Settings in the CDD (Now or Later) on page 514.

You must also add a JDBC Connection resource to your project, before or after the backing store setup.

To make the flow of tasks simpler, all TIBCO BusinessEvents Studio project configuration is documented before the setup tasks, because some TIBCO BusinessEvents Studio configuration is required in certain cases. The procedures make it clear which configuration can be done before or after backing store setup.

Cases That May Need Additional Setup

This section explains some cases that may require additional setup.

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.

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-jdbcdeploy` utility to generate short aliases for long names. You can also edit those names and rerun the utility. For details see Task I, Check the Aliases File and Modify Aliases as Desired, on page 523.

Note that alias file entries are also generated for another reason. See Ontology Identifiers that Use Database Key Words on page 506.

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-jdbcdeploy` 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 Configure the CDD for Special Cases (As Needed) on page 511 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 key word mapping file.

A provided file (*BE_HOME*`/bin/dbkeywordmap.xml`) ships with some basic mappings: `start`, `end`, `schema`, `mode`, and `index`. You can use it as a model.

Unlike the Aliases file, the key word 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 Initialize the Database and Generate Non-Project Tables on page 517.

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 Configure the CDD for Special Cases (As Needed) on page 511 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.

If later you want to include any excluded entities, you must change the setting and update the backing store setup as explained in Updating an Existing Backing Store Schema on page 526.

See Configure the CDD for Special Cases (As Needed) on page 511 for details.

Resources Required for Setting Up the Database

For specific database products and versions supported, see the readme file, which is in the *TIBCO_HOME*/release_notes directory.

Provided Configuration Resources

The table below lists resources required and sections following explain the procedures for setting up backing store tables.

Table 58 Resources Required for JDBC Backing Store Implementation

Resource	Default Location and Notes
Provided Files in *BE_HOME*/bin	
base_types.xml	The base_types.xml file is used by the deployment utility. Do not edit this file.
be-jdbcdeploy executable be-jdbcdeploy.tra	Only used for manual SQL script generation. Generally not needed. You can use a TIBCO BusinessEvents Studio option instead. (See Generated SQL Scripts below).
create_tables_oracle.sql create_tables_sqlserver.sql	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 on page 506 for details.

Table 58 Resources Required for JDBC Backing Store Implementation (Cont'd)

Resource	Default Location and Notes
`initialize_database_oracle.sql` `initialize_database_sqlserver.sql`	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`. **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 > 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-jdbcdeploy` executable. You specify script name prefix at the command line. Scripts are generated in the same directory where you run `be-jdbcdeploy`.	
yourname`.sql`	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 Task I, Check the Aliases File and Modify Aliases as Desired, on page 523.
yourname`_alter.sql`	The *yourname*`_alter.sql` script is for use in schema migration. Generated only after updates are made to the `be-jdbcdeploy.tra` file. See Updating an Existing Backing Store Schema on page 526.
yourname`_cleanup.sql`	For use as needed. This script truncates the tables.
yourname`_delete.sql`	This script is used for deleting entities that have been marked as deleted (applies only if write-behind database write strategy is used).
yourname`_remove.sql`	For use as needed. This script removes the database schema. You can use it to reset the project.

DBMS Software Requirements and Installation

Database Location Instructions in this chapter assume you are working with a local database for testing purposes. Adapt the instructions if you are working with a remote database. For example, 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;
```

Task A Ensure DBMS Software and Drivers are in Place

Before you begin to configure the backing store, you or your DBA must do the following:

- Install and start a supported DBMS product. See the product readme file for a list of supported products.
- Copy the appropriate JDBC drivers file to ***BE_HOME*****`/lib/ext/tpcl`**. You must restart BusinessEvents Studio Explorer 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.
- Now or later: If you will use the debugger or tester features, add your DBMS product’s libraries to the TIBCO BusinessEvents Studio classpath. See the section Working with External Library and Custom Function Paths on page 8.

The remainder of this section provides a few tips for each supported DBMS.

SQL Server

Here are a few helpful points about SQL Server:

- 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.

- 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:

  http://msdn.microsoft.com/en-us/data/aa937724.aspx

Oracle Database

Here are a few helpful points about Oracle Database:

- Use `ojdbc6.jar` drivers file.You can download this file from the following location:

  http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-111060-084321.html
- Maximum length for an Oracle table name or column name is 30 characters.

Configure the CDD for Special Cases (As Needed)

This section summarizes CDD configuration that may be required to handle special cases as described in Cases That May Need Additional Setup on page 505.

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-jdbcdeploy` utility.

Task B Configure the CDD for Special Cases, as Needed

1. In TIBCO BusinessEvents Studio, open the project's CDD file and select Backing Store from the list on the left.
2. In the tree on the left, click **Overrides** and click **Add**. Select the entity or entities you want to customize and click **OK**.
3. 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.

   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 checkbox, and enter the name in the Table Name field. See Ontology Identifiers that Exceed the DBMS Maximum Column Length on page 505 for details, and for an alternative way to handle long names.

   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 on page 506 for details.

4. Save the CDD.

Add a JDBC Connection Resource (Now or Later)

Add a JDBC Connection resource to your project and configure it to connect to the backing store. You can do this before or after you configure the database itself. These settings are ignored by the setup utility during the database setup process.

Details below use Oracle 10g database as an example. Adapt the instructions as needed for your database.

See the section JDBC Connection on page 221 for a reference to the fields.

- The value of the CDD Cluster tab > Backing Store > Connection > Max Size field overrides the value of the JDBC Connection Resource Maximum Connections setting.
- For design time features, such as the Test Connection feature, to work correctly, you must add your DBMS product's libraries to the TIBCO BusinessEvents Studio classpath. See the section Working with External Library and Custom Function Paths on page 8.
- For correct runtime operation, see Updating Classpath, Environment Variables, Path, and Copying JAR Files in *TIBCO BusinessEvents Administration*.

Task C Add a JDBC Connection Resource (Now or Later)

1. In TIBCO BusinessEvents Studio, open your project, and open the folder where you keep shared resources and select **New > Other > TIBCO Shared Resources > JDBC Connection**.
2. At the At the New JDBC Connection Wizard dialog, provide a name for the shared connection and click **Finish**. You see the JDBC Connection resource editor.
3. In the JDBC Driver field, select the driver for your database from the selection box on the right, for example, **oracle.jdbc.OracleDriver**. The driver name appears in the box, and a Database URL format appears to the left of the selection box, according to the driver you selected. For the Oracle thin driver, the format is:

   ```
   jdbc:oracle:thin:@<host>:<port#>:<db_instancename>
   ```

   Check the product readme file to ensure you are using the correct database and driver versions.
4. In the Database URL field, configure the provided format. For example:

   ```
   jdbc:oracle:thin:@localhost:1521:ORCL
   ```

where 1521 is the default port, and ORCL is the default instance name for Oracle Database 10g.

5. In the User Name and Password fields, enter the username and password of the database user (see Task F, Run the Initialize Database Script as the DBA or System User, on page 517).
6. Save the resource.

Test Connection. After you have configured the database and it is running, you can test the connection. Click the Test Connection button. If the database is running and the details are correct, you see a success message.

Before you can use this feature, you must first add the JDBC driver to the project Properties > Build Path > Java Libraries area in TIBCO BusinessEvents Studio. See note in introduction).

Configure Backing Store Settings in the CDD (Now or Later)

You can do the backing store configuration before or after you configure the database itself. These settings are ignored by the setup utility.

Task D Configure Backing Store Settings and Properties (Now or Later)

Reference Tables

Reference tables for all the properties are documented here:

CDD Cluster Tab Backing Store Settings Reference on page 435

CDD Cluster Tab Backing Store Properties Reference on page 439

Configure Database-Related Settings

1. In the CDD editor, select Cluster tab > Object Management > Backing Store.
2. To enable backing store functionality, check the **Enabled** checkbox.
3. In the Database Type field, select **`oracle`** or **`sqlserver`**, depending on the type of database you are using.
4. If you select `oracle` then in the Strategy setting, also select the pooling strategy: **`oracle`** or **`jdbc`**. Connection pool settings are interpreted differently depending on your choice here, as documented in the reference tables listed above.
5. Choose the database write strategy:

   — To use the cache-aside strategy, check the Cache Aside checkbox.

   — To use the write behind aside strategy, uncheck the Cache Aside checkbox.

   See Post-RTC and Epilog Handling and Tuning Options in *TIBCO BusinessEvents Architect's Guide* for more on write strategies.
6. If you want to enforce JDBC connection pool settings and properties, check the **Enforce Pools** checkbox. If you do not want to enforce connection pool settings and properties, uncheck the box.

7. Select **Connection** on the left do the following:

   — Select the JDBC Connection shared resource configured for the backing store.

   — Set the minimum, maximum, and initial sizes of the JDBC connection pool as needed.

   — Add additional properties to the Cluster properties sheet as needed (for example, if you are using Oracle Strategy). See CDD Cluster Tab Backing Store Properties Reference on page 439 for details.

Domain Object Overrides

As desired, you can configure CDD settings related to domain objects (entity instances) such as mode, preloading, limited cache options. See Chapter 28, Domain Objects Configuration, on page 455.

If you change settings that enable or disable the backing store for individual entity types after you have created the backing store schema, you must update the schema, as explained in Updating an Existing Backing Store Schema on page 526.

Build the EAR File

The EAR file is required when you run the database setup utility. Ontology information in the EAR is used to build tables in the database.

Task E Build the EAR File

Ensure that the project ontology is configured correctly and is complete. Ontology information is required by the database setup utility. Other aspects of TIBCO BusinessEvents Studio configuration are ignored by the database setup utility and can be done before or after you run the utility.

1. In BusinessEvents Studio Explorer, highlight the project name, then from the top menus select **Project > Build Enterprise Archive**.

   If you see a message asking you to save all project resources, click Yes. It means an unsaved resource editor is open.

   At the Build Enterprise Archive dialog, you can change the EAR file name, specify the location, and complete any other settings as desired.

2. Click **Apply**, then click **OK**. You see messages as the EAR file builds, then you see a message that the EAR file has built correctly:

Initialize the Database and Generate Non-Project Tables

Ensure that you have done all earlier tasks that may pertain to your case (see Cases That May Need Additional Setup on page 505).

Task F   Run the Initialize Database Script as the DBA or System User

This script creates the TIBCO BusinessEvents user and initializes the database.

Running the `initialize_database`*YourDBMS*`.sql` script deletes the user before creating it again. Running the `create_tables_`*YourDBMS*`.sql` drops all database tables before creating them again. This means you can run these scripts again during 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 an Existing Backing Store Schema on page 526.

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 (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.

   If you are using SQL Server, use this instead:

```
osql -S Your-Server -U sys_user -P sys_user_password -n -i initialize_database_sqlserver.sql
```

   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.
```

Using your database product, you can configure additional users to access the database, in addition to this user.

Task G Run the Create Tables Scripts as the TIBCO BusinessEvents User

Log on as the TIBCO BusinessEvents user, `BE_USER` by default and run a script to create non-project specific tables.

1. Open a command window in the *BE_HOME*`/bin` directory (default location of the scripts), and type the appropriate command for your DBMS at the prompt:

   For Oracle:

```
sqlplus BE_USER/BE_USER@SID @create_tables_oracle.sql
```

   Type **`exit`** to exit and commit.

   If you are using SQL Server, use this instead:

```
osql -S Your-Server -d Your-DB -U BE_USER -P BE_USER -n -i @create_tables_sqlserver.sql
```

   Use the credentials defined in the `initialize_database_oracle.sql or initialize_database_sqlserver.sql` files. By default those are: username `BE_USER`, with password `BE_USER`.

You see various harmless error messages the first time you run the script.

TIBCO BusinessEvents Studio provides a wizard to configure the required properties and generate the project-schema-specific SQL scripts. The wizard is an alternative approach to perform and Task H. See Generating Scripts Using the JDBC Deployment Wizard on page 519 for details.

Generate the Project-Schema-Specific SQL Scripts

You can do this task using a TIBCO BusinessEvents Studio wizard, or you can do it manually.

Before you Begin

- Ensure that you copied your JDBC drivers file to *BE_HOME*`/lib/ext/tpcl` (or other location in your class path). See Task A, Ensure DBMS Software and Drivers are in Place, on page 509.
- Ensure the DBMS is started.

Generating Scripts Using the JDBC Deployment Wizard

You can generate the project-schema-specific SQL scripts using the **JDBC Deployment** wizard in TIBCO BusinessEvents Studio.

Task H Generate the Project-Schema-Specific SQL Scripts (with Wizard)

1. In TIBCO BusinessEvents Studio Explorer, right-click the name of the project for which are creating SQL scripts and select **Export > TIBCO BusinessEvents > JDBC Deployment**. Click **Next**.

Table Creation Scripts

2. You see the **Generate JDBC deployment scripts** wizard. Complete the values as follows.

Field Name	Description
Database Type	Select the type of database you are using from the drop-down list. Choose one of: oracle and sqlserver The default value is `oracle`
Generate ANSI Scripts	Check this checkbox to use ANSI compatible SQL types during script generation. Default setting is checked.
Cluster Deployment Descriptor	Browse to and select the CDD file you want to use for the project.

Field Name	Description (Cont'd)
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, the following scripts are generated: acme.sql, acme.aliases, acme_alter.sql, acme_cleanup.sql, acme_delete, and acme_remove.sql.

3. Do one of the following:

— If you are creating a new JDBC backing store, click **Finish**. The scripts are generated in the location you specified. Information about script generation is also printed to the Console tab.

— If you are migrating an existing JDBC backing store, click **Next**. You see the Generate Migration SQL Scripts page. See step 4 for details.

Migration Scripts

4. At the Generate Migration SQL Scripts page, complete the values as follows.

Field Name	Description
Generate Migration Scripts	Check this checkbox to generate SQL scripts to migrate an existing backing store. Once checked, the rest of the fields on this page are enabled. The default value is unchecked.
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.

Field Name	Description (Cont'd)
Test Connection	Click Test Connection to test if a success

Generating Scripts Manually

This section explains how to generate scripts using a manual procedure, instead of using the TIBCO BusinessEvents Studio wizard.

If you are using SQL Server: Before you execute `be-jdbcdeploy`, open the *BE_HOME*`/bin/be-jdbcdeploy.tra` file for editing. Specify **`sqlserver`**. in the last line:

```
java.property.jdbcdeploy.database.type [oracle | sqlserver]
```

This step requires the EAR file for the project. The database utility uses the project ontology information from the EAR file.

1. Open a command window and navigate to *BE_HOME*`/bin`.
2. Run `be-jdbcdeploy.exe` using a command with the following format:

`be-jdbcdeploy [-h] [-p` *property file*`] [-o` *schema output file*`] [-c` *CDDpath*`]` *EAR Path*

For example:

```
be-jdbcdeploy -o acme -c D:/myproj/acme.cdd D:/ears/acme.ear
```

The generated scripts appear in the directory where you run the executable. For example, if you provide the schema output filename `acme`, you would see files called `acme.sql`, `acme.aliases`, `acme_alter.sql`, `acme_cleanup.sql`, `acme_delete, and acme_remove.sql`.

The user-defined part of the database schema is in the schema output file (*yourname*`.sql`) as schema definition commands. In Task J you run this script (together with provided scripts) to build the schema in the database.

The options are explained in the following table:

Option	Description
`-p`, `/p`, `-property`, or `/property`	Optional. Specifies the property file. If not specified, the default property file is used, that is, `be-jdbcdeploy.tra` in the current directory.

Option	Description
`-o`	Required. Specifies the schema output filename for deployment. **Tip** 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`	Optional. Specifies the absolute path to the CDD file. See Cases That May Need Additional Setup on page 505 to understand when the CDD file is needed.
EARpath	Required. The last option is always the EAR file path.
`-h`, `/h`, or `/help`	Displays this help.
-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.

Check the Aliases File, Run the 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 on page 505 for more information, and for an alternative way to specify short table names.

It's 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.

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 an Existing Backing Store Schema on page 526). 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.

Key word mapping file

Entries in the key word mapping file are also added to the aliases file so you can replace the key word aliases with project-specific ones, as desired (generally in a second pass). For details see Task K, If Needed — Map Key (Reserved) Words to Aliases, on page 524.

Task I Check the Aliases File and Modify Aliases as Desired

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's a good idea to leave any system generated prefixes or suffixes in place for consistency of names across the database.
3. Re-run the `be-jdbcdeploy` tool, using the same parameters as before. (For details see Task H, Generate the Project-Schema-Specific SQL Scripts (with Wizard), on page 519.) This time, the aliases you created are used.

Task J Run the Project Schema Script (as BE_USER)

In this step, you log on as the user you created and run a script to create the project related part of the database schema.

1. Open a command window in *BE_HOME*`/bin` and run the *yourname*.sql script. (for example, `@acme.sql`). Use the user `BE_USER`, password `BE_USER (or` whatever username and password you set in the script in Task F).

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
```

If there are no errors, your database tables are now configured for use. If there are errors you may need to add some mappings to the key word mapping file.

Task K If Needed — Map Key (Reserved) Words to Aliases

Complete this task only if you saw errors after completing Task J, 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.

1. Edit the *BE_HOME*/`dbkeywordmap.xml` file to add entries. Below is the format followed by an example:

   ```
   <keyword name=“dbKeyWord" mapname=“nonDbKeyWord"/>
   ```

   ```
   <keyword name="start" mapname="start_"/>
   ```

2. Repeat Task H, Generate the Project-Schema-Specific SQL Scripts (with Wizard), on page 519, and tasks following as needed.

Providing Project-Specific Key Word Aliases When you repeat Task H, 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 Task H again and continue.

Note that 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. A summary of the basic routine is as follows:

1. *Generate the SQL scripts and run them (as explained in the procedures).*
2. *Errors occur due to key word clashes, so you add the appropriate key word mapping entries to the key word mapping file.*
3. *Generate the SQL scripts 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 scripts again.*
5. *Run the SQL scripts to create the backing store.*

Next Step (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:

- Add a JDBC Connection Resource (Now or Later) on page 512
- Configure Backing Store Settings in the CDD (Now or Later) on page 514

Update your schema if your ontology changes, or if you want to include or exclude different entities in the backing store. See Updating an Existing Backing Store Schema on page 526 for details.

Updating an 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 on page 506 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 and Cannot Handle Automatically on page 527 provides more information.

Preparing to Run and Running the Schema Update Utility

1. To prepare for the update, do the following:

   — Gracefully shut down the deployed application (all agents including cache agents).

   — Back up your existing database.

   — Generate the updated EAR file for the modified project.

   — 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.

2. Open the `be-jdbcdeploy.tra` file for editing and set the following properties:

   ```
   be.jdbc.schemamigration.url=DbURL
   be.jdbc.schemamigration.user=username
   be.jdbc.schemamigration.pswd=password
   ```

   — Use the database URL that points to the existing backing store. See Add a JDBC Connection Resource (Now or Later) on page 512 for example URLs.

   — Use the same username and password you used when setting up the backing store. See Task F, Run the Initialize Database Script as the DBA or System User, on page 517.

   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.

3. Log on as the user name you specified in Task F, Run the Initialize Database Script as the DBA or System User, on page 517. For example,
4. Run the `be-jdbcdeploy` utility as explained in Task H, Generate the Project-Schema-Specific SQL Scripts (with Wizard), on page 519, using the *updated* EAR file.
5. 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 Task I, Check the Aliases File and Modify Aliases as Desired, on page 523. If you modify aliases, remember to generate the scripts again so the modified aliases are used.

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.

6. 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 and Cannot Handle Automatically on page 527 for details.
7. Run the *yourname*`_alter.sql` script.

Your database tables are now configured for use.

What the Schema Update Utility Can and Cannot Handle Automatically

You must examine the alter script before you run it. Decide what changes to make manually and what changes to make using the script, taking into account the kind of data in the tables. Entries that could result in data loss are commented. Remove or comment entries for changes you will make manually.

Adds

The schema migration utility handles 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 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 keep track of 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 alter script and enable these commands if you want to execute 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) );
```

Handling Entities Deleted from a Write-Behind Backing Store

TIBCO BusinessEvents Express This section relates to Cache OM functionality and does not apply to TIBCO BusinessEvents Express edition.

This section pertains only if write-behind strategy is used. See Post-RTC and Epilog Handling and Tuning Options in *TIBCO BusinessEvents Architect's Guide* for more information about database write strategies, and see CDD Cluster Tab Backing Store Settings Reference on page 435 for details on the settings.

When write-behind strategy is used, when entities are deleted from the cache, they are marked for deletion at system startup. An application startup process removes the deleted entities from the backing store.

Depending on the number of deleted entities, the cleanup step can make system startup unacceptably slow.

Manual Deletion of Deleted Entities

To deal with slow startup, you can run a script manually, and eliminate the need to run the cleanup action at system startup. The script is called *yourname*`_delete.sql` and it is generated when you create the backing store with the `be-jdbcdeploy` utility.

Disabling Deleted Entity Cleanup at System Startup

You can also disable the cleanup action performed at system startup and perform all cleanup manually, using the deletion script. To do so, add the following property to the CDD file and set it to false:

```
be.engine.cluster.cleanup
```

Backing 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:

Property	Note
`cacheId`	Entity version number (starts with 1)
`time_created$`	Time when the entity was created
`time_last_modified$`	Time when the entity was last modified
`parent$_id$`	Id of the parent for contained concepts
`id$`	Unique Id of the entity (must be unique across all entities)
`extId$`	Unique (or null) extId assigned
`state$`	Always set to ‘C’ meaning ‘Created’ (reserved for future use)

Secondary Tables

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.

Table 59 Secondary table structure

property Type	Column	Description
Array	`pid$`	Parent ID
	`valPid$`	Array index
	val	Item's value
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'
```

Chapter 31 Introduction to WebStudio

This chapter introduces the TIBCO BusinessEvents WebStudio application and its server component, Rules Management Server (RMS).

Topics

Introduction to TIBCO BusinessEvents WebStudio

TIBCO BusinessEvents WebStudio is a TIBCO BusinessEvents online component consisting of a client, WebStudio, and a server, Rules Management Server (RMS).

WebStudio Component

TIBCO BusinessEvents WebStudio is an online component which allows business users to create/manage business rules in a web browser. In TIBCO BusinessEvents WebStudio user defines an executable rule (business rule) based on the rule template and rule template view defined by developer in TIBCO BusinessEvents Studio. Similar to TIBCO BusinessEvents Decision Manager, a decision table in TIBCO BusinessEvents WebStudio is defined using the virtual rule function. Rule templates, rule template views, and virtual function are created in TIBCO BusinessEvents Studio by developers and stored in RMS repositories.

The decision tables are available only if TIBCO BusinessEvents Decision Manager add-on is installed.

Rules Management Server (RMS) Component

Rules Management Server (RMS) is a server-based component that manages the lifecycle of WebStudio artifacts and WebStudio user access.

Using a menu of options in TIBCO BusinessEvents WebStudio, RMS allows business users to check out projects and check them in, subject to an approval process that RMS also provides. The workflow process can be customized. For a more detailed overview see WebStudio and RMS User Workflow on page 539.

Before RMS can be used, some configuration is required, as explained in Introduction to Rules Management Server (RMS) on page 538.

Technical User Tasks

As a technical user you configure RMS and the TIBCO BusinessEvents WebStudio components, and set up project resources required for business users to create decision tables or business rules. You also deploy decision tables or business rules when they are ready for use, either in an enterprise archive (EAR) file, or as class files. Class files can be deployed for use at startup, or hot deployed. They can also be unloaded from a running engine. You also define access control settings and can act as an approver to ensure that decision tables and business rules created by business users are appropriate for use.

Virtual Rule Functions and Decision Tables

At designtime, technical users add *virtual rule functions* (VRFs) to a TIBCO BusinessEvents project. A VRF has no body, similar to a Java interface. Its implementation is provided using decision tables authored in TIBCO BusinessEvents WebStudio. VRFs are used in the TIBCO BusinessEvents project like any other rule function; they can be called from rules or other rule functions. Here is a simple example:

```
/**
 * @description
 */
virtual void rulefunction Virtual_RF.Applicant_VirtualRuleFunction {
   attribute {
      validity = ACTION;
 }
   scope {
      Concepts.Applicantapplicant;
      Events.ApplicationReceivedapplicationreceived;
   }
   body {

   }
}
```

In TIBCO BusinessEvents WebStudio, business users add decision table resources to VRFs. The decision table provides the body to the VRF, also known as the VRF implementation.

One VRF can have more than one decision table. If a VRF has more than one decision table, functions in TIBCO BusinessEvents determine how the tables are used.

Rule Template and Business Rule

At designtime users add rule template to a TIBCO BusinessEvents project. Rule template is a specialized rule that contains a pre-conditions section which defines the pre-conditions which must be met in addition to the conditions defined by the business user in a business rule.

The Action Context section of a rule template 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). For a businsess rule execution to succeed, a business rule definition must include all Actions and the order should be maintained. 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 only has to enter the binding values to complete the definition. Action context statements are of three types: create, modify, and call, plus arbitrary actions.

Business User Task

Rule Building with Business Rules

As a business user, you check out projects from RMS, build business rules, and submit them for approval.

Business rules provide builder or a user-friendly HTML form to build rules. In builder you specify the conditions in the "when" section and the actions in the "then" section. You can use artifacts and supported operators for building conditions and actions. In user-friendly HTML form, you fill up the values for the conditions and actions.

Rule Building with Decision Tables

As a business user, you check out projects from RMS, build decision tables, and submit them for approval.

Decision tables are available in TIBCO BusinessEvents WebStudio in edit mode only if you have TIBCO BusinessEvents Decision Manager add-on installed on your system.

Decision tables provide a graphical way to build complex business rules. You create table columns by dragging and dropping predefined properties onto the decision table framework. The properties belong to ontology resources defined in the TIBCO BusinessEvents project. However, you can only use the properties specified in the VRF. Columns can be created in other ways too. You then define threshold values (conditions) and actions in the cells of the table. Each row can be thought of as one rule in a table made up of many rules. The individual rules are often straightforward, as in the following examples.

Three Rule Conditions

```
Person.age < Max(20, Parent.age)
Person.creditscore >= Math.function(...)
Person.gender == "female"
```

Three Rule Actions

```
Application.status = "ACCEPTED"
Application.credit = 4000
sendNotification()
```

However, one decision table can consist of hundreds, even thousands of rules each of which is executed only when its specific conditions are satisfied.

Exception Tables

Each decision table can optionally have another table known as an exception table. The purpose of the exception table is purely organizational: it enables you to separate the business logic of the main decision table (added by business users) from any non-business logic (generally added by technical users). For example, in the exception table, you could capture situations where fields are blank or contain invalid values, and define actions that send notifications or set return values. The rows of a decision table plus the rows of its exception table are considered in an RTC. If you prefer, you can put non-business logic in the main table instead of using an exception table.

Table Analyzer

The Table Analyzer feature analyzes decision tables and reports problems, such as uncovered ranges for conditions, uncovered domain entries, different set of actions for identical conditions. Table Analyzer creates a sparse matrix data structure representing an optimized form of decision tables in memory.

Introduction to Rules Management Server (RMS)

RMS is a lightweight rules management server component for managing the repository of TIBCO BusinessEvents projects. RMS provides an easy, secure, and scripted deployment lifecycle. RMS is supported on all platforms that support TIBCO BusinessEvents. Users can access RMS remotely. RMS does not have to be installed on users' machines.

Under certain circumstances, use of RMS is optional. (see *TIBCO BusinessEvents Decision Manager User's Guide*).

Before you can use RMS some basic configuration may be required. You also set up an RMS project for each TIBCO BusinessEvents project that includes decision tables or business rules. See Chapter 32, Configuring RMS and RMS Projects, on page 541.

RMS enables business users to check out projects. The necessary project resources loaded into the cache. If you have the TIBCO BusinessEvents Decision Manager add-on installed, then the artifacts are also copied into the clients machine. When users are finished working with the projects, they submit the business rules and/or decision tables to RMS for approval. (See WebStudio and RMS User Workflow on page 539 for a more detailed overview.)

RMS enables technical users to do the following:

- Configure authentication and access control as needed. See *TIBCO BusinessEvents Administration* for authentication and access control configuration.
- Approve or reject commit requests from Decision Manager and TIBCO BusinessEvents WebStudio users, check on the status of all such requests, keep track of all project versions, and view audit trails of status changes during workflow execution.
  - — Chapter 36, Working with the Approval Workflow in RMS, on page 597.
- Generate project Enterprise Archive (EAR) files or individual decision table classes or rule template instance file for business rule, for deployment, and deploy them.
  - — Chapter 37, Deployable Files Generation, on page 603.

RMS is implemented using TIBCO BusinessEvents. Knowledgeable TIBCO BusinessEvents users can customize it. The RMS projects are located in the *BE_HOME*`/examples/standard/WebStudio` directory. Note that documentation is provided for the product as shipped. If you customize RMS some documentation may not apply.

WebStudio and RMS User Workflow

This section describes how rules authored in TIBCO BusinessEvents WebStudio are made available to a TIBCO BusinessEvents application.

Step 1: TIBCO BusinessEvents Designtime

A TIBCO BusinessEvents user creates a TIBCO BusinessEvents project, adding the ontology, and writing rules that make use of virtual rule functions (VRFs) and Rule Templates. At designtime, VRFs have a signature but no body.

RMS Project Setup

The TIBCO BusinessEvents project for RMS is stored at the location defined by a property in the RMS server configuration. The RMS project requires an access control file.

The RMS server must be running so that the login and workflow actions are available in the TIBCO BusinessEvents WebStudio component.

Step 2: TIBCO BusinessEvents WebStudio

A business user starts the TIBCO BusinessEvents WebStudio component, logs on to RMS, and checks out the project. The business user creates one or more decision tables or business rules and saves the modified project locally, then commits them for approval.

Step 3: Approval

An RMS user working in TIBCO BusinessEvents WebStudio receives the request and reviews the checked-in artifacts and then approves or rejects them. The approved artifacts are available for subsequent checkouts or updates.

Step 4: Generating Deployable Files

An RMS user generates deployable files for resources that are ready for deployment. You can generate EAR files or class files.

Step 5: Deployment

EAR files are deployed in the usual way, as explained in *TIBCO BusinessEvents Administration*. Class files for decision tables and rule template instance files for business rules are deployed by placing them in a configured location recognized by the TIBCO BusinessEvents engines at startup. They can also be hot-deployed.

Chapter 32 Configuring RMS and RMS Projects

This chapter explains the directory structure of the Rules Management Server (RMS). The chapter also explains the configuration tasks for RMS, for RMS projects, and for TIBCO BusinessEvents WebStudio.

Topics

RMS Project Directory Structure

All RMS projects exist in a repository under a *root location* directory and their ACL file exist under a RMS config location, which are *BE_HOME*`/examples/standard/WebStudio` and *BE_HOME*`/rms/config/security` respectively as shipped. The root location and RMS config location are configurable using `ws.scs.rootURL` and `ws.projects.acl.location` properties in the `RMS.cdd` file, respectively.

Adding a Project to RMS

You can create a project in TIBCO BusinessEvents Studio and follow below steps to add the TIBCO BusinessEventsProject to RMS:

1. Under the RMS project root location directory, create a directory with a name appropriate for the project.

   The root location is configurable. See Configuring RMS Server Properties on page 543.

2. Copy the TIBCO BusinessEvents Studio project contents that will be used in Decision Manager add-on or in TIBCO BusinessEvents WebStudio to the project directory.

3. Copy the ACL file for the project (*RMSProjectName*`.ac`) to the RMS config directory. See *TIBCO BusinessEvents Administration* for authentication topics.

   The RMS config location is configurable. See Configuring RMS Server Properties on page 543.

Configuring RMS Server Properties

RMS works out of the box on a local machine. Configuration is required to enable access by remoteTIBCO BusinessEvents WebStudio and Decision Manager clients.

You you may need to make other changes to the server configuration. For example if you change the server location or the location of the RMS project repository, you must update RMS server properties accordingly.

Configuring RMS Server Properties

To configure RMS server properties, you edit the `RMS.cdd` file.

It is not possible to edit a CDD file in TIBCO BusinessEvents Studio outside of its project context. In order to edit the CDD in TIBCO BusinessEvents Studio, you must import the BRMS project into your workspace. When you are done, you must copy the `RMS.cdd` file to its installed locations, as explained below

Whenever you change the `RMS.cdd` file you must restart the RMS server so that it uses the updated values.

To Configure RMS Server Properties

See RMS Server Configuration Property Reference on page 546 for information about each property.

- As with any procedure that changes files, first backup any files that could be affected. The `RMS.cdd` file is located in two places, in the product as shipped:
  - — *BE_HOME*`/rms/project/BRMS/Deployment/RMS.cdd`
  - — *BE_HOME*`/rms/bin/RMS.cdd`
- After you update the `RMS.cdd` file in your workspace, overwrite the existing files with the updated ones, in the above locations (or the ones currently in use if different).
- Ensure that the CDD files in the BRMS project and the RMS startup directory are kept in sync.

1. In TIBCO BusinessEvents Studio, choose **File** > **Import > Existing Projects into Workspace.**
2. Ensure the Copy the projects into workspace check box is checked.
3. Select the following project:

BE_HOME`/rms/project/BRMS`

4. In Studio Explorer, double-click **RMS.cdd** to open it in the CDD editor.
5. In the CDD editor Cluster tab, click **Properties**. In the Configuration panel on the right, expand groups to see the individual properties. Complete the values as explained in RMS Server Configuration Property Reference on page 546.
   - — Enabling Remote Connection to RMS from WebStudio, page 544
   - — RMS Server Configuration Property Reference, page 546
6. When you have finished editing, save the `RMS.cdd` file.
7. In the file system, copy the `RMS.cdd` file from your workspace to the BRMS project and to the RMS server startup location:

   BE_HOME`/rms/project/BRMS/Deployment/RMS.cdd`

   BE_HOME`/rms/bin/RMS.cdd`
8. Restart the RMS server.

Enabling Remote Connection to RMS from WebStudio

As shipped, the `tibco.clientVar.Webstudio/hostname` property is set to localhost. This setting enables the product to be used on a single machine. For production settings take the following actions to ensure that TIBCO BusinessEvents WebStudio users can connect to RMS from remote machines.

To Define the Location of the RMS Server Host for WebStudio

If you have not already done so, specify the RMS server hostname and port as follows.

1. Import the BRMS project into your workspace and open the `RMS.cdd` file for editing. See To Configure RMS Server Properties on page 543 for details.
2. In the CDD editor Processing Units tab, click **WS-Inference**. In the Properties panel on the right, expand the WS property group.
3. Specify the TIBCO BusinessEvents WebStudio hostname and port using the following properties:

   ```
   tibco.clientVar.Webstudio/hostname
   tibco.clientVar.Webstudio/port
   ```
4. Save the `RMS.cdd` file
5. In the file system, copy the `RMS.cdd` file from your workspace to the BRMS project and to the RMS server startup location:

   BE_HOME`/rms/project/BRMS/Deployment/RMS.cdd`

BE_HOME`/rms/bin/RMS.cdd`

6. Restart the RMS server.

RMS Server Configuration Property Reference

See Configuring RMS Server Properties on page 543 for the related procedure.

Properties not documented in this table are either for internal use or are for other types of configuration explained elsewhere in the documentation.

Table 60 RMS Server Configuration Properties

Property	Notes
WS-RMS-Common Property Group	
`ws.scs.rootURL`	
	Specifies the location of the directory which contains the repository of RMS projects. This is the root location for RMS project directories. Default value (as shipped) is *BE_HOME*`/examples/standard/WebStudio`
`rms.checkin.revisionId.initValue`	
	Initial value of the revision number to use for project check ins. Default is `10000`
`ws.artifact.deploy.location`	
	Name of the directory for storing EAR files generated for deployment. Separate directory for each project is created, where EAR files for the project are stored. See also `be.codegen.rootDirectory` below. Default is *BE_HOME*`/rms/shared`
`be.codegen.rootDirectory`	
	Name of the directory for storing class files generated by the Generate Deployable menu option. The directory location is relative to the `ws.artifact.deploy.location` directory. If this property is not present, files are stored in the deployment directory. If this property is present but has no value, a directory is created with the default name `Codegen`.

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
RMS Property Group	
`rms.project.workspace`	
	Not used in this release. Default is `workspace`
`rms.project.decisiondata`	
	Not used in this release. Default is `decisiondata`
`rms.project.deployment`	
	Not used in this release. Default is `deployment`
`rms.workflowstages.config.file`	
	Not used in this release. Default is *BE_HOME*`/rms/config/RoleWorkflowStages.xmL`
`rms.lockworkflowstages.config.file`	
	Not used in this release. Default is *BE_HOME*`/rms/config/LockWorkflowStages.xmL`
`rms.roleArtifactTypes.config.file`	
	Not used in this release. Default is *BE_HOME*`/rms/config/RoleApplicableArtifactTypesConfig.xml`
`rms.external.entities.autodetect`	
	Used internally. Do not change the value.

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
Authentication and Authorization (ACL) Properties	
User authentication topics are common to TIBCO BusinessEvents and the add-on products that use authentication. Common options are more fully documented in *TIBCO BusinessEvents Administration*. The relevant properties are also listed below for your convenience.	
`be.auth.type`	Specifies the authentication mechanism used, `file` or `ldap`. Additional configuration is required. **Note** Authentication topics are documented in the *TIBCO BusinessEvents Administration* guide.
`be.auth.file.location`	Location of the authentication file used for file-based authentication. **Note** Authentication topics are documented in the *TIBCO BusinessEvents Administration* guide. Default is *BE_HOME*`/rms/config/`security`/users.pwd`
`java.security.auth.login.config`	Location of the JAAS login configuration file. You can substitute a different implementation of the JAAS login module than the one provided. **Note** Authentication topics are documented in the *TIBCO BusinessEvents Administration* guide. Default is *BE_HOME*`/rms/config/security/jaas-config.config`.
`ws.projects.acl.location`	Location of the directory used for all ACL (authorization) files. Files must be named using the format *RMSProjectName*`.ac`. Default is *BE_HOME*`/rms/config/security`

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
RMS-GVs Property Group This group of properties override the values of global variables provided in the TIBCO BusinessEvents Studio BRMS project. They are made available as global variables so that they can be overridden at runtime, as needed. You can override the default global variable values in the `RMS.cdd` file (as shown below) for command-line startup. For details on overriding global variable values when deploying with TIBCO BusinessEvents Monitoring and Management or with TIBCO Administrator, see *TIBCO BusinessEvents Administration*. For defining and using global variables in TIBCO BusinessEvents Studio projects, see Working with Global Variables on page 14.	
`tibco.clientVar.RMS/hostname`	Specifies the host name or IP address of the machine where RMS is hosted. Remote clients can connect to the server at this location. Default is `localhost`.
`tibco.clientVar.RMS/port`	Specifies the port number of the machine where RMS hosted. This port is used for listening to client requests. See notes for `tibco.clientVar.RMS/hostname`. Default is `5000`.
`tibco.clientVar.RMS/Approval/adminRole`	The value specified here has Administrator role permissions. When you change the default value, the Administrator role still has Administrator permissions. Default is `Administrator`.
WS Property Group	
`tibco.clientVar.Webstudio/warDir`	Path to the Web archive for the client side representation of the TIBCO BusinessEvents WebStudio. Default is `rms/bin/WebStudio.war`.

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
`tibco.clientVar.Webstudio/hostName`	Specifies the host name or IP address of the machine where RMS server is hosted for WebStudio. Default is `localhost`.
`tibco.clientVar.Webstudio/port`	Specifies the port number of the machine where RMS server is hosted for WebStudio. This port is used for listening to client requests. See notes for `tibco.clientVar.RMS/hostname`. Default is `8090`.
`tibco.clientVar.Webstudio/Connection/timeout`	Timeout interval for the HTTP channel. This value is in milliseconds. Default is `60000`.
`tibco.clientVar.Webstudio/sessionTimeOut`	Time interval for which an inactive user session will be valid. This value is in minutes. Default is `30`.
`ws.validateDT.temp.dir`	Directory which is used as an interim storage of Decision Table contents during the validation process. Default is `rms/temp-dir`.
Global Variable Properties for One-Way SSL Between RMS Server and Decision Manager Clients See RMS Server Configuration Property Reference on page 546 for the procedure.	
`tibco.clientVar.RMS/security/securePort`	The port to be used for secure HTTP communication. Default is 443.

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
`tibco.clientVar.RMS/security/sslCertificateStore`	Absolute path of the JKS keystore which contains the RMS server's public/private keys. If the certificate used is CA signed, include the CA's certificate.
`tibco.clientVar.RMS/security/sslCertificateStorePassword`	The password for the keystore.
Properties for One-Way SSL Between LDAP Server, RMS Server, and Decision Manager Clients **Note** • For one-way SSL between LDAP server and RMS server, you must also change the value of `be.auth.ldap.port`. • For one-way SSL between RMS server and Decision Manager clients, you must also set the value of `be.auth.ldap.port`. See RMS Server Configuration Property Reference on page 546 for the procedure.	
`be.auth.ldap.usessl`	If set to true, enables SSL between the RMS server and the LDAP server If set to false or not set, the LDAP authentication uses a non-secure channel.
`javax.net.ssl.trustStore`	The absolute path of the keystore containing the LDAP server certificate chain.
`javax.net.ssl.trustStorePassword`	The password for the keystore.
`javax.net.ssl.trustStoreType`	The keystore type for keystores other than JKS type, for example, PKCS12.

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
Email Notification Property Group Email notification properties are used to send email notification when an artifact (business rule / decision table) undergoes a status change in approval workflow, such as, on artifact commit and on artifact reviews (Approve, Reject, BuildAndDeploy). In addition to these properties, you may have to define any SMTP protocol specific properties (`mail.smtps.auth`, `mail.smtps.ssl.enable`), depending on the mail server configuration. These properties should be defined with a prefix `ws.notify.prop`, for example, `ws.notify.prop.mail.smtps.auth`, `ws.notify.prop.mail.smtps.ssl.enable`.	
`ws.notify.enabled`	If set to true, send email notification. If set to false, do not send email notification. Default is false.
`ws.notify.mail.domain`	Specifies the domain on the email. This is used with RMS username to construct the email address to which notification is to be send. For example, if RMS username is "admin" and the property is set to "tibco.com", the email address for notification is admin@tibco.com.
`ws.notify.prop.MAIL_PROTOCOL`	Specifies the email protocol to be used to send email. SMTP and SMTPS are valid values. Default is SMTPS.
`ws.notify.prop.MAIL_SERVER_HOST`	Specifies the host name of the email server.
`ws.notify.prop.MAIL_SERVER_PORT`	Specifies the port at which mail server is listening.
`ws.notify.prop.SENDER_EMAIL`	Specifies email address to be used for sending the notification email.

Table 60 RMS Server Configuration Properties (Cont'd)

Property	Notes
`ws.notify.prop.SENDER_USERNAME`	
	Specifies the sender's username for authentication to email server.
`ws.notify.prop.SENDER_PASSWORD`	
	Specifies the sender's email password for authentication to email server.

Chapter 33 Working With Projects in WebStudio

This chapter explains how to work with projects, for example, creating projects and committing their resources for approval.

Topics

Starting and Logging in to RMS

To start RMS, do one of the following:

- Execute *BE_HOME*/`rms/bin/be-rms.exe` (or `be-rms.sh`, depending on your operating system), with valid arguments. For example, on Windows you would open a command window in *BE_HOME*/`rms/bin` and execute:

  ```
  be-rms.exe
  ```

- (Windows only) Select Start > All Programs > TIBCO > *TIBCO_HOME* > TIBCO BusinessEvents *version* > Start Rules Management Server.

The server is ready when you see a message in the command window like this:

```
Info [HTTP-Channel-Startup] - [driver.http] Channel server for HTTP
Channel [Port:5000] successfully started
```

To Log in to RMS from WebStudio

1. Type the URL http://*host*:*port*/WebStudio/ in a web browser. Default host is localhost and default port is 8090. Host and port are configurable using properties in `RMS.cdd`. Make sure that the RMS is running before logging to TIBCO BusinessEvents WebStudio.
2. Type the username and password for TIBCO BusinessEvents WebStudio.
3. Click **Login** to sign in to RMS.

The application displays the TIBCO BusinessEvents WebStudio dashboard.

The system restricts multiple simultaneous login of the same username.

To Log out of RMS

In TIBCO BusinessEvents WebStudio, click **Sign out** to log out. In case of session timeout, server logs out user automatically.

Status Check for RMS Connection

Once TIBCO BusinessEvents Studio or a TIBCO BusinessEvents Decision Manager or TIBCO BusinessEvents WebStudio client log in to RMS, a background activity on the dashboard page periodically polls the RMS server to refresh Worklist items. You can configure the interval (in seconds) of these status checks by setting the following property in `studio.tra`:

```
rms.heartbeat.delay
```

By default, the delay is of two seconds.

If the server goes down, the menu options for RMS are automatically disabled. However, if you are already performing an RMS operation and the server goes down, the operation fails. The login option is enabled when the server comes up again.

Checking Out a Project

If RMS is used in your environment, then you must log into RMS and check out a project from RMS before you can work with the project resources in the TIBCO BusinessEvents WebStudio UI.

When generating an EAR file using the Generate Deployable option, the project resources from the RMS repository are used and not from the workspace.

Depending on your role or roles, you may not have permission to check out all resources or to do all of the documented tasks.

For instructions on updating a project you already checked out, see Updating (Synchronizing) a Project on page 560.

The checkout dialog do not list the imported 4.x release projects, if the project name contains a dot(.). These imported projects should be renamed inside TIBCO BusinessEvents Studio to remove dot(.) from the project name.

To Check Out a Project in WebStudio

1. Open the checkout window using any of the following methods:
   - In the **Dashboard** tab, click the link in the **My Projects** section if no projects are checked out.
   - In the **My Workspace** tab, click **WebStudio**, with the down-pointing triangle in front of it, and select **RMS > Checkout**.
   - In the **My Workspace** tab, right-click inside the **Group Contents** section and select **RMS > Checkout**.
   - In the **My Workspace** tab, click the checkout icon in the RMS toolbar.
2. In the checkout window, select the project you want to checkout from the **Project List** drop-down.
3. Click on the checkboxes of the files you want to check out and click **OK**. Alternatively, click on the header checkbox to select all the files for checkout and click **OK**.

In TIBCO BusinessEvents WebStudio the checked out project is displayed under Group Contents and the project files are loaded to your workspace. The Group Contents section displays the artifacts based on the groups (Projects/Business Rules/Decision Tables) selected in the Groups section. The Business Rules and Decision Tables group are selected by default after you check out a project.

The TIBCO BusinessEvents WebStudio displays the project files in the Group Contents section either as a list or in a tree structure. In the Group Contents section click on the project view icon to select the **Show items as tree** or **Show items as list** options to toggle between the tree view ( )and the list view ( ). List view is the default view.

Updating (Synchronizing) a Project

This section explains how to update the projects on your workspace with the most recent changes from the corresponding RMS projects.

It is recommended that you log into the RMS server, and update (synchronize) your local copy of a project before continuing to work on it. Other users may have checked in changes, additions, and deletions that are now available for checkout. There may be other changes, too.

You can choose what updates to accept. Changes and additions to individual decision tables and business rules are listed.

To Update a Project in WebStudio

1. In the Group Contents section, select the artifact you need to synchronize.

> Synchronize command does not support multiple artifact selection using Shift or Ctrl key. Alternatively, select a folder/project to select all artifacts under that folder/project, when project files are displayed in a tree structure.

2. Open the synchronize window using any of the following methods:
   - — Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Synchronize**.
   - — Right-click on the selected artifact and select **RMS > Synchronize**.
   - — Click the synchronize icon in the RMS toolbar.
3. Click on the checkboxes of the files you want to synchronize and click **OK**. Alternatively, click the header checkbox to synchronize all listed project files and click **OK**.

   The selected artifacts are updated with the latest project files from RMS.

Committing Artifacts for Approval

After you are done with the additions, changes, or deletions to a project you checked out of RMS, you commit the artifacts for approval. You can make a commit request for a selection of artifacts, or all.

Copies of submitted artifacts are saved in cache, with the names of the users who submitted them, version numbers, and the status `Commited`. A task is created for the Approver role to review the check-in requests. (see Chapter 36, Working with the Approval Workflow in RMS, on page 597).

A Revision ID is assigned to each commit request. This ID is used in other dialogs to allow you to view details on the request.

Viewing the Audit Trail

This is available only if TIBCO BusinessEvents Decision Manager add-on is installed.

You can view the entire audit record of previous status changes for a checkin. This is especially useful for multi-stage approval process so that different reviewers actually share views on why a particular checkin was approved or rejected.

To Commit Project Resources for Approval from WebStudio

1. In the **Group Contents** section, select the artifact you need to commit.

Commit command does not support multiple artifact selection using Shift or Ctrl key. Alternatively, select a folder/project to select all artifacts under that folder/project, when project files are displayed in a tree structure.

2. Open the commit window using any of the following methods:
   - Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Commit**.
   - Right-click on the selected artifact and select **RMS > Commit**.
   - Click the commit icon in the RMS toolbar.
3. In the Commit window enter the revision description in the **Comments** text box.

   Comments field is a mandatory field and system will not commit the changes to the repository if the Comments field is empty.

4. Click on the checkboxes for the files you want to commit to RMS project and click **OK**. Alternatively, click the header checkbox to select all listed project files.

5. Click **Finish** after the system displays checkin successful message with the revision ID.

Chapter 34 Working with Decision Tables in WebStudio

This chapter explains how to work with decision tables, in TIBCO BusinessEvents WebStudio.

Decision tables are available in TIBCO BusinessEvents WebStudio in edit mode only if you have TIBCO BusinessEvents Decision Manager add-on installed on your system.

Topics

Working With Decision Tables

This section presents process to perform various decision table operations in TIBCO BusinessEvents WebStudio.

Create/Remove a Decision Table

A decision table can only be created using a virtual rule function (VRF). For instructions on adding a VRF, see Chapter 16, Rules and Rule Functions, on page 241.

You can delete a newly added table as well as any existing table. If the table you want to delete has been checked into the RMS project, then you must submit the deletion for approval. After approval, the deleted table is removed from the RMS project.

To Add a Decision Table to a Project

1. Select **Projects** group.
2. In the **Group Contents** section, right click the virtual rule function (VRF) you want to use and click **New Decision Table**.
3. Enter the decision table name in the Decision Table pop-up window and click **OK**.

   New empty decision table is displayed in the work area.

To Remove a Decision Table

1. Do one of the following:
   - In **Group Contents**, right-click the decision table name and select **Delete**.
   - In **Group Contents**, select the decision table name. Then click the delete icon ⊖ in the **Edit** toolbar.
2. If the table has been committed to the RMS project, choose one of the following options as appropriate:
   - Commit the project or the parent folder so that the RMS master copy of the table is also deleted. OR
   - Synchronize the project to replace the deleted copy with the RMS project version of the table.

Edit a Decision Table

Open a decision table

In the **Group Contents** section, double click the decision table you want to open. You can now edit, analyze, and export the open decision table.

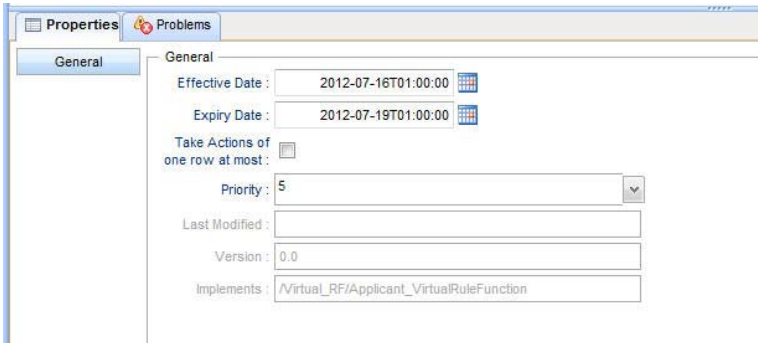
Add Rows and Columns

See Working With Rows and Columns in WebStudio on page 572 for more details on adding rows and columns to a decision table in TIBCO BusinessEvents WebStudio.

Set Decision Table Properties

Follow below steps to set properties of a new decision table:

1. In the decision table work area, click the properties icon ▤.
   The Properties view is displayed at the bottom of the work area.



General

Effective Date : 2012-07-16T01:00:00

Expiry Date : 2012-07-19T01:00:00

Take Actions of one row at most : □

Priority : 5

Last Modified :

Version : 0.0

Implements : /Virtual_RF/Applicant_VirtualRuleFunction

2. In the **Effective Date** field, you can optionally define the date and time on which the decision table becomes valid in the runtime application. Click the calendar icon and use the Calendar dialog to set the date and then edit the time as needed.
3. Similarly in the **Expiry Date** field, you can optionally define the date and time after which the decision table is no longer valid in the runtime application.
4. Check the **Take Actions of One Row at Most** check box if you want the decision table to stop after one row (rule) passes the condition tests. Only the actions of that one row are taken (until the next time the decision table’s rule function is called).

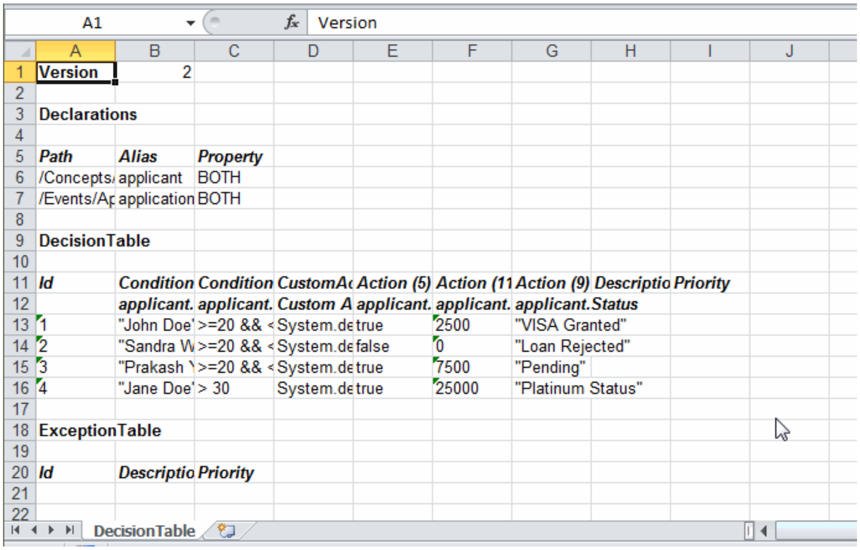
5. In the **Priority** field, set a priority as desired. When a VRF has multiple implementations (decision tables), the decision table's Priority setting determines the order in which the decision tables executes.

6. Click the Save icon in the Edit toolbar to save the changes to the decision table.

The decision table is still open for you to edit and changes are not applied to RMS unless you commit the changes to RMS.

Export a Decision Table

You can export decision tables fromTIBCO BusinessEvents WebStudio to Microsoft Excel spreadsheets. Sample exported Microsoft Excel Sheet:



A1 *fx* Version

	A	B	C	D	E	F	G	H	I	J
1	Version	2								
2										
3	Declarations									
4										
5	*Path*	*Alias*	*Property*							
6	/Concepts/	applicant	BOTH							
7	/Events/Ap	application	BOTH							
8										
9	DecisionTable									
10										
11	*Id*	*Condition*	*Condition*	*CustomA*	*Action (5)*	*Action (11*	*Action (9)*	*Descriptio*	*Priority*	
12		*applicant.*	*applicant.*	*Custom A*	*applicant.*	*applicant.*	*applicant.Status*			
13	1	"John Doe'	>=20 && <	System.de	true	2500	"VISA Granted"			
14	2	"Sandra W	>=20 && <	System.de	false	0	"Loan Rejected"			
15	3	"Prakash Y	>=20 && <	System.de	true	7500	"Pending"			
16	4	"Jane Doe'	> 30	System.de	true	25000	"Platinum Status"			
17										
18	ExceptionTable									
19										
20	*Id*	*Descriptio*	*Priority*							
21										
22										

To Export the Decision Table

1. Open the decision table you want to export.

2. Execute the export command in either of the following ways:

   — Click the export icon in the **Tools** toolbar. You see the **Save As** dialog for the exported file.

   — Right click the decision table in the Group Contents section and select **Export**.

3. Navigate to the directory in which you want to save the excel file and provide a file name.
4. Click **Save**.

Import a Decision Table

You can import correctly formatted Microsoft Excel file to create a decision table in TIBCO BusinessEvents WebStudio.

Only import from Excel files is supported You cannot import from CSV (comma-separated values) files or tab-delimited text files. You can open such files in Excel and save as Excel format binary files.

To Import the Decision Table

1. Open the **Import** dialog in either of the following ways:
   - Select the virtual rule function and click the import icon in the **Tools** toolbar.
   - Right click the virtual rule function in the Group Contents section and select **Import**.
2. Click **Choose File** in the **Import** dialog.
3. Browse the excel file you want to import and click **Open**.
4. Enter the new decision table name and click **OK**.

Understanding Columns and Rows (Rules) in a Decision Table

A decision table rule is a row in the decision table. It has one or more conditions and one or more actions. Each condition cell is equivalent to one condition (one line) in the TIBCO BusinessEvents rule editor condition area. Similarly each action cell is equivalent to one action (one line) in the TIBCO BusinessEvents rule editor action area (that is, one rule). Decision table rules are like business rules. The rule that calls the virtual rule function that implements the decision table participates in inferencing in the usual way. When the virtual rule function is called, the decision table rules are applied.

Conditions and Actions

The columns of a decision table are made up of condition columns on the left, and action columns on the right. Each column represents one condition or one action.

A condition is a test that must evaluate to true before the action is executed. If a decision table rule uses multiple conditions, all conditions for a row must evaluate to true in order for the action to execute.

In each row (rule) you define the specific conditions and actions. For example, if a condition column is Age (using a concept property of that name), then each row can define a different age range. The action for each row would define what action to take if a given concept instance contains an age property within the specified range.

If you add a second condition column called Income, then before the action is taken, a concept instance would be tested to see if both the age and the income are within the ranges specified in the rule's conditions.

Conditions that are blank, contain an asterisk, or are disabled are ignored, and are treated as if they evaluate to true.

Regular and Custom Conditions and Actions

A regular condition is the value of an entity specified in the VRF scope, or a simple comparison with the value - is greater than, is less than, is greater than or equal to, is less than or equal to.

A custom condition can use the rule language, standard functions, and data in the scope of the function at runtime (for example, scorecards and global variables). It can contain complex formulas.

A regular action sets the value of an entity specified in the VRF scope.

A custom action can use the rule language, standard functions, and data in the scope of the function at runtime to do whatever is desired. For example, the action could be to send an event out to a different system for follow-up.

Using non-literal values in a regular condition or regular action Suppose event A and event B are in scope, and event A has property ZZ, and event B has property YY. Both properties belong to the same data type. Suppose you then drag property ZZ to a condition or action column. In the cell, you can then specify a value as `b.YY`. The effect is different depending on the type of column:

- In a condition column, this means: compare the value of property ZZ with the value of property YY.
- In an action column this means: set the property ZZ to the value of YY.

Working With Rows and Columns in WebStudio

This section presents various operations you can perform as needed while working with rows and columns of a decision table in TIBCO BusinessEvents WebStudio.

Add Condition and Action Columns

The columns of a decision table are made up of condition columns on the left, and action columns on the right. Each column represents one condition or one action.

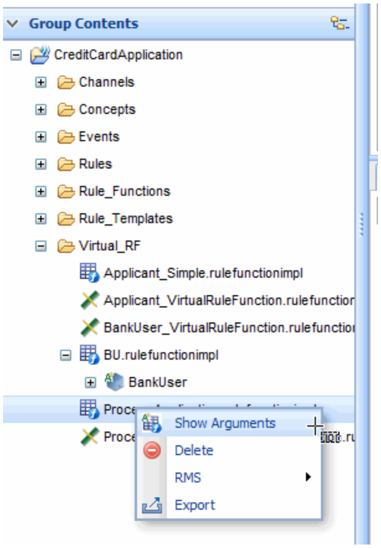
A condition is a test that must evaluate to true before the action is executed. If a decision table rule uses multiple conditions, all conditions for a row must evaluate to true in order for the action to execute.

Add condition and action columns from arguments under the VRF scope or using the rule language, standard functions, and data in the scope of the function at runtime (for example, scorecards and global variables).

To Add Condition and Action Columns from Arguments

1. Open the decision table you want to edit.
2. In the Group Contents section, right click the decision table and select **Show Arguments**.

A plus icon is displayed in front of the decision table for argument list expansion.

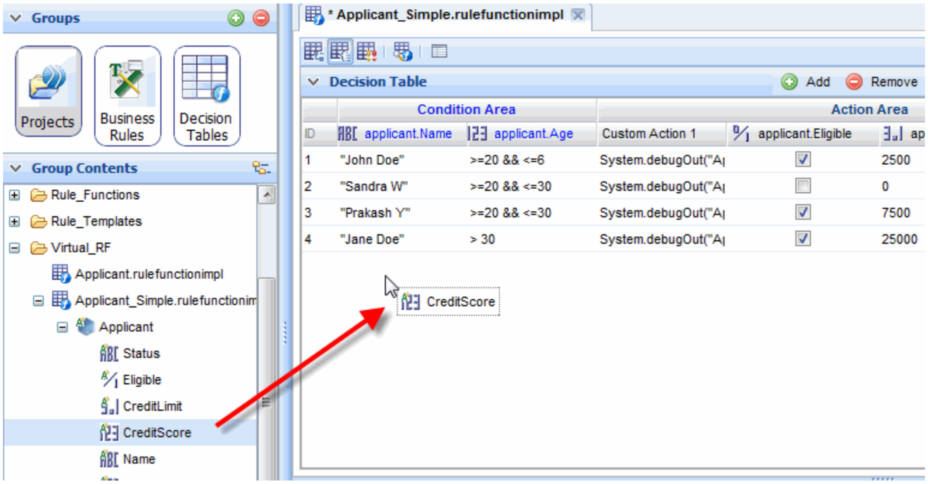


3. Click plus icon to expand the arguments list. If needed, click plus icons for concepts and rules as well.

   The decision table arguments are listed under the decision table.

4. Drag a property from the argument list to the decision table editor.

A Create Column pop-up window appears to create condition or action column.



	Condition Area		Action Area		
ID	applicant.Name	applicant.Age	Custom Action 1	applicant.Eligible	ap
1	"John Doe"	>=20 && <=6	System.debugOut("A|	☑	2500
2	"Sandra W"	>=20 && <=30	System.debugOut("A|	☐	0
3	"Prakash Y"	>=20 && <=30	System.debugOut("A|	☑	7500
4	"Jane Doe"	> 30	System.debugOut("A|	☑	25000

5. Click on the Condition or Action radio button and click **OK** to create a Condition or Action column. New Condition or Action column is added at the end of the Condition or Action column-sets respectively.

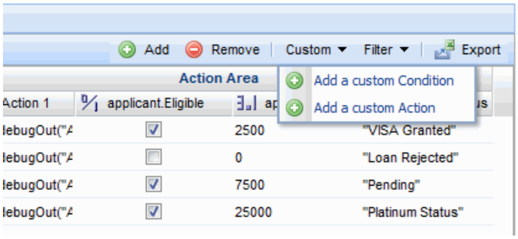
To Add Custom Condition and Action Columns

1. Open the decision table you want to edit.

2. Click **Custom** in the decision table editor menu and select either of the following options:

   — Select **Add a custom Condition** to insert a custom condition column at the end of condition column set.

   — Select **Add a custom Action** to insert a custom action column at the end of action column set.



Action Area			
Action 1	applicant.Eligible	ap	us
lebugOut("A	☑	2500	"VISA Granted"
lebugOut("A	☐	0	"Loan Rejected"
lebugOut("A	☑	7500	"Pending"
lebugOut("A	☑	25000	"Platinum Status"

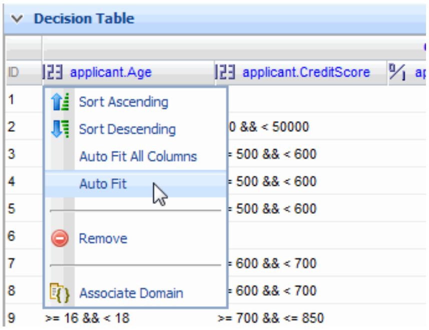
Working with Columns

You can perform actions on a column to resize, sort, associate to domain model, or remove from a decision table.

Resize a column

Perform any of the following actions to resize a column in the TIBCO BusinessEvents WebStudio:

- Drag the right boundary of a column to manually resize them.
- Open the column header menu using either of the following ways and click **Auto Fit** to resize the column to fit to the content size:
  - — Right click on the column header you want to resize.
  - — Move the mouse pointer to the column header to view the down-pointing triangle and click the triangle.



ID	applicant.Age	applicant.CreditScore	ap
1			
2		0 && < 50000	
3		= 500 && < 600	
4		= 500 && < 600	
5		= 500 && < 600	
6			
7		= 600 && < 700	
8		= 600 && < 700	
9	>= 16 && < 18	>= 700 && <= 850	

- Double click on the right boundary of a column to make it auto fit to the content size.
- Open the column header menu using either of the above mentioned ways and click **Auto Fit All Columns** to resize all columns to fit their content size.

Sort the decision table

Sort the decision tables in TIBCO BusinessEvents WebStudio in one of the following ways:

- Click once on the column header to sort the table in ascending order of the column and click again on the column header to re-sort the table in descending order of the column.

- Open the column header menu using either of the following ways and click **Sort Ascending** or **Sort Descending** to sort the table in ascending or descending order of the column:
  - — Right click on the column header you want use for sorting.
  - — Move the mouse pointer to the column header to view the down-pointing triangle and click the triangle.

Sort options are not available for columns with boolean entry values and for custom condition and action columns.

Associate and Remove Domain Association

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.

To associate a domain model, right click on the column and from the menu click **Associate Domain**. The setting toggles to ON. To clear the domain association click **Associate Domain** again. The setting toggles to OFF.

Remove a Column

Right click the column header and click **Remove** to remove the column from the decision table.

You cannot remove the last condition column because a minimum of one condition column must exist if you have any action columns.

Add a Row

Each row in a decision table represents a separate business rule. You can control the order in which sets of rows are executed using the row priority setting, see Setting Rule (Row) Priorities on page 577.

Set Row Priority in TIBCO BusinessEvents WebStudio

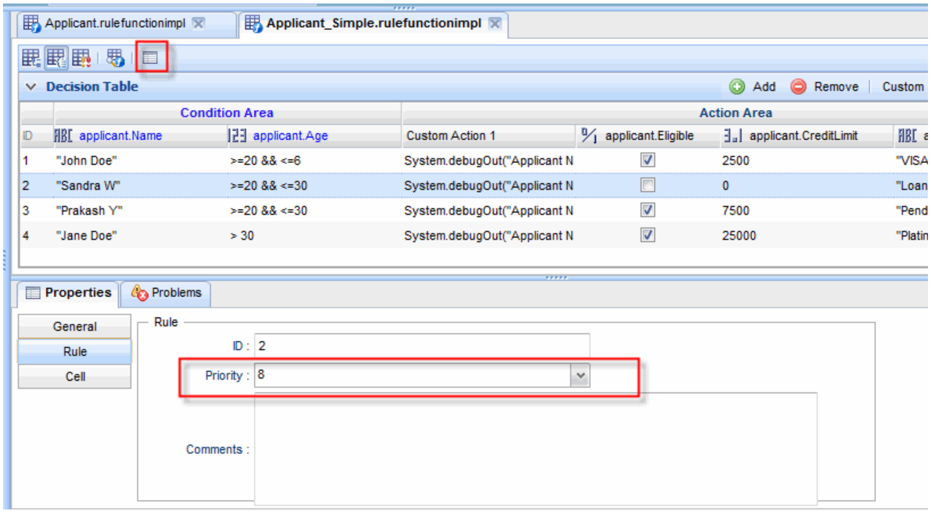
To set the row priority, with ten as the lowest priority and one as the highest, do the following:

1. Select the row whose priority you want to set.
2. Click the Properties icon to display the Properties view.

3. Select the Properties view, and select the **Rule** side-tab.

4. In the **Priority** field, select the appropriate value.

5. Click the Save icon in the Edit toolbar to save the changes to the decision table.



	Condition Area		Action Area			
ID	applicant.Name	applicant.Age	Custom Action 1	applicant.Eligible	applicant.CreditLimit	a
1	"John Doe"	>=20 && <=6	System.debugOut("Applicant N	☑	2500	"VISA
2	"Sandra W"	>=20 && <=30	System.debugOut("Applicant N	□	0	"Loan
3	"Prakash Y"	>=20 && <=30	System.debugOut("Applicant N	☑	7500	"Pend
4	"Jane Doe"	> 30	System.debugOut("Applicant N	☑	25000	"Platin

Rule

ID : 2

Priority : 8

Comments :

Setting Rule (Row) Priorities

Each row is like a separate business rule. You can control the order in which sets of rows are executed using the row priority setting. Rows with higher priorities are executed before those with lower priorities, as follows:

1. First, all conditions are checked for all rows that have the highest priority. (The checking order within a set of rows with the same priority is not determinate.)

2. Then the rule actions for all of those rows whose conditions evaluate to true are executed. (The execution order is not determinate. The runtime engine optimizes rule execution.)

3. The process is repeated for all rows with the next highest priority, and so on.

To Set Row Priority

Ten is the lowest priority and one is the highest. Five is the default priority.

1. Select the row whose priority you want to set.

2. Select the Properties view, and select the Rule side-tab.

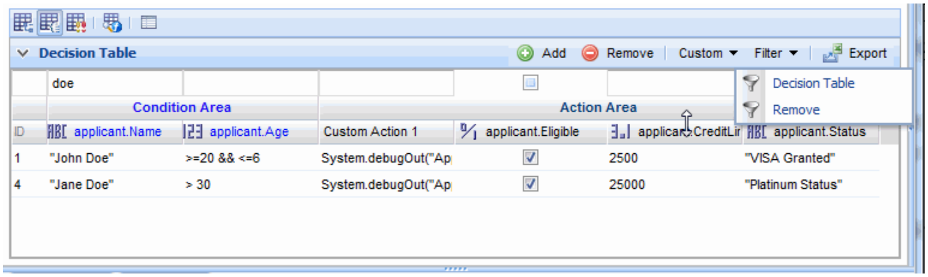
3. In the Priority field, select the appropriate value and click Apply.

Filter the Rows

Filtering rows helps you to focus on certain rows instead of entire table. Apply filters on multiple columns for more refined results. In TIBCO BusinessEvents WebStudio decision table filter makes only textual comparison between the row values and filter values.

Click **Filter > Decision Table** in the decision table editor menu to enable the filter for the decision table. Enter the values in filter cells (above column header) to filter the table.

To remove the filter click **Filter > Remove** in the decision table editor.



	doe			□		
	Condition Area		Action Area			
ID	applicant.Name	applicant.Age	Custom Action 1	applicant.Eligible	applicant.CreditLir	applicant.Status
1	"John Doe"	>=20 && <=6	System.debugOut("Ap	☑	2500	"VISA Granted"
4	"Jane Doe"	> 30	System.debugOut("Ap	☑	25000	"Platinum Status"

Remove a Row

Select the row you need to remove and click **Remove** in the decision table editor. Click **OK** in the Remove Decision Table Record window to confirm removal of the row. You can remove only one row at time.

Supported Operators

Table 61 lists the operators supported by Decision Tables:

Table 61 Supported Operators

Operator	Description
==	Equals to
!=	Does not equal to
>	Greater than
>=	Greater than or equals to
<	Less than
<=	Less than or equals to
||	Logical OR
&&	Logical AND
+	Addition or String concatenation (depending on context)
!	Logical NOT (only with boolean)
-	Subtraction
*	"Don't Care" (that is, ignore), or multiplication (depending on context)
/	Division
=	Assignment
.	Scope resolution
()	Operator precedence order or function call (depending on context)
[]	Array declaration or array indexing (depending on context)
{}	Block resolution or array Initialization (depending on context)
?:	Ternary conditional
++	Increment (used only with custom actions)

Table 61 Supported Operators (Cont'd)

Operator	Description
`--`	Decrement (used only with custom actions)

Using Operators in Tables

Named or Scoped Conditions must Evaluate to Boolean

For logical operators, named conditions will have to use them explicitly from the second operand onwards.

For example, if the LHS of the condition is the column name, `bankuser.Age`, the following are valid entries:

```
50 || ==60 , 40 && ==80, 20 || >=80
```

Custom Conditions must evaluate to Boolean

Same as above, except that you must specify the entire Boolean expression in the cell, including LHS.

Actions or Custom Actions

Named actions perform assignment of the cell expression to the column name property.

Custom actions are treated as normal strings and can be anything that you would enter in the THEN section of rules (LHS included).

Analyze a Decision Table

Table Analyzer will create a sparse matrix data structure representing an optimized form of the decision table in memory. In the Table Analyzer you can set example condition values and perform various validation checks on the currently displayed decision table.

- Table analyzer supports the Not Equal To (`!=`) operator.

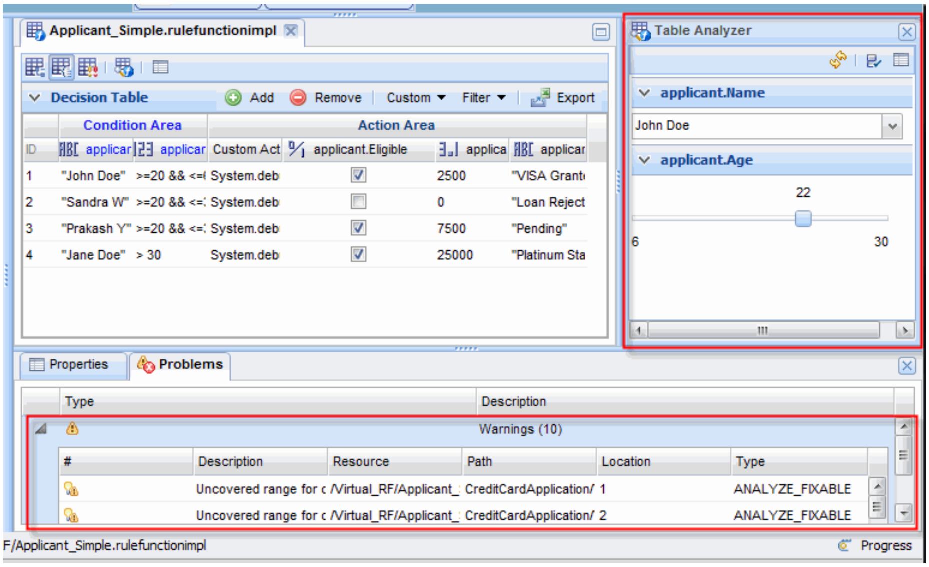
The table analyzer feature provides various tools that you can use to check the validity of your decision tables.

To analyze a decision in the TIBCO BusinessEvents WebStudio, do the following:

1. Open the decision table you want to analyze.
2. Click the Decision Table Analyzer icon to view Table Analyzer.
3. In the Table Analyzer section, controls are created dynamically for setting values and ranges for each condition in the table. Enter or select the example values for which you want to analyze the table.

   For example, if the condition is a range, for instance “< 40” or “> 10 && < 100”, then the corresponding control in the Table Analyzer section is a Slider that spans the range from the minimum value to the maximum value. You can then select a range of values within the range.
4. Click the Analyze icon to analyze the decision table based on the selected values.

A pop-up window displays the count of any issues. The Problems section at the bottom displays more information on issues. Click on the issue to highlight the row with the issue.



	Condition Area		Action Area			
ID	applicar	applicar	Custom Act	applicant.Eligible	applica	applicar
1	"John Doe"	>=20 && <=(	System.deb	☑	2500	"VISA Grant(
2	"Sandra W"	>=20 && <=:	System.deb	□	0	"Loan Reject
3	"Prakash Y"	>=20 && <=:	System.deb	☑	7500	"Pending"
4	"Jane Doe"	> 30	System.deb	☑	25000	"Platinum Sta

John Doe

Type	Description
⚠	Warnings (10)

#	Description	Resource	Path	Location	Type
⚠	Uncovered range for c	/Virtual_RF/Applicant_	CreditCardApplication/	1	ANALYZE_FIXABLE
⚠	Uncovered range for c	/Virtual_RF/Applicant_	CreditCardApplication/	2	ANALYZE_FIXABLE

You can click the Show Coverage icon ▤ in the table analyzer section to highlight all rows that meet the criteria.

For example, if you select `Male` for `Gender`; `true` for `Eligible`; and `50000-60000` for `Income`, clicking Show Coverage displays all rows that have `Male`, `true`, and an income between 50000 and 60000.

Validate a Decision Table

In TIBCO BusinessEvents WebStudio, you can validate decision tables for the changes using the validate command. Select the decision table and click on the validate icon ✓ to validate the table.

The validate icon is activated when the decision table is opened in the editor. EAR file for the project should be present in the deploy location; otherwise, validate command gives error.

If there are any access control violations or syntax errors in the tables, they are shown in the Problems View tab at the bottom of the application. New errors in syntax are added to these existing errors. Double-click errors to see the problematic view. Take any needed corrective actions and then validate the table again until all errors are resolved.

Chapter 35 Business Rules

Business Rules artifact allows non-technical business users to build complex business rules in TIBCO BusinessEvents WebStudio. Business Rules are created using the Rule templates developed inTIBCO BusinessEvents Studio. See Chapter 17, Rule Templates, on page 269 for details on how to create rule template. This chapter explains about business rules and how to create them.

Topics

Business Rules Overview

Business rules are defined inside of TIBCO BusinessEvents WebStudio, and are intended to be used by non-technical users. TIBCO BusinessEvents WebStudio users complete the definition of the rule instance using one of the following types of user interface:

- A form where users can provide values for bindings defined in TIBCO BusinessEvents Studio. The form interface is the visual presentation of the rule template defined by rule template views. See Rule Template Views on page 273 for more details on rule template views.
- A builder style of user interface where users can build conditions and actions using various operators. A builder based user interface is presented to user only if there is no view associated with the rule template.

Business Rule Builder

The builder for business rule in the TIBCO BusinessEvents WebStudio consists of two sections:

When section - condition builder

When section lets user define additional conditions that must be met, in addition to the pre-condition defined in the Rule Template. The builder determines the scope of the rule template, and allows the user to define individual conditional clauses based on that scope. Each conditional clause has one or more sub-clause, and a match operator determines whether conditional clause is true when all sub-clause are true, or any one sub-clause is true, or none of the sub-clauses evaluate to true.

Then section - command builder

Then section defines additional actions to be taken based on the conditions defined. The builder lets user define actions based on the statements defined in the Action Context section of the rule template.

Ensure that al the actions in the business rule are defined and the order should match that of the rule template. Otherwise, the business rule will be invalid.

Working with Business Rules

In TIBCO BusinessEvents WebStudio, you can create, modify, or delete business rules. Any modification after commit goes through the RMS approval process (see Chapter 36, Working with the Approval Workflow in RMS, on page 597).

A business rule in a project is identified by the `.ruletemplateinstance` file extension. The rule template instance file is generated when BuildAndDeploy option is selected for the business rule in approvals. See Generating One Business Rule's Rule Template Instance File on page 606 for more details.

Adding Business Rules

In TIBCO BusinessEvents WebStudio, business rule is created using a rule template defined in TIBCO BusinessEvents Studio. If a rule template view is associated for the rule template, the business rule has a easy-to-use form based interface to specify conditions and action; otherwise, condition and actions are added using builder. See Rule Template and Rule Template View Overview on page 270 for overview on rule template, and rule template view.

See Chapter 33, Working With Projects in WebStudio, on page 555 for more details on starting and logging to RMS, and checking out project resources.

To add a business rule

1. In TIBCO BusinessEvent WebStudio, under the Group Contents section, right click on the rule template you want to use and select **Create Business Rule**.
2. Type the business rule name in dialogue box and click **OK**.
3. If the rule template view is defined for the rule template then HTML form is presented in the business rule editor. Then follow steps in To add a business rule using HTML form on page 587 to create a business rule.
4. Otherwise, if view is not associated with the rule template, a builder interface open up in the editor. Follow the steps in the following sections to create a business rule:

   — To add a conditional clause using builder on page 588

   — To add a action using builder on page 589

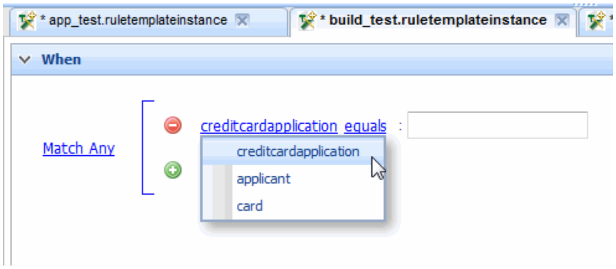
To add a business rule using HTML form

1. In WebStudio, enter the threshold values in the input fields of the HTML form.

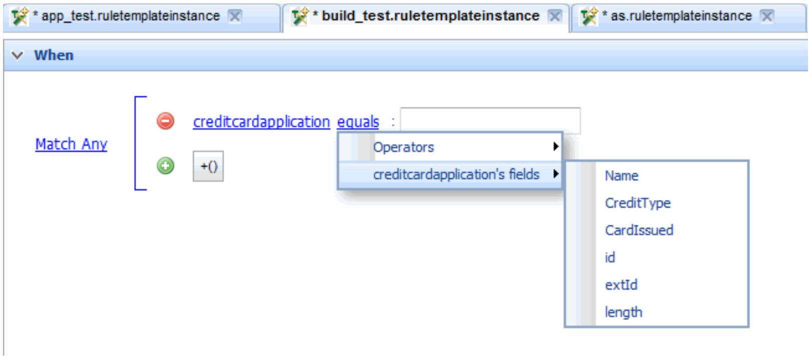
2. Click the save icon to save updates to the business rule.

To add a conditional clause using builder

1. In the business rule editor, under the When section, click on artifact-name link and select the artifact you require.



2. Click on the operator-name link to select another operator, or a child artifact of the previously selected artifact, as per your requirement. See Condition Builder Operators on page 592 for list of supported operators.


* app_test.ruletemplateinstance
* build_test.ruletemplateinstance
* as.ruletemplateinstance
When
Match Any
creditcardapplication
equals
:
+()
Operators
creditcardapplication's fields
Name
CreditType
CardIssued
id
extId
length

3. Enter the conditional value for the artifact in the empty box.
4. Click the add icon to another condition to the conditional clause, or click the remove icon to remove the existing condition from the conditional clause, if you want.
5. Click the add sub-clause icon +() to add conditional sub-clause to the conditional clause, if you want.

6. Click match-operator link to select the matching condition for which the conditional clause evaluates to true. The values are:

   — **Match Any** Conditional clause is true if any of the condition evaluates to true

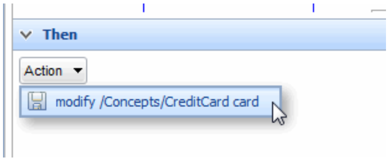
   — **Match All** Conditional clause is true only if all of the conditions evaluates to true

   — **Match None** Conditional clause is true only if none of the conditions evaluates to true.

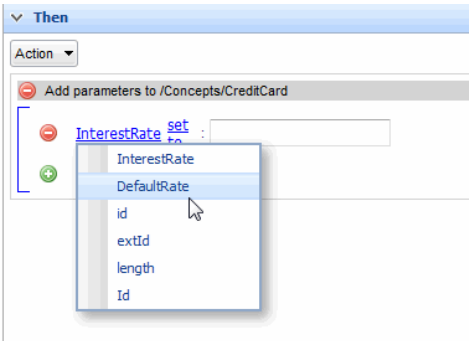
7. Click the save icon to save updates to the business rule.

To add a action using builder

1. In the business rule editor, under the Then section, click **Action** dropdown to select the actions available for the rule template.



2. Click the artifact-name link and select the artifact which you want modify.



3. Click the operator-name link and select the command for the artifact. See Action Builder Operators on page 594 for list of available on operators.

4. Type the value, you want to set for the artifact, in the empty box.

5. Click the add icon to add another parameter for the action, or click the remove icon to remove the parameter for action, if you want.

6. Click the save icon to save updates to the business rule.

Export a Business Rule

You can export business rules fromTIBCO BusinessEvents WebStudio to rule template instance file. Sample exported rule template instance file content:

```
<?xml version="1.0" encoding="UTF-8"?>
<RuleTemplateInstance id="738cd4ea-8569-4744-aa86-fb88a89bf0a3"
implementsPath="/Rule_Templates/Applicant_PreScreen">
    <actions/>
    <binding id="minimumIncome" value="5000"/>
    <binding id="minimumAge" value="25"/>
    <binding id="creditType"/>
</RuleTemplateInstance>
```

To Export the Business Rule

1. Open the **Export** dialog in either of the following ways:
   - Select the business rule and click the export icon in the **Tools** toolbar.
   - Right click the business rule in the Group Contents section and select **Export**.
2. Navigate to the directory in which you want to save the rule template instance file and provide a file name.
3. Click **Save**.

Deleting Business Rules

If the business rule you want to delete has been checked into the RMS project, then you must submit the deletion for approval. After approval, the deleted table is removed from the RMS project, but it is not removed from user's workspace.

If you are not sure whether a table you deleted was committed to the RMS project in a prior action, select Project > Commit. If the deleted table does not appear in the Changes Made panel, then it is a local table.

To Delete a Business Rule

1. Do one of the following:
   - In **Group Contents**, right-click the business rule name and select **Delete**.
   - In **Group Contents**, select the business rule name. Then click the delete icon in the **Edit** toolbar.

2. If the business rule has been committed to the RMS project, choose one of the following options as appropriate:
   - Commit the project or the parent folder so that the RMS master copy of the business rule is also deleted (see To Commit Project Resources for Approval from WebStudio on page 561). OR
   - Synchronize the project to replace the deleted copy with the RMS project version of the business rule (see To Update a Project in WebStudio on page 560).
3. Do one of the following:
   - In Studio Explorer, right-click the business rule name and select **Delete**.
   - In Studio Explorer, select the business rule name. Then select **File > Delete**.
4. If the business rule has been committed to the RMS project, choose one of the following options as appropriate:
   - Select **Project > Commit** so that the RMS master copy of the table is also deleted. OR
   - Select **Project > Synchronize** to replace the local deleted copy with the RMS project version of the table.

⚠ When an artifact deletion is approved, the artifact is permanently deleted from the RMS project. However, When Decision Manager or WebStudio users update their projects, the deleted artifacts do not appear in the list of changes.

Users who have already checked out the project must manually delete their local copy of the artifact.

Business Rule Operators

This section lists the operators supported in the business rules builder. RTI builder operators consists of a conditional clause operators and action builder operators. The operator list changes as per the parent entity type in the builder.

Condition Builder Operators

Parent Type	Operator Value on WebStudio Client	Operator Value on RMS Server
Concept / Event		
	Matches Other Field	== *<field>*
	Differs From Field	!= *<field>*
	Is Null	== null
	Is not Null	!= null
Data / Time / Datetime		
	Matches Other Field	== *<field>*
	Differs From Field	!= *<field>*
	Is Null	== null
	Is not Null	!= null
Integer / Float		
	Matches Other Field	== *<field>*
	Differs From Field	!= *<field>*
	Is Null	== null
	Is not Null	!= null
	Greater Than	>
	Greater Than Field	> *<field>*

Parent Type	Operator Value on WebStudio Client	Operator Value on RMS Server
	Greater Than Equal To	>=
	Greater Than Equal To Field	>= <*field*>
	Less Than	<
	Less Than Field	< <*field*>
	Less Than Equal To	<=
	Less Than Equal To Field	<= <*field*>
	Equals	==
	Not Equals	!=
Text / String		
	Matches Other Field	== <*field*>
	Differs From Field	!= <*field*>
	Is Null	== null
	Is not Null	!= null
	Equals	==
	Not Equals	!=
Boolean		
	Equals	==
	Not Equals	!=
	Matches Other Field	== <*field*>
	Differs From Field	!= <*field*>

Action Builder Operators

Parent Type	Operator Value on WebStudio Client	Operator Value on RMS Server
Concept / Event		
	Set To	=
	Set To Field	= *<field>*
	Set To Null	= null
Data / Time / Datetime		
	Set To	=
	Set To Field	= *<field>*
	Set To Null	= null
Integer / Float		
	Set To	=
	Set To Field	= *<field>*
	Increment By	+=
	Increment By Field	+= *<field>*
	Decrement By	-=
	Decrement By Field	-= *<field>*
Text / String		
	Set To	=
	Set To Field	= *<field>*
	Set To Null	= null
Boolean		
	Set To True	= true
	Set To False	= false

Parent Type	Operator Value on WebStudio Client	Operator Value on RMS Server
	Set To Field	= <*field*>

Chapter 36 Working with the Approval Workflow in RMS

This chapter explains the approval workflow of decision tables or business rules.

Topics

Working with the Approval Workflow—Overview

When business users have finished working with artifacts (decision tables or business rules), they submit them for approval. All business user actions, such as committing or deleting artifacts are committed to the RMS project only when a user whose role gives permission to do so approves the requests.

The approval permissions are set at project level using the ACL file for the project. Allow the `approval` action type for the user role in the ACL file to grant approval permission. See *TIBCO BusinessEvents Administration* for details on setting permission for a user role at project level.

This section explains the provided workflow for approving and rejecting business users' commit requests.

This manual documents the behavior of the product as shipped. The workflow, the roles used, and the permissions granted to the roles are all configurable. However, the general flow is likely to be similar to that described in this guide.

With this release, the approval can only be done in TIBCO BusinessEvents WebStudio.

Use of RMS—and therefore the approval process—is required when TIBCO BusinessEvents WebStudio is used.

See Committing Artifacts for Approval on page 561 for business user procedures.

See Checking a Worklist and Taking Action on page 600 for approver procedures.

Who Can Approve or Reject Commit Requests

In TIBCO BusinessEvents WebStudio users with Administrator role permissions can approve or reject commit requests, or delegate approval. They can also delegate the task of approving or rejecting commit requests to any specified user role and not to a specific user. In this case, the approval or rejection requests will be sent to all the users in that role. See To Delegate a Workitem on page 600.

Approval Status Values

COMMITED When a user commits one or more resources for approval, the status of the request is set to Commited. The approver then sets the status as appropriate.

APPROVE The request was approved. If the submission request was to add or change an artifact, that artifact is copied to the RMS project. If the submission request was to delete an approved artifact, that artifact is deleted from the RMS project.

REJECT The change was rejected. If the submission request was to add or change an artifact, that artifact is not copied to the RMS project. If the submission request was to delete an approved artifact, that artifact is not deleted. It is up to the business user to make changes in their local project accordingly.

BUILDANDDEPLOY The artifact is build and ready to be deployed. The artifact after approval is built and kept ready to be deployed at the location specified by a property in the `RMS.cdd` file. See RMS Server Configuration Property Reference on page 546 for more details on property.

Checking a Worklist and Taking Action

Approval requests are handled in the Worklist window in TIBCO BusinessEvents WebStudio.

To Check Your Worklist and Take Action

1. Open the worklist window using any of the following methods:

   — Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Show Worklist**.

   — Right-click on any artifact and select **RMS > Show Worklist**.

   — Click the show worklist icon in the RMS toolbar.

   The worklist window displays all the requests submitted for approval.

2. To view the artifacts submitted for approval, click on the triangle in front of the Revision ID. To examine the artifact before taking an action, double-click the artifact. The decision table or business rule displays in the editor.

3. To take action on a artifact, click on the `Commited` status under the Status column, and select a value from status drop-down menu:

   — Approve

   — Reject

   A message field displays error messages in the event of a problem.

4. Click anywhere outside the drop-down menu and click **Ok**.

To Delegate a Workitem

A user in the Administrator role can delegate a workitem to one or more roles. Users belonging to any of those roles see the workitem in their worklist.

Users in the role specified by the RMS global variable property `tibco.clientVar.RMS/Approval/adminRole` have Administrator role privileges. See Configuring RMS Server Properties, page 543.

1. Open the worklist window using any of the following methods:

   — Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Show Worklist**.

   — Right-click on any artifact and select **RMS > Show Worklist**.

   — Click the show worklist icon in the RMS toolbar.

   The worklist window displays all the requests submitted for approval.

2. Check the checkbox for one or more Revision IDs, you need to delegate.

3. Select one or more user roles from the **DelegateTo** dropdown and click **Apply**. You see all roles that you do not belong to, and that have permission to approve or reject the artifact.

   The workitem disappears from your worklist. It can be seen in the worklist of the users in roles you selected in step 3.

Chapter 37 Deployable Files Generation

This chapter explains configuration and procedures for generating EAR files, decision table classes, and business rule's rule template instance file.

Topics

Generating Deployable Files (EAR and Class Files)

Deployable files are generated using the TIBCO BusinessEvents Studio project files, located in the RMS project's folder.

The deployable files are of two types: EAR files and class files.

EAR files are generated in the Decision Manager and TIBCO BusinessEvents WebStudio UI.

You can generate class files using the Decision Manager and TIBCO BusinessEvents WebStudio UI, or at the command line. You can generate an entire project's class files, or one decision table's class file, or one business rule's rule template instance file. If you generate an entire project's class files, a property in the CDD file enables you to exclude unwanted packages.

You can generate one decision table's class file and business rule's rule template instance file using TIBCO BusinessEvents WebStudio only.

Location of Generated Files

Location of generated files can be configured using deploy location properties in the `RMS.cdd` file. See RMS Server Configuration Property Reference on page 546 for more details on deploy location properties. Below are the location of generated files as per the preconfigured location in the shipped product.

- EAR files are saved in the RMS project directory in the RMS's shared directory. For example, EAR file for CreditCardApplication project (`CreditCardApplication.ear`) is stored at

  BE_HOME`\rms\shared\CreditCardApplication`

- Project's class files are saved in the `codegen` subdirectory under the RMS project directory in the RMS's shared directory. For example, class files for CreditCardApplication project (`CreditCardApplication.ear`) are stored at

  BE_HOME`\rms\shared\CreditCardApplicatio\codegen`

- Individual decision table's class file and business rule's rule template instance are saved in the RMS project directory in the RMS's shared directory.

Generating the Project EAR or All Project Class Files

You can generate deployable EAR files only if you have permission to check out all the required project resources (ACLs may limit what you can check out).

Generate Deployable command in TIBCO BusinessEvents WebStudio does not work on AIX and HPUX as studio-tools are not supported for these platforms.

To Generate the Project EAR or All Project Class Files in WebStudio

1. Open the Generate Deployables window using any of the following methods:

   — Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Generate Deployables**.

   — Right-click on any artifact and select **RMS > Generate Deployables**.

   — Click the Generate Deployables icon in the RMS toolbar.

   The Generate Deployables window is displayed.

2. Do one of the following:

   — To generate class files, check the **Generate Classes Only** checkbox.

   — To generate EAR files, do not check the **Generate Classes Only** checkbox.

3. From the drop-down list, select the project for which you want to generate the deployable files.

4. (For EAR files only.) If you want to generate the debug information, check the **Generate Debug Info** checkbox.

5. (For EAR files only.) If you want to include all service-level global variables, check the checkbox.

6. Click **OK**.

Files are saved to the preconfigured location (see Location of Generated Files on page 604). Any existing files are overwritten.

Generating One Decision Table’s Class File

In TIBCO BusinessEvents WebStudio, the BuildAndDeploy status menu option in the Worklist window generates the class file for just selected decision tables. You can generate a decision table’s class file only after it is approved. See Checking a Worklist and Taking Action on page 600 on how to approve a commit request.

You can generate one decision table’s class file using TIBCO BusinessEvents WebStudio only.

To Generate One Decision Table Class File

1. Open the worklist window using any of the following methods:

   — Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Show Worklist**.

   — Right-click on any artifact and select **RMS > Show Worklist**.

   — Click the show worklist icon in the RMS toolbar.

   The worklist window displays all the requests submitted for approval as well approved requests.

2. To view the approved decision tables, click on the triangle in front of the Revision ID. To examine the decision table before taking an action, double-click the table. The decision table displays in the editor.

3. To generate classes for the decision table, click on the `Approved` status under the Status column, and select **BuildAndDeploy** from the status drop-down menu. The project EAR must already be created.

   A message field displays error messages in the event of a problem.

4. To start generating classes click anywhere outside the drop-down menu and click **Ok**.

Generating One Business Rule's Rule Template Instance File

In TIBCO BusinessEvents WebStudio, the BuildAndDeploy status menu option in the Worklist window generates the rule template instance file for just selected business rules. You can generate a business rule's rule template instance file only after it is approved. See Checking a Worklist and Taking Action on page 600 on how to approve a commited request.

To Generate One Rule Template Instance File

1. Open the worklist window using any of the following methods:

   — Click **WebStudio**, with the down pointing triangle in front of it, and select **RMS > Show Worklist**.

   — Right-click on any artifact and select **RMS > Show Worklist**.

   — Click the show worklist icon in the RMS toolbar.

   The worklist window displays all the requests submitted for approval as well approved requests.

2. To view the approved business rules, click on the triangle in front of the Revision ID. To examine the business rule before taking an action, double-click the rule. The business rule displays in the editor.
3. To generate rule template instance file for the business rule, click on the `Approved` status under the Status column, and select **BuildAndDeploy** from the status drop-down menu.

   A message field displays error messages in the event of a problem.
4. To start generating rule template instance file click anywhere outside the drop-down menu and click **Ok**.

Chapter 38 ActiveMatrix BusinessWorks Integration

This chapter explains how to use the TIBCO ActiveMatrix BusinessWorks plug-in, to integrate ActiveMatrix BusinessWorks and TIBCO BusinessEvents applications.

Topics

Overview of Integration with ActiveMatrix BusinessWorks

If TIBCO ActiveMatrix BusinessWorks used in your environment, you can take advantage of integration components provided by TIBCO BusinessEvents. Integration between TIBCO BusinessEvents and ActiveMatrix BusinessWorks enables each product to take advantage of the strengths of the other. For example, ActiveMatrix BusinessWorks can use TIBCO BusinessEvents as a light-weight rules engine, and TIBCO BusinessEvents can use transports available in ActiveMatrix BusinessWorks.

This guide assumes you know how to work with ActiveMatrix BusinessWorks processes. See *TIBCO BusinessWorks Process Design Guide* if you need detailed guidance.

Integration components include an ActiveMatrix BusinessWorks plug-in, which provides a palette of TIBCO BusinessEvents activities, and various TIBCO BusinessEvents functions. See Integration Components on page 612 for details. You can use these components to do the following:

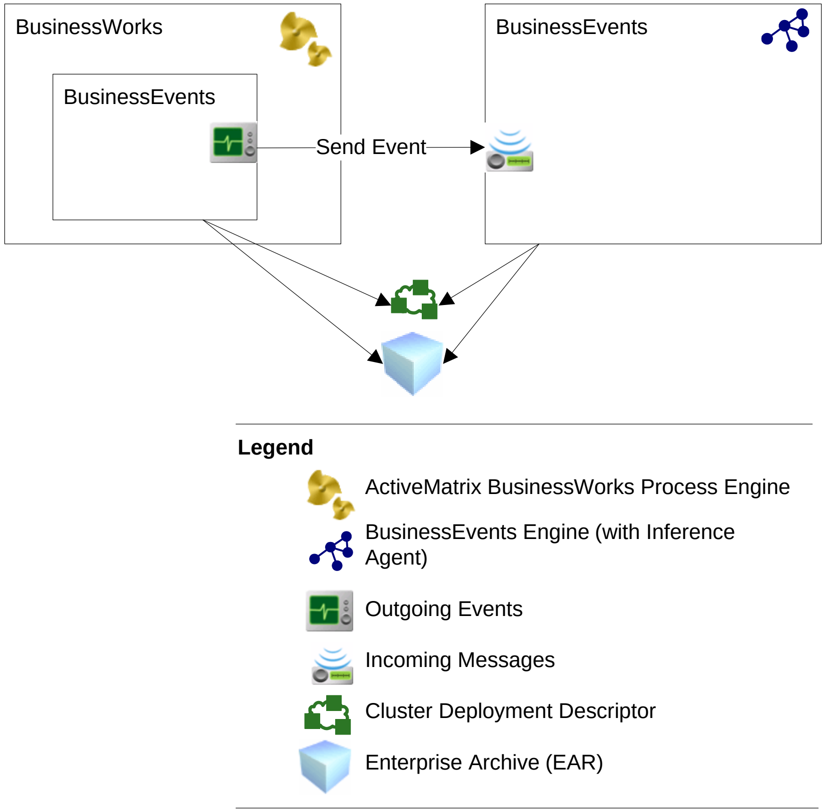
- More easily send and receive TIBCO BusinessEvents events using ActiveMatrix BusinessWorks activities.
- Invoke TIBCO BusinessEvents rule functions from ActiveMatrix BusinessWorks
- Call ActiveMatrix BusinessWorks processes from TIBCO BusinessEvents.

This chapter explains how to configure the environment and use these features.

See the product readme for the supported version of TIBCO ActiveMatrix BusinessWorks.

TIBCO BusinessEvents requires JRE 6. You must also use JRE 6 with ActiveMatrix BusinessWorks in order to integrate with TIBCO BusinessEvents.

The Container and the Contained Engine


BusinessWorks
BusinessEvents
Send Event
BusinessEvents
Legend
ActiveMatrix BusinessWorks Process Engine
BusinessEvents Engine (with Inference Agent)
Outgoing Events
Incoming Messages
Cluster Deployment Descriptor
Enterprise Archive (EAR)

You can design integration projects in different ways. You can start an ActiveMatrix BusinessWorks engine inside a TIBCO BusinessEvents container, or you can start a TIBCO BusinessEvents engine inside a ActiveMatrix BusinessWorks container — or you can do both.

In all cases, TIBCO BusinessEvents and ActiveMatrix BusinessWorks must be deployed on the same machine, and both must reference the same CDD file and EAR file (as shown in the figure above).

Which engine runs as the container affects project design and runtime behavior, as explained in Design Considerations on page 614.

Integration Components

Integration components include a palette of activities in ActiveMatrix BusinessWorks, and a category of functions in TIBCO BusinessEvents.

ActiveMatrix BusinessWorks Activities Palette

RuleServiceProvider Configuration Resource

The RuleServiceProvider Configuration resource is used to identify the source of the ontology definitions (in an EAR file) and the location of the TIBCO BusinessEvents application at runtime.

Event Related Activities

The Receive Event, Send Event, And Wait for Event activities enable you to work more easily with TIBCO BusinessEvents events within ActiveMatrix BusinessWorks. The ontology definitions from the TIBCO BusinessEvents application are available to these activities so you can easily configure them. See the following sections:

- Receive Event Resource Reference on page 628
- Send Event Resource Reference on page 630
- Wait for Event Resource Reference on page 631

Invoke RuleFunction Activity

An Invoke RuleFunction activity invokes a TIBCO BusinessEvents rule function in a specified agent instance and passes information to it. The Invoke RuleFunction activity can be combined with any process starter. Execution is synchronous.

TIBCO BusinessEvents Functions

TIBCO BusinessEvents provides a BusinessWorks category of functions, as follows:

`BusinessWorks.invokeProcess()`: Invokes an ActiveMatrix BusinessWorks process in synchronous mode and waits for completion of the process before returning to the rule or rule function. Starts the process engine if it is not already started. Returns an event, or null. Generates an advisory event if it times out.

`BusinessWorks.startProcess()`: Invokes an ActiveMatrix BusinessWorks process in asynchronous mode and returns the process ID (job ID). Upon completion, the ActiveMatrix BusinessWorks process passes an event to the rule function specified in an argument of `startProcess()`. Starts the process engine if it is not already started.

`BusinessWorks.cancelProcess()`: Cancels the specified ActiveMatrix BusinessWorks process. Useful for canceling a long running job. (Cancellation is not guaranteed because the process may complete before receiving the cancellation command.)

`BusinessWorks.init()`: Used as needed to initialize the ActiveMatrix BusinessWorks engine.

`BusinessWorks.shutdown()`: Used as needed to stop the ActiveMatrix BusinessWorks engine.

Design Considerations

This section presents some considerations to keep in mind when you are designing your integration project.

Integration Scope

You can only use TIBCO BusinessEvents resources that are included in the TIBCO BusinessEvents EAR file referenced in the RuleServiceProvider Configuration resource.

TIBCO BusinessEvents and ActiveMatrix BusinessWorks must be installed on the same machine.

Thread Management

TIBCO BusinessEvents generates its own threads to execute the rule function that the ActiveMatrix BusinessWorks process calls through the Invoke RuleFunction activity. Then the ActiveMatrix BusinessWorks thread is released and the process is set to a pending state. When the rule function returns, the ActiveMatrix BusinessWorks process resumes its Ready state.

Avoiding Threading Issues

It is possible to configure a complex execution path in your project, for example, one in which TIBCO BusinessEvents invokes an ActiveMatrix BusinessWorks process, which in turn invokes a TIBCO BusinessEvents rule function, and so on.

The `BusinessWorks.InvokeProcess()` rule function and the Invoke RuleFunction activity both operate synchronously. Both attempt to acquire a lock on the TIBCO BusinessEvents RTC.

Check your execution path carefully to ensure that there are no threading issues leading to deadlock. In general, make sure that no action in the entire execution path attempts to use the same working memory that is already locked.

For example, if you execute `BusinessWorks.InvokeProcess()` function in TIBCO BusinessEvents, then in the ActiveMatrix BusinessWorks process it calls, you cannot use an Invoke RuleFunction activity to invoke a rule function in the same agent. You could invoke a rule function running in a different agent, however.

Design Considerations Related to Container

The engine used as the container is responsible for state and object management, fault tolerance, and logging. The contained engine delegates management of these features to the container. The main points to keep in mind are listed below.

Table 62 Design Considerations Related to Container (Integration with ActiveMatrix BusinessWorks)

Item	TIBCO BusinessEvents Containers	ActiveMatrix BusinessWorks Containers
State and OM	TIBCO BusinessEvents manages state. See also Fault Tolerance below.	ActiveMatrix BusinessWorks manages state. Use In Memory object management in TIBCO BusinessEvents. Other OM methods are not supported.
Engine TRA file	The ActiveMatrix BusinessWorks engine TRA file is not used. Provide any properties needed, such as classpath and palette path, in the TIBCO BusinessEvents engine TRA file (see Configuring the Environment for ActiveMatrix BusinessWorks Containers on page 617).	The TIBCO BusinessEvents engine TRA file is not used. Provide any properties needed for TIBCO BusinessEvents in the ActiveMatrix BusinessWorks property file (see Configuring the Environment for ActiveMatrix BusinessWorks Containers on page 617).
CDD file	Provide configuration values for TIBCO BusinessEvents.	The CDD file is also used. Provide configuration values for TIBCO BusinessEvents.
TIBCO Hawk microagent	With configuration of a property in the TIBCO BusinessEvents CDD file, you can use TIBCO Hawk methods for TIBCO BusinessEvents and ActiveMatrix BusinessWorks. See Configuring the Environment for ActiveMatrix BusinessWorks Containers on page 617.	Only the TIBCO Hawk methods for ActiveMatrix BusinessWorks are used.
Fault Tolerance	TIBCO BusinessEvents fault tolerance is used. See Fault Tolerance With a TIBCO BusinessEvents Container on page 616.	ActiveMatrix BusinessWorks fault tolerance is used. Do not configure TIBCO BusinessEvents in fault tolerant mode.
Logging	TIBCO BusinessEvents manages logging.	ActiveMatrix BusinessWorks manages logging

Fault Tolerance With a TIBCO BusinessEvents Container

When TIBCO BusinessEvents is the container, ActiveMatrix BusinessWorks checkpointing can be used in a limited way. Do not use ActiveMatrix BusinessWorks checkpointing in any process that is called by `startProcess()` or `invokeProcess()`, or in any process called by such processes.

When an active TIBCO BusinessEvents processing unit (PU) fails, standby agents in other processing units become active and take over for the agents that failed. The newly active agents (which are of the same agent class) start the ActiveMatrix BusinessWorks engine as needed.

On failback, agents of lower priority become standbys. During deactivation they stop the ActiveMatrix BusinessWorks engine they were running.

Tips for Working With ActiveMatrix BusinessWorks Containers

Keep the following points in mind when designing your project.

- The TIBCO BusinessEvents hot deployment feature is not available.
- The internal TIBCO BusinessEvents engine that ActiveMatrix BusinessWorks starts communicates only with the ActiveMatrix BusinessWorks process that invokes it. TIBCO BusinessEvents channels do not listen for incoming messages.
- Do not execute any action in a TIBCO BusinessEvents startup rule function that results in use of `BusinessWorks.invokeProcess()` or `BusinessWorks.startProcess()` functions. The ActiveMatrix BusinessWorks engine may not be fully initialized when these functions are executed in a startup rule function (or in any rule in the RTC cycle of a startup rule function).

Configuring the Environment for ActiveMatrix BusinessWorks Containers

To use the integration features you must configure the environment as explained in this section.

- TIBCO BusinessEvents Studio and ActiveMatrix BusinessWorks must be installed on the same machine.
- If you are upgrading an integration project created in an earlier release, see Chapter 4, Migrating Projects from Earlier Versions in *TIBCO BusinessEvents Installation* for actions you must take.
- The values shown in this section work with various test projects. Depending on the ActiveMatrix BusinessWorks services used in your project, you may have to make additional changes.

Task A Configure the TRA Files

You must add various items to the ActiveMatrix BusinessWorks class path and palette path. The `designer.tra` file is used at design time. The `bwengine.tra` file is used at runtime. Update them as explained next.

Remember to add a path separator variable, `%PSP%` to separate the paths, when you paste path content into a TRA file.

If any properties have been modified after installation, examine the paths carefully. Ensure that all paths are specified with appropriate variables.

1. Open these files in a text editor:

   — *BE_HOME*`/bin/be-engine.tra` (as the source of values)

   — *TIBCO_DESIGNER_HOME*`/bin/designer.tra`

   — *BW_HOME*`/bin/bwengine.tra`

2. Copy the **`tibco.env.BE_HOME`** and **`tibco.env.AS_HOME`** environment variables from `be-engine.tra` to the `designer.tra` and `bwengine.tra` files.

   (You only need to copy the **`tibco.env.AS_HOME`** environment variable if you use TIBCO BusinessEvents DataGrid as your cache provider.)

3. Update the following properties in `designer.tra`:

   a. Locate the custom class path property **`tibco.env.CUSTOM_CP_EXT`** and append the following:

   — A **`%PSP%`** separator

   — The entire contents of `be-engine.tra` property **`tibco.env.STD_EXT_CP`**

   — The following text:

   ```
   %PSP%%BE_HOME%/hotfix/lib/palettes%PSP%%BE_HOME%/lib/palette
   s%PSP%%BE_HOME%/hotfix/lib/ext/tpcl/jide%PSP%%BE_HOME%/lib/e
   xt/tpcl/jide
   ```

   b. Locate the custom palette path property **`tibco.env.CUSTOM_PALETTE_PATH`** and append the following text to the property:

   ```
   %PSP%%BE_HOME%/hotfix/lib/palettes%PSP%%BE_HOME%/lib/palette
   s
   ```

4. Locate the custom class path property **`tibco.env.CUSTOM_EXT_PREPEND_CP`** in `bwengine.tra` and append the following:

   — A **`%PSP%`** separator

   — The entire contents of `be-engine.tra` property **`tibco.env.STD_EXT_CP`**

   — The following text:

   ```
   %PSP%%BE_HOME%/hotfix/lib/palettes%PSP%%BE_HOME%/lib/palette
   s%PSP%%BE_HOME%/hotfix/lib/ext/tpcl/jide%PSP%%BE_HOME%/lib/e
   xt/tpcl/jide
   ```

   — Also append the following text:

   ```
   %PSP%%BE_HOME%/hotfix/lib/palettes%PSP%%BE_HOME%/lib/palette
   s
   ```

Configuring the Environment For TIBCO BusinessEvents Containers

Using a TIBCO BusinessEvents container requires additional configuration so that TIBCO BusinessEvents can locate ActiveMatrix BusinessWorks resources at runtime.

First perform all steps in the section Configure the TRA Files on page 617 and then all steps in this section.

Task A Add the Repo URL for BusinessWorks to the CDD

This task assumes all other cluster deployment descriptor (CDD) configuration has been done. For more information about editing the CDD file, see Chapter 23, Cluster Deployment Descriptor (CDD), on page 385.

1. Open the TIBCO BusinessEvents project in TIBCO BusinessEvents Studio.
2. Open the CDD Editor resource and select the Agent Classes tab.
3. For all inference agent classes whose instances will run ActiveMatrix BusinessWorks, edit the BusinessWorks Repo URL field: add the path to the runtime TIBCO Designer project.
4. Save.

Use this CDD when deploying the TIBCO BusinessEvents project.

Example BusinessWorks Repo URL filed values

The repo URL format depends on the deployment transport used. Supported formats for the URL are `tibcr`, `http`, `https`, and `file`.

Rendezvous transport format:

```
tibco.bwrepourl=tibcr@domain name-deployment name:service=repo rvService:daemon=repo
rvDaemon:userName=uid:server=domain_name:password=encrypted_ password
```

Local transport (also known as file transport) format:

```
tibco.bwrepourl=domain_home>/domain_name/datafiles/deployment_name_root
```

HTTP transport format:

```
tibco.bwrepourl=http://machine_name:domain_http_port?domain_name-deployment-name&server=domain
_name&timeout\=600&userName=uid&password\=encrypted_password
```

Task B Modify the TIBCO BusinessEvents TRA File

be-engine.tra

1. Open the *BE_HOME*`/bin/be-engine.tra` file for editing.
2. Add the following properties and their values. Take values from the *BW_HOME*`/bin/bwengine.tra` file, if they are present there:

```
tibco.env.BW_HOME
tibco.env.BW_MIGRATION_APPEND_VERSION
tibco.env.BW_PLUGINS_HOME_OLD
tibco.env.BW_PLUGINS_HOME
tibco.env.TPCL_HOME
tibco.env.TRA_HOME
tibco.env.TRA_APPEND_VERSION
tibco.env.HAWK_HOME
java.property.palettePath
```

3. Copy the following properties and their values from the `bwengine.tra` file to the `be-engine.tra` file and rename them: add `BW_` to the property name as shown.

In bwengine.tra	In be-engine.tra
`tibco.env.CUSTOM_EXT_PREPEND_CP`	`tibco.env.BW_CUSTOM_EXT_PREPEND_CP`
`tibco.env.CUSTOM_EXT_APPEND_CP`	`tibco.env.BW_CUSTOM_EXT_APPEND_CP`
`tibco.env.STD_EXT_CP`	`tibco.env.BW_STD_EXT_CP`
`tibco.env.PATH`	`tibco.env.BW_PATH`
`tibco.env.LD_LIBRARY_PATH`	`tibco.env.BW_LD_LIBRARY_PATH`
`tibco.env.SHLIB_PATH`	`tibco.env.BW_SHLIB_PATH`
`tibco.env.LIBPATH`	`tibco.env.BW_LIBPATH`

4. In the value of all properties listed in step 3, remove references to any TIBCO BusinessEvents library, if there are any.

   The `be-engine.tra` file (as shipped) uses the above names in various path properties, so you do not have to make any additional modifications.
5. Save the file.

Task C Configure for Non-Default TIBCO Hawk Microagent Names (if Used)

When TIBCO BusinessEvents is the container, the ActiveMatrix BusinessWorks TIBCO Hawk microagent (HMA) name is the same as the TIBCO BusinessEvents HMA name, appended with `-bw`. The name is defined using the `Hawk.AMI.DisplayName` property (see Table 63, ActiveMatrix BusinessWorks integration Properties for TIBCO BusinessEvents Containers, on page 622 for details).

If you want to use default HMA names, skip this task.

Choose a method of specifying non default names:

- If you want to use the same non-default names, except that the ActiveMatrix BusinessWorks HMA is appended with `-bw`: Define the non-default name by adding the `Hawk.AMI.DisplayName` property to the Processing Units tab property sheet in the Cluster Configuration editor.
- If you want to use different names for the ActiveMatrix BusinessWorks and TIBCO BusinessEvents HMAs, do the following: Define the name to be used by the ActiveMatrix BusinessWorks HMA using `Hawk.AMI.DisplayName` property. Whatever name you specify will be automatically appended with `-bw`.
- Add a property called `be.hawk.microagent.name` and define its value as the desired name of the TIBCO BusinessEvents HMA.

Task D Configure for ActiveMatrix BusinessWorks Checkpointing (if Used)

Used only when TIBCO BusinessEvents is the container. See Design Considerations Related to Container on page 615 for advice on using checkpointing in this case.

By default, an ActiveMatrix BusinessWorks instance running inside TIBCO BusinessEvents has the same name as the TIBCO BusinessEvents instance. If you will use ActiveMatrix BusinessWorks checkpointing, and if TIBCO BusinessEvents engine instances are deployed with different names, you must add a property that specifies the same ActiveMatrix BusinessWorks instance name to all processing units. Add the property to all processing units, using the Cluster Configuration editor, Processing Units tab property sheet.

`tibco.bwengine.name` *all-same-engine-name*

You can specify the property and its value in the CDD, or in the TIBCO Administrator Advanced tab at deploy time.

Table 63 ActiveMatrix BusinessWorks integration Properties for TIBCO BusinessEvents Containers

Property	Notes
`tibco.bwengine.name`	The name of the ActiveMatrix BusinessWorks engine. Used for ActiveMatrix BusinessWorks integration projects where TIBCO BusinessEvents is the container, and used only if the following is true: • You will use ActiveMatrix BusinessWorks checkpointing • TIBCO BusinessEvents engine instances are deployed with different names. Use this property to ensure that the same ActiveMatrix BusinessWorks engine name is specified in all nodes. Available in TIBCO Administrator by default. No default value.
`Hawk.AMI.DisplayName`	The name of the TIBCO Hawk microagent (HMA) instance used for TIBCO BusinessEvents. You can change the name as desired. The default name uses this format: `com.tibco.Adapter.{be-engine|bw-engine}.`*DOMAIN*`.` *DEPLOYMENT*`.`*COMPONENT_INSTANCE* For container mode TIBCO BusinessEvents-ActiveMatrix BusinessWorks integration projects where TIBCO BusinessEvents is the container, TIBCO BusinessEvents internally appends `-bw` to this name and uses it for the ActiveMatrix BusinessWorks HMA name. However, if `be.hawk.microagent.name` is used the behavior is different. See `be.Hawk.microagent.name` for related information.

Table 63 ActiveMatrix BusinessWorks integration Properties for TIBCO BusinessEvents Containers (Cont'd)

Property	Notes
`be.hawk.microagent.name`	This property is used only for container mode TIBCO BusinessEvents-ActiveMatrix BusinessWorks integration projects where TIBCO BusinessEvents is the container. This property is not present by default. You must add it if you want to use it. If you want to use different names for the ActiveMatrix BusinessWorks and TIBCO BusinessEvents TIBCO Hawk microagents (HMAs), but you do not want to use the `Hawk.AMI.DisplayName` naming scheme (where both HMAs have the same name, except that the ActiveMatrix BusinessWorks name is appended with `-bw`), then do the following: • Define the name of the TIBCO BusinessEvents HMA using `be.hawk.microagent.name`. • Define the name of the ActiveMatrix BusinessWorks HMA using `Hawk.AMI.DisplayName` (or just use the default).

Configuring a RuleServiceProvider Configuration Resource

You reference a RuleServiceProvider Configuration resource in various activities so ActiveMatrix BusinessWorks can locate the TIBCO BusinessEvents resources used in the integration.

1. Open the ActiveMatrix BusinessWorks project in TIBCO Designer and open the project folder where you want to store the resource.
2. Right-click in the design panel and select **Add Resource > TIBCO BusinessEvents Activities > TIBCO BusinessEvents RuleServiceProvider Configuration**.
3. Enter values in the Configuration tab, following guidelines in TIBCO BusinessEvents RuleServiceProvider Configuration Resource Reference on page 625.
4. **Click** Apply and save the resource.

A RuleServiceProvider Configuration shared resource is also available In TIBCO BusinessEvents Studio. It can be used during migration to display a migrated version 3.x resource for informational purposes.

TIBCO BusinessEvents RuleServiceProvider Configuration Resource Reference

This resource specifies which TIBCO BusinessEvents project to use for the integration. See also Configuring a RuleServiceProvider Configuration Resource on page 624.

Configuration

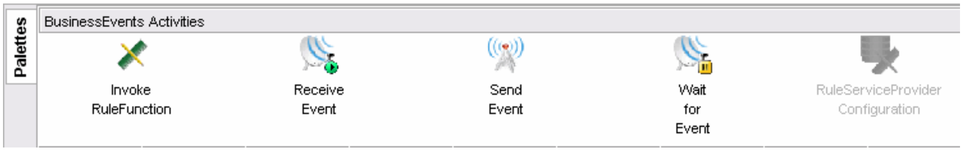
The Configuration tab has the following fields.

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 on page 313.
Description	No	Short description of the resource.
Repo URL/EAR Path	Yes	One of the following: Path to and name of the TIBCO BusinessEvents EAR file. Ontology definitions in this EAR file are used by activities in the TIBCO BusinessEvents activities palette. (At design or testing time you can use an exact copy of the EAR that is used for production. You can use a global variable so that you can define the location of the EAR at deploy time.) or The server-based repository URL for a TIBCO BusinessEvents project that was deployed to a TIBCO Administrator domain.

Field	Global Var?	Description
Deployment Descriptor Path	Yes	Path to the CDD (Cluster Deployment Descriptor) file used by the deployed TIBCO BusinessEvents application. At design or testing time you can use an exact copy of the CDD that is used for production. You can use a global variable so that you can define the location of the CDD at deploy time.
Processing Unit ID		ID of a processing unit in the specified CDD whose values are used for the integration.

Working With the TIBCO BusinessEvents Activities

This guide assumes you know how to work with ActiveMatrix BusinessWorks processes. See *TIBCO BusinessWorks Process Design Guide* if you need detailed guidance.



The TIBCO BusinessEvents Activities palette allows you to easily send and receive TIBCO BusinessEvents events in TIBCO ActiveMatrix BusinessWorks applications.

Before you work with other activities, configure a RuleServiceProvider Configuration resource so that the ActiveMatrix BusinessWorks project can locate TIBCO BusinessEvents resources. See TIBCO BusinessEvents RuleServiceProvider Configuration Resource Reference on page 625.

To Work with the TIBCO BusinessEvents Activities

1. Open the folder in which you want to add a process definition. Right-click in the design panel, and select **Add Resource > Process > Process Definition**.
2. Double-click the Process Definition resource to open it. You see the Start and End activities.
3. Right-click in the design panel, and select **Add Resource > TIBCO BusinessEvents Activities**. Select activities as needed to configure the process. See the following for guidelines:
   - Receive Event Resource Reference on page 628
   - Send Event Resource Reference on page 630
   - Wait for Event Resource Reference on page 631
   - Invoke RuleFunction Resource Reference on page 638
4. Click **Apply** and save the resource.

Receive Event Resource Reference

This process starter starts a process when the specified simple event arrives from its default destination channel, or from a destination specified in the configuration tab.

Configuration

The Configuration tab has 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 on page 313.
Description	No	Short description of the resource.
RuleServiceProvider Configuration		Browse to and select the RuleServiceProvider Configuration resource used in this process. The lists of events and destinations shown in the Event Reference and Custom Destination fields comes from the EAR file referenced in the selected RuleServiceProvider Configuration resource.
Event Reference	No	Browse to and select the simple event you want to receive.
Custom Destination	No	If you do not want to listen for the specified event on its default destination, browse to and select an alternate destination.

Misc

The Misc tab has the following fields.

Field	Description
Sequencing Key	This field can contain an XPath expression that specifies which processes should run in order. Process instances with sequencing keys that evaluate to the same value will be executed sequentially in the order the process instance was created. See *TIBCO ActiveMatrix BusinessWorks Process Design Guide* for more information about controlling the execution order of process instances and about XPath expressions.
Custom ID	This field can contain an XPath expression that specifies a custom ID for the process instance. This ID is displayed in the View Service dialog of TIBCO Administrator, and it is also available in the `$_processContext` process variable.

Output

The Output tab has the following fields.

Output Item	Data Type	Description
`BEReceiveEventOutput`	complex	Expand `BEReceiveEventOutput` to show the name, properties, and attributes of the event received.

Send Event Resource Reference

Data from the ActiveMatrix BusinessWorks process context is used to send an event of a specified type to its default destination or to a destination specified in the configuration tab.

Configuration

The Configuration tab has 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 on page 313.
Description	No	Short description of the resource.
RuleServiceProvider Configuration		Browse to and select the RuleServiceProvider Configuration resource used in this process. The lists of events and destinations shown in the Event Reference and Custom Destination fields comes from the EAR file referenced in the selected RuleServiceProvider Configuration resource.
Event Reference	No	Browse to and select the simple event you want to send.
Custom Destination	No	If you do not want to send the specified event to its default destination, browse to and select an alternate destination.

Input

The Input tab has the following fields.

Input Item	Data Type	Description
`BESendEventInput`	complex	Expand `BESendEventInput` to show the name, properties, and attributes of the event type selected in the Configuration tab.

Wait for Event Resource Reference

This activity listens to the default destination of a specified event type or to a destination specified in the configuration tab, and waits for a matched simple event.

Configuration

The Configuration tab has 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 on page 313.
RuleServiceProvider Configuration		Browse to and select the RuleServiceProvider Configuration resource used in this process. The lists of events and destinations shown in the Event Reference and Custom Destination fields comes from the EAR file referenced in the selected RuleServiceProvider Configuration resource.
Description	No	Short description of the resource.
Event Reference	No	Browse to and select the simple event you want to wait for.
Custom Destination	No	If you do not want to listen for the specified event on its default destination, browse to and select an alternate destination.

Event

The Event tab has the following fields.

Input Item	Global Var?	Description
Candidate Event Key	No	Use to filter events. Only events whose candidate event key matches the key provided will trigger the activity. For example you can use the event's extID as the event key. You can also use any of the event's properties as a matching candidate event key. There can be multiple keys, but only the first one that matches is considered. As an example use of this property, suppose you want "Wait for Event" to fire only for the 10th occurrence of an event. You would set the Candidate Event Key value to 10. In the input tab, you would map the input item `key` to a global variable that is incremented by 1 each time the event occurs. Refer to *ActiveMatrix BusinessWorks Palette Reference* sections on other "Wait For" types of activities (for example, for JMS queues or topics, or for Rendezvous) for more information.
Event Timeout	No	If the event is received before the process reaches this activity, the event waits for this number of milliseconds. If the timeout period ends before the process reaches this activity, the event is discarded.

Input

The Input tab has the following fields.

Input Item	Data Type	Description
`key`	String	Candidate Event Key value (specified on the Event tab
`processTimeout`	Integer	The `processTimeout` value is the Event Timeout from Event tab

Output

The Output tab has the following fields.

Input Item	Data Type	Description
`BEReceiveEventOutput`	complex	Expand `BEReceiveEventOutput` to show the name, properties, and attributes of the event type selected in the Configuration tab.

Invoking a TIBCO BusinessEvents Rule Function from a Process

To enable an ActiveMatrix BusinessWorks process to call a TIBCO BusinessEvents rule function, you need to configure an Invoke RuleFunction activity.

See Integration Components on page 612 for an introduction to the integration features.

- See Thread Management on page 614 for important information about avoiding deadlock.
- Be careful in using the `ThreadLocal` variable because the thread of execution may not be same as that in the prior activity.
- Concepts passed to the rule function are not automatically asserted. Similarly, events passed to the rule function are not automatically asserted. You must assert them explicitly.
- `Object` support is not provided. You can't set `Object` in the scope of the rule function or as its return value. You must specify a specific type of TIBCO BusinessEvents object or a primitive (other than `Object`).

Specifying Input Arguments

All TIBCO BusinessEvents primitives (except Object) are supported. Appropriate boxing and unboxing of XML primitive types to TIBCO BusinessEvents primitive types is handled. For more details, see Chapter 20, Rule Language Datatypes, on page 329.

You can pass null values. Select the argument in the Activity Input panel of the Input tab. Click the Edit Statement (exclamation point) button. In the Edit Statement dialog, Content tab, check the Set Explicit Nil checkbox. If `xsi:nil==true`, then a null is passed as the argument.

Base Concept and Base Event are supported as argument types. You must map a specific type to the base concept or base event. Open the Edit Statement dialog, and in the Type tab, check the Type Substitution check box. In the Type field, select XML Type Reference. Select a Schema and a Type.

Using Synchronous Invocation

Execution is synchronous. The process waits for the rule function to return a value. The rule function acquires a lock on TIBCO BusinessEvents working memory until the RTC cycle completes and then returns a value to the process.

You can use more than one Invoke RuleFunction activity in a process only if each uses a different agent. See Thread Management on page 614 for more details.

Using the lockWM Parameter

By default the working memory is locked during the invocation (that is, the value is true). You can change this behavior by setting the `lockWM` parameter to false.

When you set the value to false, the rule function executes outside the context of the working memory.

The results must not modify concept instances or scorecards (as in the case of a preprocessor).

Set the `lockWM` parameter to false for optimization purposes only and use with care. Set it to false only if the call does not modify the working memory in any way. For example, use it to perform stateless calls or calculations, or to create new concept instances or events, or for lookups and so on.

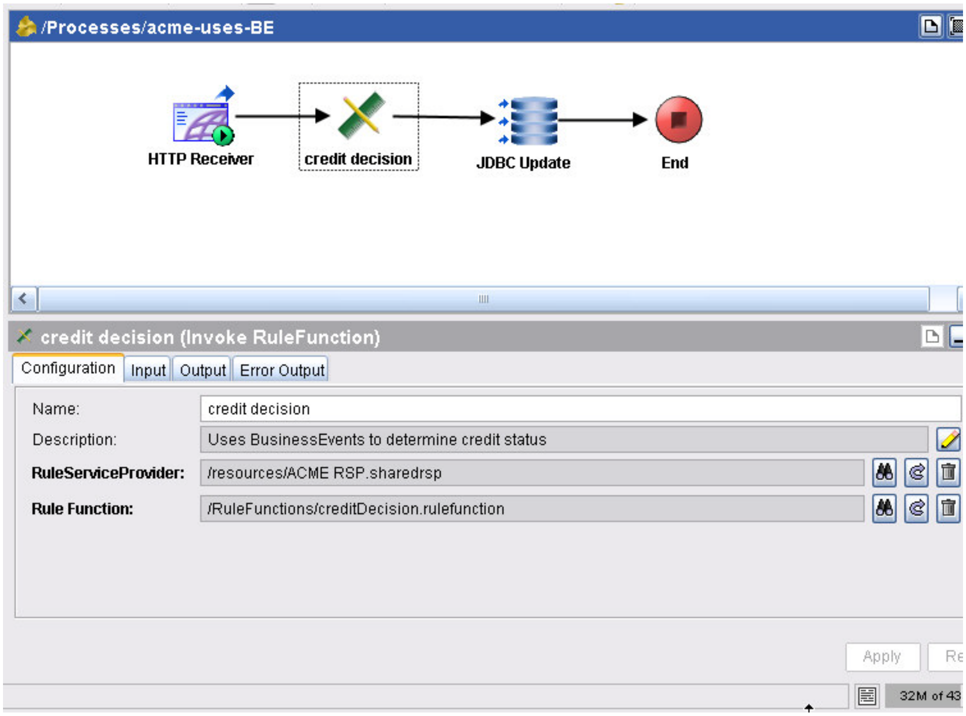
Overriding the Rule Function at Runtime

You can override which function is called at runtime. To do so, enter a global variable as the value for `rulefunction` in the input tab. At runtime, you can specify a rule function with a different name but the same signature as the one specified in the activity's Configuration tab.

You can only use rule functions that are in the TIBCO BusinessEvents EAR file deployed for the integration project.

Working With Invoke RuleFunction Activities

See Invoking a TIBCO BusinessEvents Rule Function from a Process on page 634 for important information about using the Invoke RuleFunction activity in your ActiveMatrix BusinessWorks process.



Name: credit decision

Description: Uses BusinessEvents to determine credit status

RuleServiceProvider: /resources/ACME RSP.sharedrsp

Rule Function: /RuleFunctions/creditDecision.rulefunction

To Configure an Invoke RuleFunction Activity

1. Start TIBCO Designer and open (or add) the ActiveMatrix BusinessWorks process in which you want to use an Invoke RuleFunction activity.
2. Right-click in the design panel and select **Add Resource > TIBCO BusinessEvents Activities > Invoke RuleFunction**. Link the activity to other activities as appropriate.

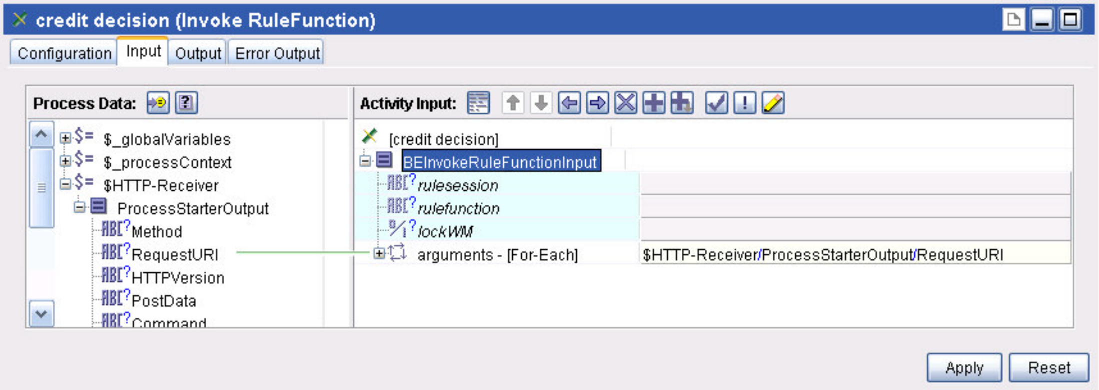
3. Complete the Configuration tab, following guidelines provided in Invoke RuleFunction Resource Reference on page 638.

   — Name the activity and give it a description as desired.

   — Browse to and select the RuleServiceProvider Configuration resource that contains the ontology for the project you are integrating with.

   — Browse to and select the TIBCO BusinessEvents rule function you want to invoke.

4. Select the **Input** tab. Expand each item to see all fields. Enter values or map values from process data. Follow guidelines provided in Invoke RuleFunction Resource Reference on page 638.



credit decision (Invoke RuleFunction)

Configuration | Input | Output | Error Output

Process Data:
- $_globalVariables
- $_processContext
- $HTTP-Receiver
  - ProcessStarterOutput
    - Method
    - RequestURI
    - HTTPVersion
    - PostData
    - Command

Activity Input:
- [credit decision]
  - BEInvokeRuleFunctionInput
    - rulesession
    - rulefunction
    - lockWM
    - arguments - [For-Each]: $HTTP-Receiver/ProcessStarterOutput/RequestURI

5. Click **Apply** and save the resource.

Invoke RuleFunction Resource Reference

To enable an ActiveMatrix BusinessWorks process to call a TIBCO BusinessEvents rule function, you configure one or more Invoke RuleFunction activities according to your needs. Execution is synchronous. See Invoking a TIBCO BusinessEvents Rule Function from a Process on page 634 and Working With Invoke RuleFunction Activities on page 636 for more details.

Configuration

The Configuration tab has 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 on page 313.
Description	No	Short description of the resource.
RuleServiceProvider Configuration		Browse to and select the RuleServiceProvider Configuration resource used in this process. The list rule functions you can select for the Rule Function field comes from the EAR file referenced in the selected RuleServiceProvider Configuration resource.
Rule Function	No	Select the desired rule function from the TIBCO BusinessEvents project.

Input

The Input tab has the following fields.

Field	Global Var?	Description
`rulesession`	Yes	Specifies the agent where the rule function is to execute. Optional if the processing unit has only one agent To specify a value, enter the agent name (as specified in the CDD file).
`rulefunction`	Yes	Optional. Allows you to override the rule function specified in the Configuration tab. Enter a global variable that is used to specify the name of the rule function at runtime. Do not enter the `.rulefunction` extension. The rule function you specify must have the same signature as the function specified in the Configuration tab (and a different name). See Overriding the Rule Function at Runtime on page 635
`lockWM`	Yes	Default value is true. The working memory is locked during the invocation. You can set the value to false only if conditions explained in the section Using the lockWM Parameter on page 635 are met.
`arguments`	No	Displays the arguments for the rule function, if it has any. Map the available process data to the activity input, or enter values in the fields as appropriate for the datatype of each argument.

Output

The Output tab has the following fields.

Output Item	Data Type	Description
Return	(Varies)	The return type for the specified rule function. Return can be void or any of the following types: String, integer, long, double, Boolean, datetime, concept, event.

Working with the BusinessWorks Functions

The integration model provides a BusinessWorks category of functions. This category is located in the General Functions library. Use and configuration of the functions in the BusinessWorks category are described in this section. See Integration Components on page 612 for a brief introduction to these features.

Providing Paths to TIBCO BusinessEvents Project Resources Using Schemas

The `BusinessWorks.invokeProcess()` and `BusinessWorks.startProcess()` functions can send events to and receive events from ActiveMatrix BusinessWorks processes. However, the paths to TIBCO BusinessEvents project resources are not available by default in TIBCO Designer. To use these functions successfully, you must generate the event schemas in TIBCO BusinessEvents and import them into the TIBCO Designer project. See Exporting (Generating) Concept and Event Schema (XSD) Files on page 5 for details on exporting the event schemas.

In the TIBCO Designer project, import these schemas and use them in the relevant processes, so that the event resource path is known to TIBCO Designer. See ActiveMatrix BusinessWorks documentation for details.

Using invokeProcess()

Purpose

Use this function to take advantage of ActiveMatrix BusinessWorks features. For example, you could use `invokeProcess()` to send customer information to an ActiveMatrix BusinessWorks process that gets the discount information from a database and returns it to TIBCO BusinessEvents.

Runtime Behavior

If the ActiveMatrix BusinessWorks engine is not already started, TIBCO BusinessEvents will start it before `invokeProcess()` executes. See Using init() on page 644 for another way to start the engine.

When a rule containing or calling `invokeProcess()` is triggered, `invokeProcess()` calls the specified ActiveMatrix BusinessWorks process and passes it an event. The `invokeProcess()` function executes synchronously, so the TIBCO BusinessEvents rule engine waits. The ActiveMatrix BusinessWorks process performs its work and returns an event or null at completion of the process, or it times out if you set a timeout that is exceeded. It generates an advisory event if it times out. Events received are not automatically asserted.

Error Handling and Advisory Events

TIBCO BusinessEvents asserts an advisory event and returns `null` if the ActiveMatrix BusinessWorks process fails, or if the invocation times out. The advisory event category is `Engine`, and the type is `INVOKE BW PROCESS`. The message contains the error message from the failed ActiveMatrix BusinessWorks process, or the timeout message. The advisory event is also created if the process is cancelled using `cancelProcess()`, because the system cannot differentiate between different causes for the process stopping before completion.

Because the `invokeProcess()` function returns null when an error occurs, you must handle the possibility of a `null` return in rules that use any property or attribute of the returned event.

See Chapter 11, Advisory Events, on page 177 for more information about advisory events.

Do not use `invokeProcess()` more than once in the same thread of execution. See Thread Management on page 614.

Configuring the Function

When configuring the `invokeProcess()` function in a rule or rule function, specify the parameters as follows:

- The ActiveMatrix BusinessWorks process that the function invokes. The specified process must not contain a process starter.
- The event to pass to the process (or specify null if you don't want to pass an event to the process).
- A timeout (or specify zero if you don't want to use a timeout).

For example, you send customer information from an event (alias `neworder`) to an ActiveMatrix BusinessWorks process, which returns an event with the discount level to offer:

```
Events/Discountlevel Discount;
Discount = Businessworks.invokeProcess("/Processes/CustInfo",
neworder, 0);
```

Timeouts If you set a timeout period and a timeout occurs, the rule or rule function containing `InvokeProcess()` continues without waiting for the ActiveMatrix BusinessWorks process to complete. If you use a timeout, set it to a period long enough for the ActiveMatrix BusinessWorks process to complete. Include logic to handle the case that the `invokeProcess()` function does time out.

Configuring the Process

Configure the Start activity of the specified ActiveMatrix BusinessWorks process to accept the event passed by the `invokeProcess()` function, if an event is passed. If you specify null, of course, this step is not required.

Configure the rest of the activities in the process to carry out whatever processing is desired using the data passed into it by `invokeProcess()`.

Similarly, in the End activity, specify the event type to return to the `invokeProcess()` process.

The returned event is then used as needed by the logic of the rule or rule function.

Events returned are not asserted. You must explicitly assert them as needed.

See Providing Paths to TIBCO BusinessEvents Project Resources Using Schemas above for additional setup you must do.

Using startProcess()

Purpose

Use the `startProcess()` function when you want to invoke an ActiveMatrix BusinessWorks process that performs work that can be completed asynchronously. The `startProcess()` function invokes an ActiveMatrix BusinessWorks process in asynchronous mode and immediately returns the job ID of the process. Rule processing continues. When the ActiveMatrix BusinessWorks process completes, it passes an event to a callback rule function that is specified in a `startProcess()` argument. For example, you send order information to an ActiveMatrix BusinessWorks order fulfillment process. When the order ships, notification is returned to TIBCO BusinessEvents, which updates a customer concept instance.

Runtime Behavior

If the ActiveMatrix BusinessWorks engine is not already started, TIBCO BusinessEvents will start it before `startProcess()` executes. See Using init() on page 644 for another way to start the engine.

When a rule containing or calling `startProcess()` is triggered, `startProcess()` calls the specified ActiveMatrix BusinessWorks process and passes it an event. `startProcess()` returns the `jobID` of the ActiveMatrix BusinessWorks process. (You can use the returned job ID, for example, in the `cancelProcess()` rule function.) The function executes asynchronously, so the TIBCO BusinessEvents rule engine continues while at the same time the ActiveMatrix BusinessWorks process executes. The ActiveMatrix BusinessWorks process performs its work and passes an event to the callback (`ruleFnURI`) rule function specified in the `startProcess()` function arguments. The `ruleFnURI` rule function performs its work, for example, creating and asserting the event.

The `ruleFnURI` rule function must not modify concept instances or scorecards.

Configuring the Function

When configuring the `startProcess()` function in a rule or rule function, specify the following in the parameters:

- The ActiveMatrix BusinessWorks process that the function invokes. The specified process must not contain a process starter.
- The event to pass to the process (or specify null if you don't want to pass an event to the process).
- The TIBCO BusinessEvents rule function that the process calls on completion (the callback rule function). The required signature for the callback rule function is shown below.

Configuring the ruleFnURI Rule Function

The callback rule function, specified in the `startProcess()` `ruleFnURI` argument, is called when the ActiveMatrix BusinessWorks process completes. Add the `ruleFnURI` rule function to the TIBCO BusinessEvents project.

For convenience, this rule function is referred to as the `ruleFnURI` rule function in this section. The `ruleFnURI` rule function must have the following signature, and the Validity field (in the Configuration tab) must be set to Action:

```
void ruleFn(long jobID, int status, Event outputEvent, Object
closure)
```

- **`jobID`** Type: `long.` The job id can be used to correlate the information passed to `ruleFnURI` with related information in TIBCO BusinessEvents.
- **`status`** Type: `int`. Returns 0 (zero) if the process completed successfuly, and `-1` if the process did not complete successfully (for example because `cancelProcess()` was called).
- **`outputEvent`** Type: `Event`. An event passed to the rule function by the ActiveMatrix BusinessWorks process. It can be created by the process, or it can be an existing event. If the process fails or is cancelled (for example because `cancelProcess()` was called), an advisory event is returned.
- **`closure`** Type: `Object`. A closure object could be, for example, a value from the original context that has to be passed back to the TIBCO BusinessEvents engine, for example, a loan rate that has been promised. (Note that type is Object, so you cannot pass an event or concept.)

Configuring the Process

Configure the Start activity of the specified ActiveMatrix BusinessWorks process to accept the event passed by the `startProcess()` function, if an event is passed. If you specify null, of course, this step is not required.

Configure the rest of the activities in the process to carry out whatever processing is desired using the data passed into it by `startProcess()`.

In the End activity, specify the event type that is to be passed to the specified ruleFnURI rule function.

See Providing Paths to TIBCO BusinessEvents Project Resources Using Schemas on page 640 above for additional setup you must do.

The returned event is then used as needed by the logic of the rule or rule function. Note that if you want to assert the event that is returned, you must explicitly assert it.

Using cancelProcess()

Purpose

Use the `cancelProcess()` function to cancel a long running process, specified by `jobID`.

Cancellation may fail if the process has already completed before receiving the cancellation command. In this case, the following exception is thrown:

`java.lang.Exception: Job` *JobId* `not found`

Configuration

The job ID for the process you want to cancel is provided in the return value of `startProcess()`.

Using init()

Purpose

The `init()` function initializes the ActiveMatrix BusinessWorks engine if it is not already running. Use of `init()` when the engine is already running is harmless.

Use of `init()` is optional. It is provided as a convenience. For example you can use it in a TIBCO BusinessEvents startup rule function to initialize the ActiveMatrix BusinessWorks engine at startup, so as to save valuable time later.

If `invokeProcess()` or `startProcess()` are executed when the ActiveMatrix BusinessWorks engine is not already started, they will start the engine at that time.

Configuration

A general good practice is to call `init()` to start the ActiveMatrix BusinessWorks engine in a startup rule function (specified in the Startup/Shutdown tab of the BAR). The `init()` function takes no parameters. The rule function might simply contain this code:

```
BusinessWorks.init();
```

Using shutdown()

Purpose

The shutdown `()` function shuts down the ActiveMatrix BusinessWorks engine if it is running. Use of shutdown`()` when the engine is already shut down is harmless.

Use of `shutdown()` is optional. It is recommended that you use it only when you have finished using the ActiveMatrix BusinessWorks engine, and won't need to start it again. Stopping and restarting the engine is not necessary and can affect performance

Configuration

Simply place the shutdown command so that it is executed when needed. The `shutdown()` function takes no parameters. The rule function might simply contain this code:

```
BusinessWorks.shutdown();
```

Chapter 39 TIBCO BusinessEvents Performance Profiler

This chapter explains how you can run a 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.

Topics

Overview of Profiler

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 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, statistics data is written to the specified (or default) file and cleared from memory. The engine continues to run. If the engine stops before before the profiler completes, the file is not created.

You can execute the profiler and turn it off in three ways:

Using properties Profiler turns on when the agent initializes at system startup. Used to profile RTC time, including startup rule functions.

Using TIBCO BusinessEvents catalog functions 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 Profiler turns on by invoking a TIBCO BusinessEvents microagent Hawk method. 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 dynamically turn the profiler on and off.

Using a TIBCO BusinessEvents Monitoring and Management method Similar to using a Hawk method.

See Turning Profiler On and Off on page 650 for details.

See Profiler Reference on page 656 for a reference to all output data file column headings.

Changing 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.

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

This section explains the different ways you can turn the profiler on and off.

To Turn Profiler On and Off Using TIBCO BusinessEvents Monitoring and Management

If you have deployed the processing unit using TIBCO BusinessEvents Monitoring and Management (MM), you can turn the profiler on and off using the MM Console.

Use of MM is documented in the *TIBCO BusinessEvents Administration* guide. In particular, see the section To Execute a Method and the section Profiler Group, in Chapter 6, Monitoring and Managing a TIBCO BusinessEvents Cluster with MM.

To Turn Profiler On and Off 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.

Table 64 Profiler Configuration Properties (Sheet 1 of 3)

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.

Table 64 Profiler Configuration Properties (Sheet 2 of 3)

Property	Notes
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*.files 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 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.

Table 64 Profiler Configuration Properties (Sheet 3 of 3)

Property	Notes
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. 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.

To Turn Profiler On and Off Using Functions

This section assumes you understand how to use TIBCO BusinessEvents functions. It tells you which functions to use and the effect of each function.

> 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 Table 64, Profiler Configuration Properties, on page 650.

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 and starts collecting data for the specified duration for the agent in which the rule or rule function that calls this function is executed. The profiler starts collecting data at the beginning of 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 which case it is 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 Table 64, Profiler Configuration Properties, on page 650:

`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 executed.)

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 and 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.

To Turn Profiler On and Off 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

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 Table 64, Profiler Configuration Properties, on page 650.

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 next RTC for the specified duration.

This method works the same way as the `Engine.Profiler.startCollectingToFile()` function (see To Turn Profiler On and Off Using Functions on page 653), except that it requires you to specify an agent class.

Input arguments are the same as the engine properties shown in Table 64, Profiler Configuration Properties, on page 650:

`String` *session*: If you want to monitor multiple agents, execute the method once for each, specifying the agent class name in each case. If 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 and writes the profile data into a file for the specified agent when the current RTC has completed. You must execute 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.

Table 65 Profiler Column Heading Reference (Sheet 1 of 5)

Column Heading	Notes
Statistics Relating to the Overall RTC	

`RTC_Stats_Type`	Type of rule evaluation cycle (RTC) There are 10 different types: 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`	Total number of times the same RTC has been executed.

Table 65 Profiler Column Heading Reference (Sheet 2 of 5)

Column Heading	Notes
`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.
`MaxResolvedTime`	The maximum time in milliseconds spent to resolve a single RTC, including condition evaluation and action execution, but excluding operations related to object management (OM).
`MinResolvedTime`	The minimum time in milliseconds spent to resolve a single RTC, including condition evaluation and action execution, but excluding 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 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`	Total number of times this condition is evaluated in the same RTC.
`NumEvalTrue`	Total number of times the `Join` condition is evaluated to true. This value is the sum of `NumEvalTruePropagatedLeft` and `NumEvalTruePropagatedRight`.

Table 65 Profiler Column Heading Reference (Sheet 3 of 5)

Column Heading	Notes
`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`	Average rate that the condition evaluates to true and evaluation is triggered by object assertion propagated from the left side of the condition.
`MaxNumEvalTruePropagatedLeft`	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`	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.
`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`	Average rate that the condition evaluates to true and evaluation is triggered by object assertion propagated from the right side of the condition.

Table 65 Profiler Column Heading Reference (Sheet 4 of 5)

Column Heading	Notes
`MaxNumEvalTruePropagatedRight`	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`	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 first time the action execution begins.
`Description`	Information about the action corresponding to current action type. For example, description of type `ACTION-Rule-Action` is the name of the rule.
`NumExecuted`	Total number of times the same action has been executed. 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.

Table 65 Profiler Column Heading Reference (Sheet 5 of 5)

Column Heading	Notes
`TotalExecutionTime`	Total time in milliseconds spent on the total number of action execution phases. Action execution time is the time Rete network spends on executing 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 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 rule agenda as a result of all action operations.
`MinAgenda`	The minimum size of rule agenda as a result of all action operations.

Chapter 40 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.

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`

Topics

Overview of Testing and Debugging Projects

You can run and debug projects in TIBCO BusinessEvents Studio, using test data to understand how TIBCO BusinessEvents rules behave in response to inputs. Debugger shows potential problems in execution of rules and rule functions. You can also simply test a project by running it against test data. You can run multiple engines at the same time. For example, one engine's processing unit might run an inference agent while another's runs a cache agent.

The TIBCO BusinessEvents Decision Manager add-on product also uses a tester feature, Table Analyzer, to enable technical and non-technical users to test rules, rule functions, and decision tables. See TIBCO BusinessEvents Decision Manager documentation for details.

Debugging Projects

TIBCO BusinessEvents Debugger allows you to control the execution of a project by setting breakpoints, stepping through your code, suspending launched programs, examining the contents of variables, providing rule input, and so on.

TIBCO BusinessEvents Debugger integrates with the Eclipse Java development toolkit debugger. Much of the functionality is standard Eclipse debugger functionality. See the Eclipse help for details on features such as breakpoint preferences and other functionality. This chapter explains only the TIBCO BusinessEvents-specific features.

You can debug local projects using their CDD and EAR files, and you can also debug remote engines.

Declarative Programming and Stepping Step into, step over, and step return may not behave as you might expect. In a declarative rule language like TIBCO BusinessEvents, unlike with a procedural language, there is no predetermined path through the code. The inference engine logic determines the next action (next rule) based on various conditions and settings, not the rule itself.

Testing Projects

Running projects to test them works in a way similar to debugging, without the ability to add break points and so on. For testing purposes, you choose options to run the engine, rather than to debug the engine.

Launch Configurations

Before you debug or test a project, you define settings needed to launch the engine, using a *launch configuration* file.

You can add multiple launch configurations for one project, each configured for a different purpose. For example, you may want to test the effect of different startup arguments. Or if you are using cache OM, you could create one configuration for the processing unit (PU) that runs an inference agent, and one for the PU that runs the cache agent.

Test Data

When you test or debug an engine, you must assert test data directly to working memory. The test data in the working memory triggers rules in the normal way, so you can observe conflict resolution and run to completion cycles in the engine.

You can send data to a destination as you would in a runtime environment. For example, you can send messages from a JMS or Rendezvous source.

You do not have to have external sources, however. You can also provide test data using the following TIBCO BusinessEvents Studio features:

Test Data You can create and save concept and event instance data for later use. You can then assert this data selectively when running or debugging the engine.

Rule Data You can provide data as it is expected by rules in the Rule Data tab of the Debugger perspective. See Working with Rule Data on page 674.

See Creating and Working With Test Data on page 672.

The Rule Agenda and Variable Views

When you run the debugger and reach a a breakpoint in a rule or rule function the execution is suspended. You can now examine the state of things:

- The Variable view displays value of all the variables in scope.
- The Rule Agenda view displays rules to be executed from the rule agenda, excluding the current rule (where execution is suspended due to the breakpoint).

For example, Event1 is asserted, which causes Rule1 to execute. Rule1 creates an instance of Concept1. Rule2, Rule3, and Rule4 all have Event1 and Concept1 in scope, and their conditions are satisfied by the instance of Event1 and Concept1. If a breakpoint suspends execution in Rule2, then you see Rule3 and Rule4 in the Rule Agenda view.

For details about how the rule agenda is built, see Understanding Conflict Resolution and Run to Completion Cycles in *TIBCO BusinessEvents Architect's Guide*.

Viewing and Understanding Results

After every run, TIBCO BusinessEvents creates a consolidated results file in XML format with detailed information on the test run. This results XML file is easy to view and compare.

An editor also appears that displays the results in TIBCO BusinessEvents Studio.

See Viewing the Results on page 682.

Preparing to Run (Test) or Debug a Project

You can do these tasks in any order. For information on setting up Tester Preferences, see Tester Preferences on page 722.

Build an EAR File

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, check the **Generate Debug Info** checkbox (checked 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 (as Desired)

You can create test data for use across multiple sessions, or you can provide rule input data while the engine is running. See Creating and Working With Test Data, page 672.

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 the engine uses 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 on page 667. You must specify the same JDI port number in the TRA file and in the debug configuration.

To Configure Java Debug Interface (JDI)

For each TIBCO BusinessEvents engine you want to enable for remote debugging, do the following.

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,ser
   ver=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 that 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.)

To Add and Work with Launch Configurations

A reference to the settings is provided in Launch Configurations Reference on page 670.

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) on page 665.

1. Do one of the following:

   — To create a debug configuration, click the down-arrow to the right of the debugger button (🐞 ▾) on the toolbar. From the drop-down 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 drop-down 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. Do one of the following depending on your needs:

   — To edit a configuration, expand TIBCO BusinessEvents Application or Remote TIBCO BusinessEvents and select an existing debug configuration.

   — To add a new 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 selected TIBCO BusinessEvents Application in step 2, do the following:

   — Select the **Main** tab and configure values as explained in Launch Configurations Reference on page 670.

   — Select the **Classpath** tab and configure the classpath for external libraries or custom functions as needed, for example if the project uses Rendezvous or JMS channels. See Working with External Library and Custom Function Paths on page 8 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.

   On Linux platforms when TIBCO BusinessEvents DataGrid is used, you must set `LD_LIBRARY_PATH` to *AS_HOME*`/lib` in the Environment tab.

   — For information on standard Eclipse features incorporated into this area, see Eclipse help.

5. If you 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) on page 665).

6. Click **Apply** to save configuration settings.

7. Do one of the following:

   — 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 launch the debugger, or click **Run** to save the configuration and run the engine.

   You can then assert test data as desired in order to observe the effect of the data on the engine. See Asserting Rule Input Data on page 680.

Launch Configurations Reference

See Adding and Working with Launch (Debug or Run) Configurations on page 667 for the related procedure.

The Source and Common tabs are standard Eclipse dialogs. See Eclipse help for details on use of those tabs. If the project uses third party JARs, 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 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 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.*
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 drop-down 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. 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 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

Creating and Working With 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.

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.

To Create Concept and Event Instance Test Data

1. In BusinessEvents Studio Explorer (or in the Debug perspective BusinessEvents Studio Explorer view), 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.

2. 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**.

3. 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 on page 722).

To Edit Test Data for Concepts and Events in BusinessEvents Studio Explorer

To add more test data or edit test data you created earlier, do the following.

1. In BusinessEvents Studio Explorer, expand the `TestData` folder in the root of the project.

   The `/TestData` folder is the default location (see Tester Preferences on page 722).

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. Add, remove, and edit rows of test data as desired.

   Test data filenames use the format *eventName*`.eventtestdata` and *conceptName*`.concepttestdata`.

To Edit Concept and Instance Test Data in the Rule Input View

1. Open the Rule Input view, if it is not already shown. To make the Rule Input View visible, do one of the following:
   - 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

1. Open the Rule Input view, if it is not already shown. To make the Rule Input View visible, do one of the following, 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 Mapper on page 256.

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. Do one of the following:

   — 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 on all breakpoint functionality. You can set or change breakpoints during a debug session also.

You cannot step through custom Java code.

To Set Breakpoints in Rules and Rule Functions

1. In TIBCO BusinessEvents Studio, open the source editor for a rule or rule function.

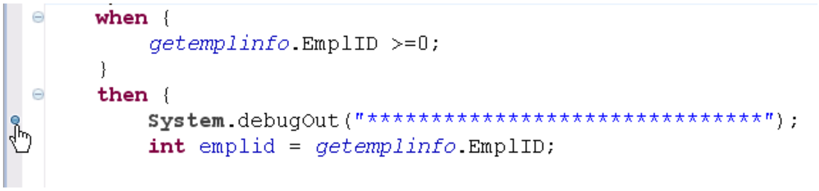
   You can work with breakpoints in the debugger perspective as well as in the TIBCO BusinessEvents Studio development perspective.

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. Do one of the following:

   — 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 break point, right click, and do any of the following:

   — To disable a breakpoint, select Disable.

   — To remove a breakpoint, select Toggle Breakpoint. (Or just 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.

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 setup as needed. You can create test data and set breakdpoints before you run debugger, or using views within the Debug perspective. Other setup must be done before you begin. See the following sections for details:

- Preparing to Run (Test) or Debug a Project, page 665
- Adding and Working with Launch (Debug or Run) Configurations, page 667
- Creating and Working With Test Data, page 672
- Setting Breakpoints in Rules and Rule Functions, page 675

Then run debugger as explained below.

To Run Debugger

1. As needed, switch to Debug perspective. Select **Window > Open Perspective**, or click the Open Perspective ( ) button). Then select **Other > Debug**.

   Alternatively, wait till TIBCO BusinessEvents prompts you to change to debug perspective. This happens when the debugger reaches the first breakpoint.

2. Click the down-arrow to the right of the debugger (▾) button. You see a drop-down list. Do one of the following:

   — Select a debug configuration from the list.

   To add configurations to the drop-down 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 debugger with a configuration.)

   Debugger starts a TIBCO BusinessEvents engine, using parameters provided in the launch configuration, if any were provided.

3. Assert data as explained in Asserting Rule Input Data.

4. View the results as explained in Viewing the Results on page 682.

5. 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.

6. 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. 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, page 665
- Adding and Working with Launch (Debug or Run) Configurations, page 667
- Creating and Working With Test Data, page 672

Then run tester as explained below.

To Run Tester

1. As needed, switch to Debug perspective. Select **Window > Open Perspective**, or click the Open Perspective (⊞) button). Then select **Other > Debug**.
2. 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.
3. At the Run Configurations dialog select a run configuration and click **Run**.

   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.
4. Assert data as explained in Asserting Rule Input Data.
5. View the results as explained in Viewing the Results on page 682.
6. 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.

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 Creating and Working With Test Data on page 672 for details about creating and saving data.

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.

To Assert Tester Data

1. Select the Rule Input tab and then the Tester Data bottom 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. Check 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 on page 672 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 on page 682 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

1. 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**.
2. Do one of the following:
   - — Provide input from the mapper as explained in Using the Function Argument Mapper on page 256. 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.
3. Specify the **Launch Target, Entity URI, Destination URI** and **Rule Session**.
4. Click **Assert**.

See Viewing the Results on page 682 for details about understanding the results of a test or debugger run.

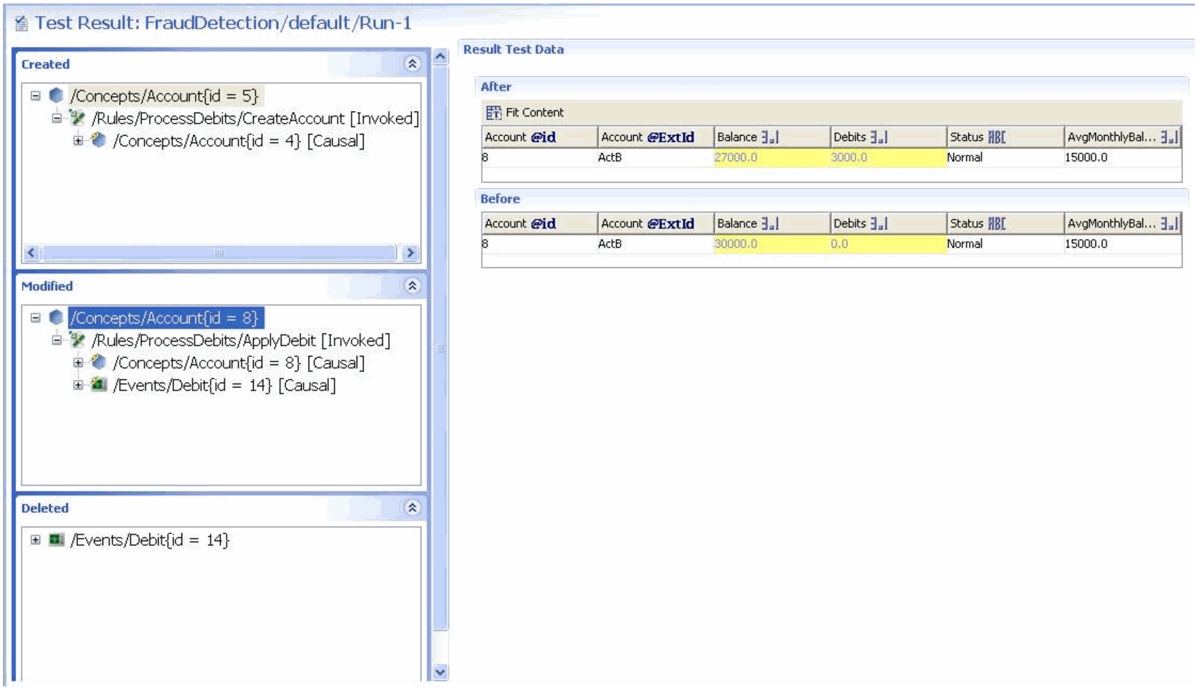
Viewing the Results

After data is asserted to the working memory (see Running Tester on page 679), a run to completion (RTC) cycle occurs and you see the following:

- The Console tab displays engine console messages.
- The result data editor appears showing 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 on page 722 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.



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

Fit Content

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

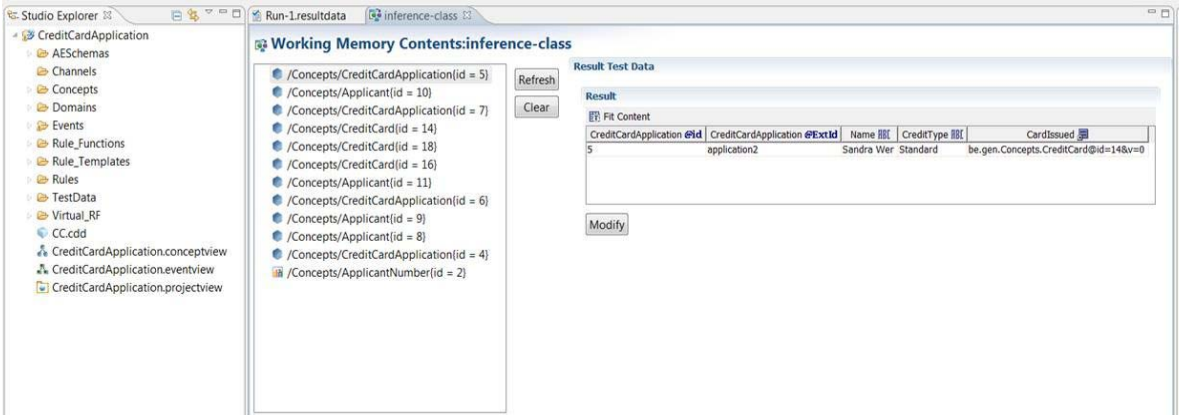
The Result Data view shows the results of a single RTC cycle. On the left of the view you see the **Created**, **Modified**, and **Deleted** entities (if any).

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.

Viewing and Understanding Working Memory Contents

The Working Memory concepts or scorecards can be manipulated in Working Memory View. Change the property values and click modify.



Working Memory Contents:inference-class

/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

To see the Working Memory Contents

1. Click the Working Memory Contents icon on the tool bar, or from the **Project** menu.

2. Select the appropriate rule session (agent).

   The number of objects in the working memory contents is specified in the Preferences. For more details, see Tester Preferences on page 722.

Chapter 41 Diagrams

This chapter covers TIBCO BusinessEvents diagrams, including a summary of the diagrams available, how to access and work them, and what options and preferences are available for working with them.

Topics

Overview of Diagrams

TIBCO BusinessEvents provides several kinds of diagrams, which are visualization tools that help you to understand and analyze even very large and complex projects.

The diagrams in TIBCO BusinessEvents are based on Unified Modeling Language (UML), but are NOT completely compliant to UML.

The diagrams allow you to 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 runtime. 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.

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.

Add-on Products State model diagrams are available in the TIBCO BusinessEvents Data Modeling add-on. Metrics are available in the TIBCO BusinessEvents Views add-on. For metrics, you can view a dependency diagram only by right-clicking in Project View. See the add-on product documentation for more details

.

Table 66 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			
Metric	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

See Diagram Preferences on page 717 to understand and set preferences so that diagrams display as you desire.

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

Performing 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

Using 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:

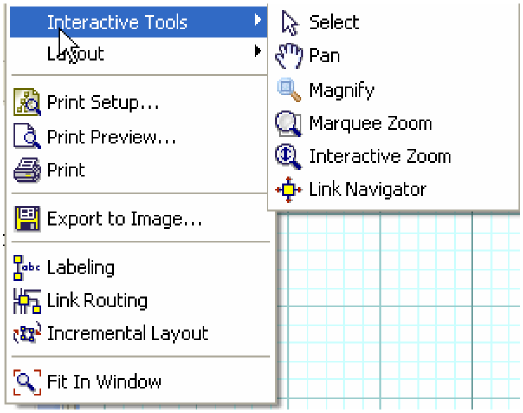
- Refresh Diagram
- Zoom Percentage
- Search Diagram Entities

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 on page 705.

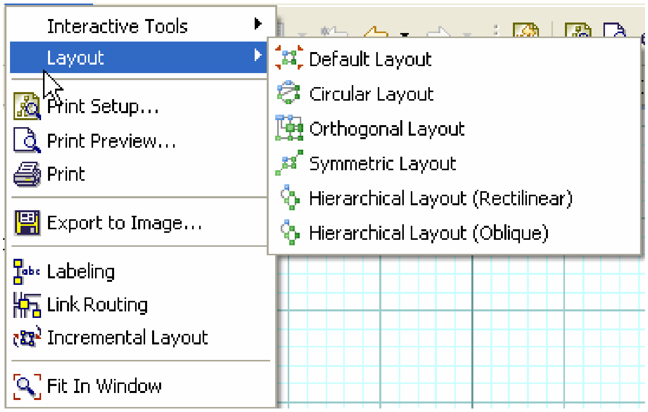
Interactive Tools

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 on page 705.



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 on page 705.



Context Menu Diagram Tools

Diagram tools are available as context menus for diagram canvas and objects. Table 67, Context Menu Options for Canvas and Objects lists these context menu options.

Table 67 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

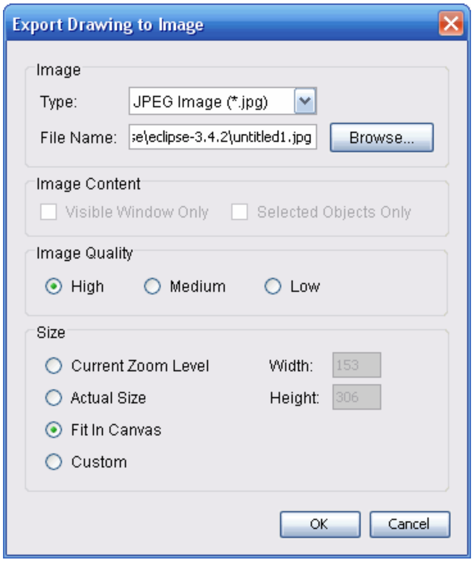
For more details about the diagram tools, see Diagram Options and Tools Reference on page 705.

Exporting a Diagram to an Image

The Export to Image option enables you to save the diagrams locally for later use.

To Export a Drawing (Diagram) to Image

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: se\eclipse-3.4.2\untitled1.jpg

Image Content

□ Visible Window Only □ Selected Objects Only

Image Quality

● High ○ Medium ○ Low

Size

○ Current Zoom Level Width: 153

○ Actual Size Height: 306

● Fit In Canvas

○ Custom

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. Check the check boxes Visible Window Only and Selected Objects Only as appropriate to specify the content of the image.

Selected Objects Only gets enabled only if you select an object in the diagram.

5. Specify the Image Quality as appropriate.
6. Specify the size, and click OK.

The diagram is exported to an image, and saved on the system.

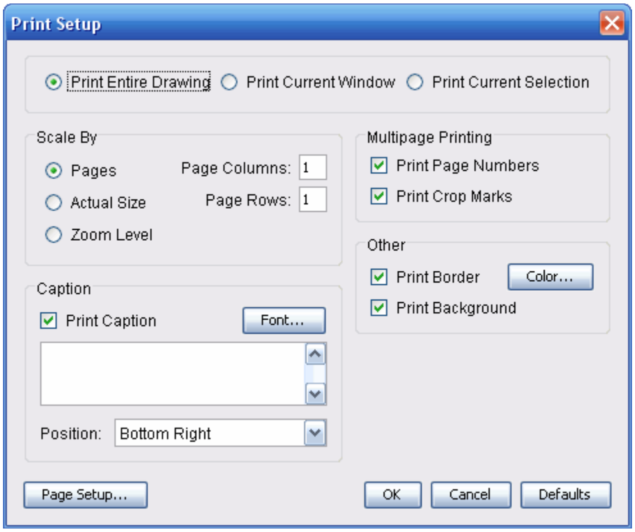
Printing a Diagram

You can set up preferences, preview the print job, and then print the diagram.

To Set up Printing Preferences

The Print Setup dialog enables you to specify the print setup for a diagram.

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

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**.

To Preview and Print a Diagram

1. From the top menu, select **Diagram > Print Preview**. An image of the diagram to be printed displays.
2. From the top menu select **Diagram > Print**.

Setting Diagram Preferences

You can set the preferences for all types of diagrams.

To Set Diagram Preferences

For more information on Preferences, see Chapter 42, TIBCO BusinessEvents Studio Preferences, on page 709.

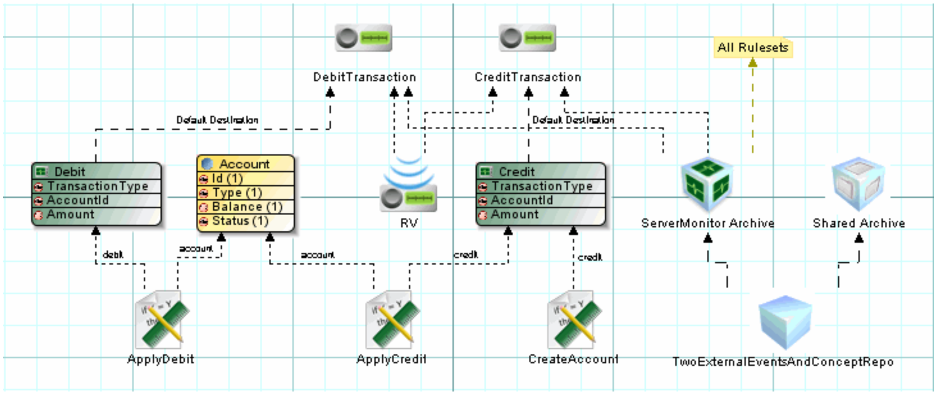
1. From the top menu, select **Window** > **Preferences**.
2. In the Preferences dialog, expand **TIBCO BusinessEvents**, and then expand **Diagram**.
3. Select any type of diagram.

   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, score cards, 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 to have 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

Working with 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.

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**. Check 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, page 696.

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.

To Create a Selected Entity Project Diagram

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**.

When you select a folder all elements in that folder are selected.

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. You can change the depth of dependencies shown using preferences.

To Filter a Selected Entity Project Diagram

To filter what you see, check and uncheck the options in the Project Filter list within the Palette view, and click **Apply**.

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.

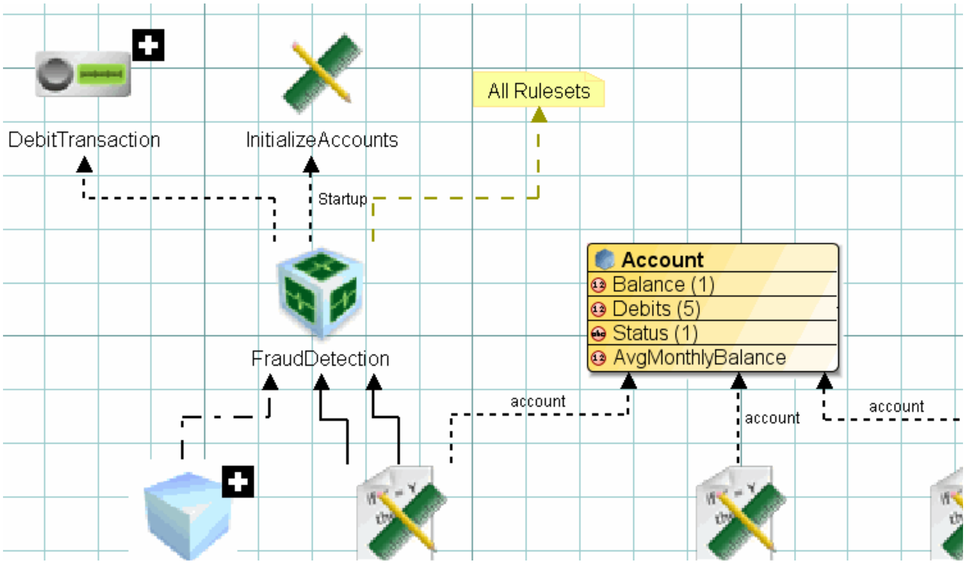
Group Options Group Concepts, Group Events, Group Rules, Group Rule Functions

The filter options in the Palette view are loaded from diagram preferences. See Diagram Preferences on page 717 for more details.

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.



Rule actions are not represented in rule dependency diagrams. Use Debugger to analyze rule actions. See Chapter 40, Testing and Debugging Projects, on page 661.

To Create a Dependency Diagram

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 (showing only direct dependents). White crosses indicate hidden dependencies. The default appearance is also driven by the diagram preferences. See Diagram Preferences on page 717 for more details.

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:

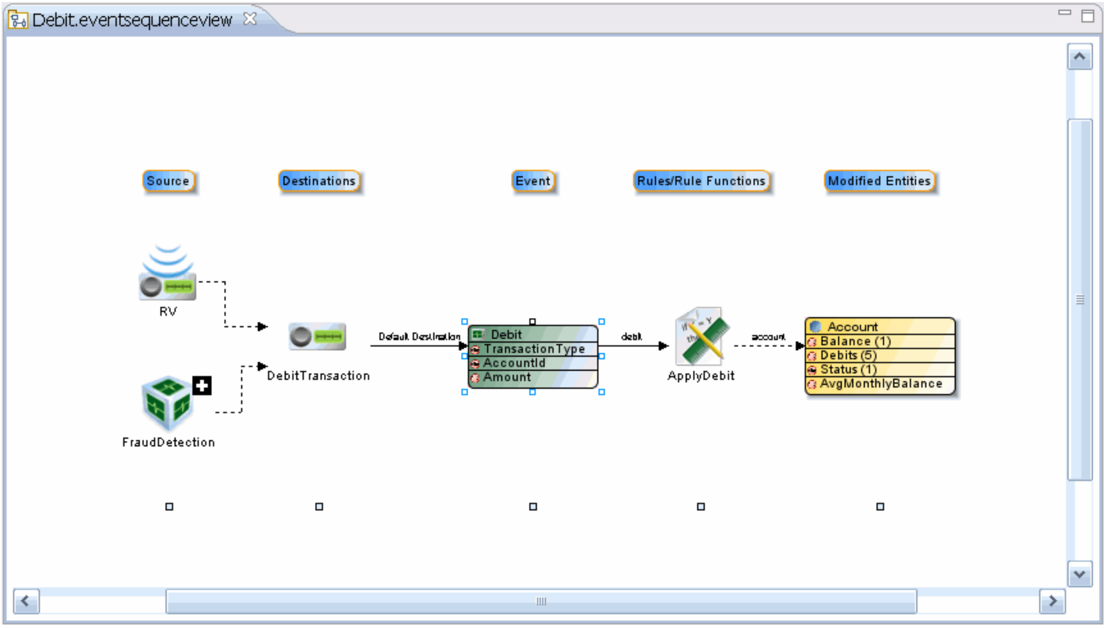
   — One: Shows the direct dependencies of the selected item

   — 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.



To 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.

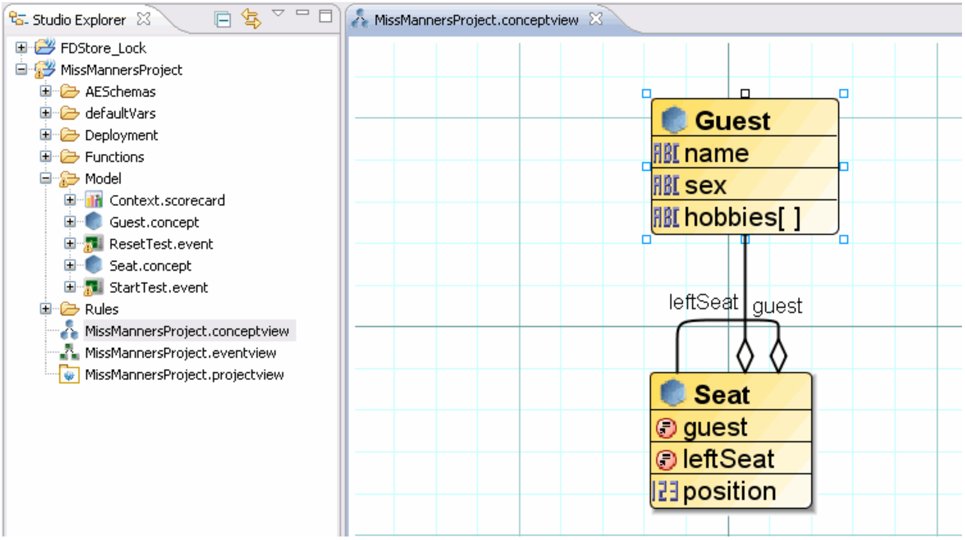
The sequence diagram for the item appears in the editor area.

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).



To Create a Concept Model Diagram

Do one of the following:

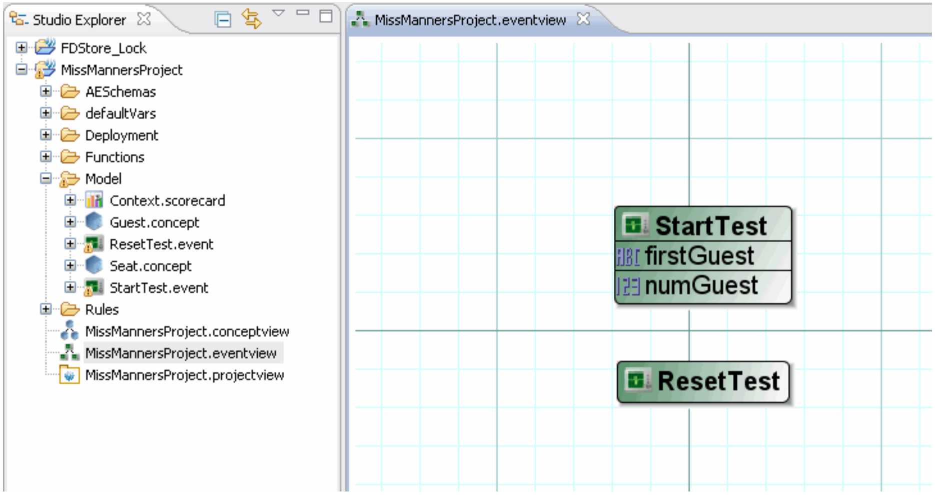
- Right-click a resource or a project in BusinessEvents Studio Explorer and select **Create Concept Model Diagram**
- Open a concept editor and click the Concept Model Diagram ( ) button in the top right of the editor
- Double-click the `*.conceptview` file in the Studio Explorer.

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 disk).



To 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.

Table 68 TIBCO BusinessEvents Diagram Tools Reference

Button	Description
Interactive Tools Menu Options and Toolbar Buttons	
	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 on page 707.
	Orthogonal Layout It is available in the layout list. See Orthogonal Layout on page 707
	Symmetric Layout It is available in the layout list. See Symmetric Layout on page 707.
	Hierarchical Layout (Rectilinear) It is available in the layout list. See Hierarchical Layout on page 708.

Table 68 TIBCO BusinessEvents Diagram Tools Reference

Button	Description
	Hierarchical Layout (Oblique) It is available in the layout list. See Hierarchical Layout on page 708.
Other Diagram Menu Options and Toolbar Buttons	
	Print Setup. Sets options for printing the diagram or elements of the diagram. See Printing a Diagram on page 693.
	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 on page 692.
	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. See Diagram Preferences on page 717 for details.
	Incremental Layout See Incremental Layout on page 708.
	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.

Additional Layout Options

Various options affect the appearance of a diagram, no matter what layout is chosen. (The hierarchical layout offers additional options.) See Diagram Preferences on page 717 for more details.

The main layout options are:

- Circular Layout, page 707
- Orthogonal Layout, page 707
- Symmetric Layout, page 707
- Hierarchical Layout, page 708 (default)
- Incremental Layout, page 708

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 Configuring Diagram Preferences and Properties on page 688 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 Configuring Diagram Preferences and Properties on page 688 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.

See Diagram Preferences on page 717 for more details.

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

Chapter 42 TIBCO BusinessEvents Studio Preferences

This chapter shows the preferences available in TIBCO BusinessEvents Studio.

Topics

Setting Preferences

To access preferences select **Window > Preferences** and expand **TIBCO BusinessEvents**. Also see Setting Diagram Preferences on page 695 for a more detailed procedure.

To Set Preferences:

1. From the Window menu, select **Preferences > TIBCO BusinessEvents**.
2. Expand the options on the left and select each option to view a panel of options on the right.

   Details on 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.

Preference Sections

For tester and tester appearance preferences, see Tester Preferences on page 722.

TIBCO BusinessEvents (Top Level)

The top level preferences allow you 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.

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.

For Decision Table preferences see Decision Table Related Preferences on page 712 for details.

Decision tables are available only when the TIBCO BusinessEvents Decision Manager addon is in use..

Diagram Preferences

In addition to a global preferences section, there are sections for each type of diagram. See Diagram Preferences on page 717 for details.

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 checked (enabled). 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.

Process Preferences

Process preferences are used with the TIBCO BusinessEvents Process Orchestration add-on.

Rules and Rule Debug Preferences

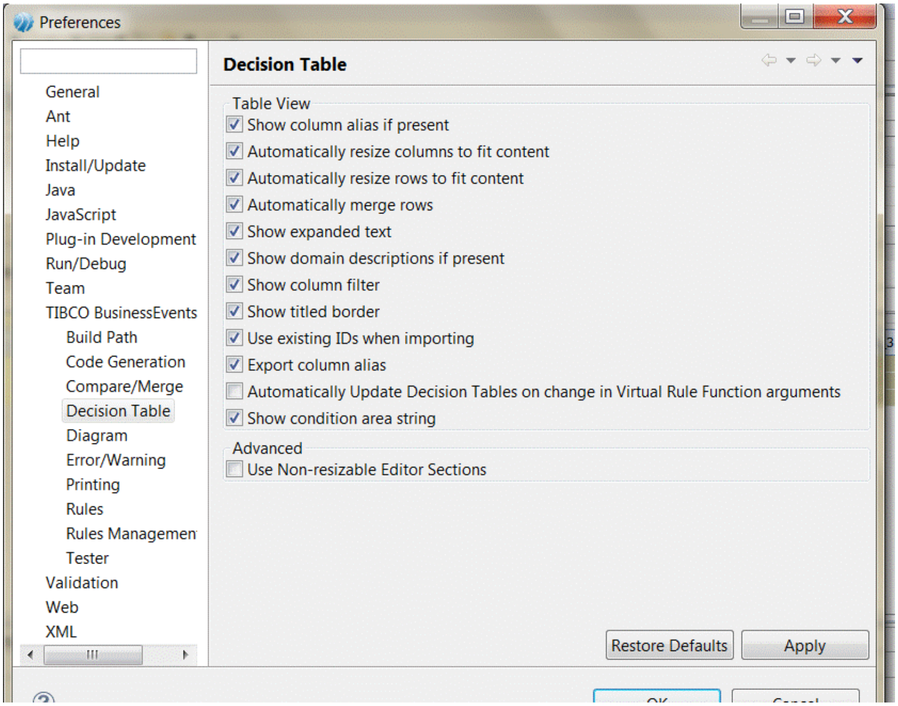
In addition to the standard Eclipse Run/Debug preferences, TIBCO BusinessEvents requires some rule debug preferences.

Rules Management Server Preferences

These preferences are used with TIBCO BusinessEvents Decision Manager.

Decision Table Related Preferences

Decision Table Preferences



Decision Table

Table View

- ☑ Show column alias if present
- ☑ Automatically resize columns to fit content
- ☑ Automatically resize rows to fit content
- ☑ Automatically merge rows
- ☑ Show expanded text
- ☑ 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

Advanced

- □ Use Non-resizable Editor Sections

The following 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

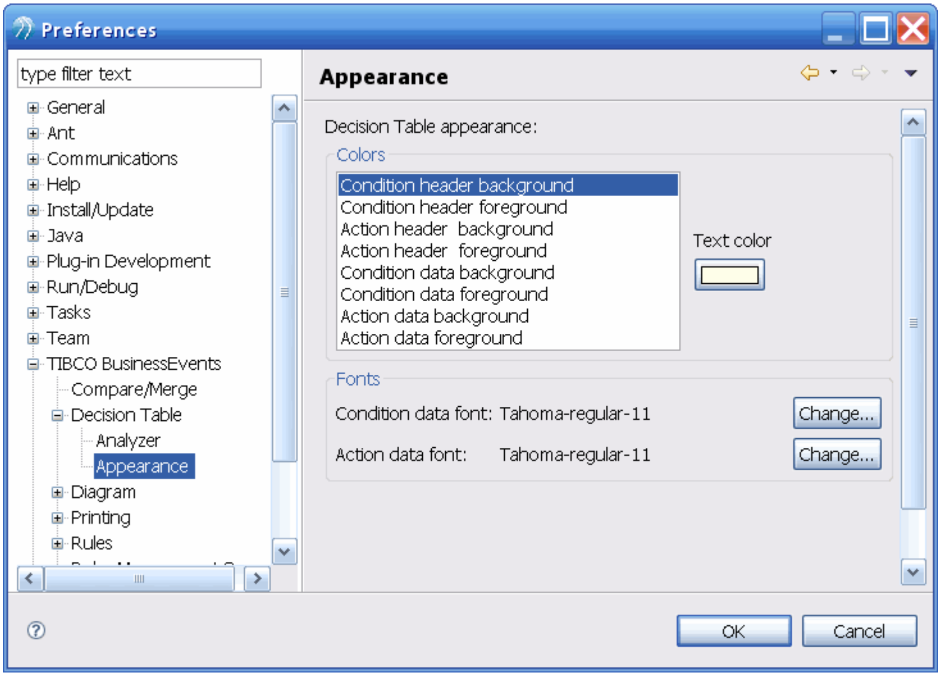
- Show titled border
- Use existing IDs when importing
- Export column alias
- Automatically update decision tables on change in virtual rule function arguments
- Show condition area string
- The Use non-resizable editor sections option prevents or allows users from resizing editor sections.

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



Appearance

Decision Table appearance:

Colors

- Condition header background
- Condition header foreground
- Action header background
- Action header foreground
- Condition data background
- Condition data foreground
- Action data background
- Action data foreground

Text color

Fonts

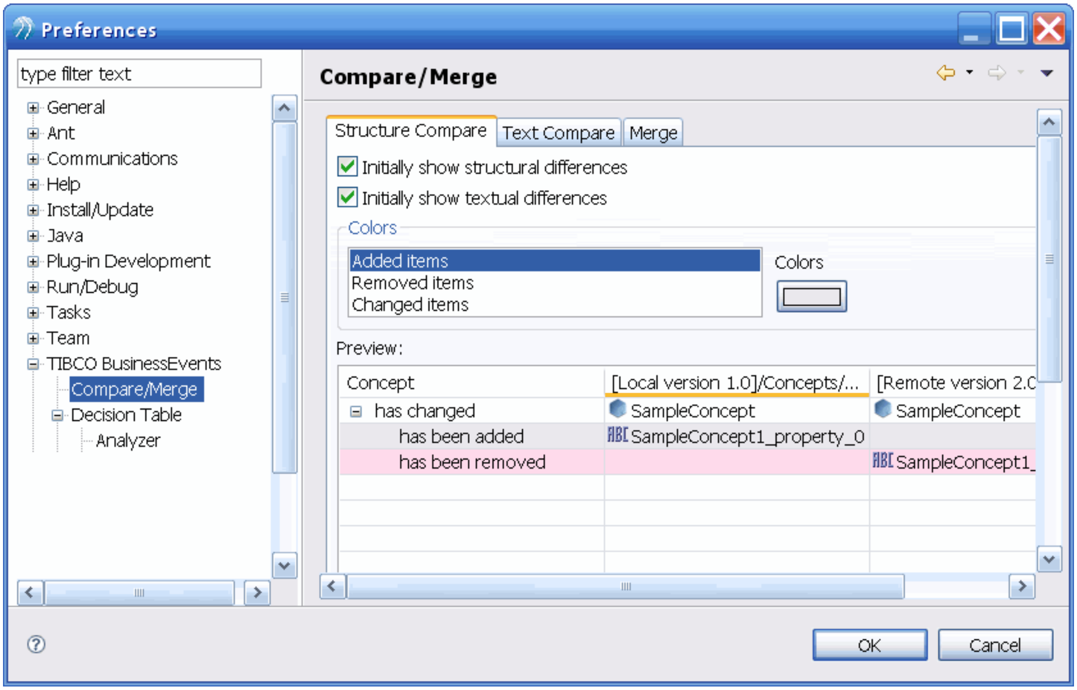
Condition data font: Tahoma-regular-11 Change...

Action data font: Tahoma-regular-11 Change...

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

This tabbed dialog is in the TIBCO BusinessEvents preferences. It applies to decision tables.



Compare/Merge

Structure Compare

☑ Initially show structural differences

☑ Initially show textual differences

Colors

- Added items
- Removed items
- Changed items

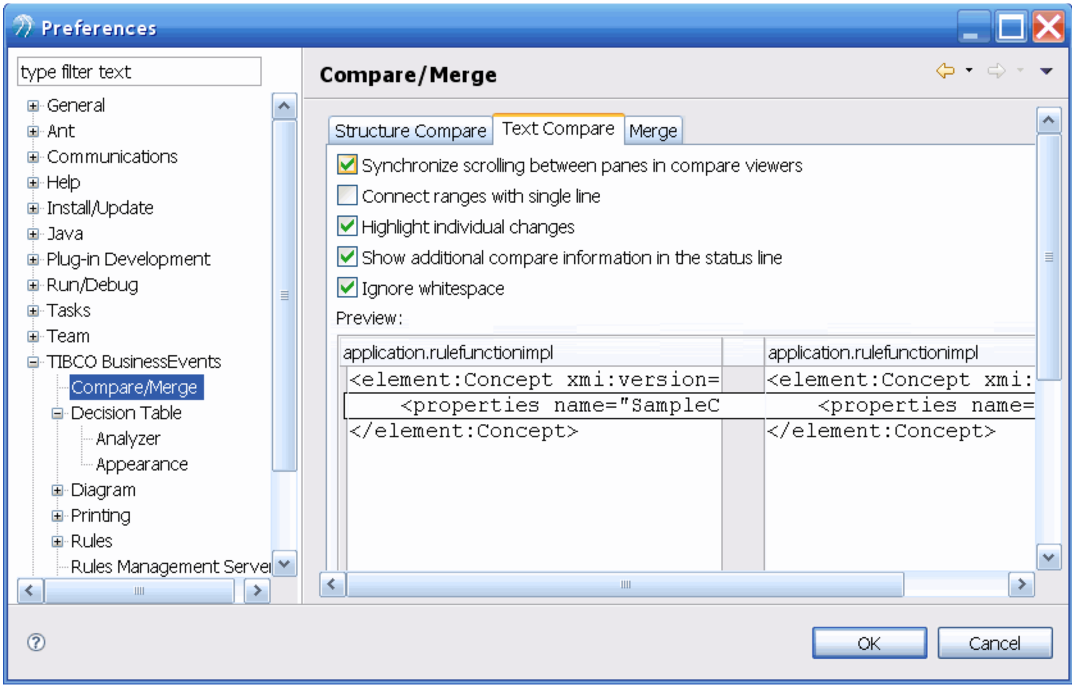
Colors

Preview:

Concept	[Local version 1.0]/Concepts/...	[Remote version 2.0
⊟ has changed	SampleConcept	SampleConcept
has been added	SampleConcept1_property_0	
has been removed		SampleConcept1_

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.



☑ Synchronize scrolling between panes in compare viewers
□ Connect ranges with single line
☑ Highlight individual changes
☑ Show additional compare information in the status line
☑ Ignore whitespace

Preview:

```
application.rulefunctionimpl              application.rulefunctionimpl
<element:Concept xmi:version=             <element:Concept xmi:
    <properties name="SampleC                 <properties name=
</element:Concept>                        </element:Concept>
```

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 in the TIBCO BusinessEvents Data Modeling add-on.)

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.

Table 69 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.

Table 69 Global Diagram Preferences (Cont'd)

Option	Description
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. 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 Table 70 on page 719. The preferences are organized alphabetically and not grouped according to the diagram types.

Table 70 Specific Diagram Preferences (Alphabetical) (Sheet 1 of 3)

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.

Table 70 Specific Diagram Preferences (Alphabetical) (Sheet 2 of 3)

Option	Description
Layout Quality	Options are Draft (lowest quality), Medium, and Proof (highest quality). It 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. Options are: Top to Bottom, Bottom to Top, Left to Right, and Right to Left. Default: Top to Bottom.

Table 70 Specific Diagram Preferences (Alphabetical) (Sheet 3 of 3)

Option	Description
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.
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 Chapter 40, Testing and Debugging Projects, on page 661.

Table 71 Reference to Tester Preferences

Option	Description
Input Directory	The directory in which test data files you create are stored. Default: `/TestData`
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, 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**.

Table 72 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

Table 72 Reference to Tester Appearance Preferences

Option	Description (Cont'd)
Font for the modified value	The font in which the modified value appears. Click **Change...** to change the font. Default: Tahoma Regular 11

Appendix A Understanding Entity Caches

This appendix is provided for those who want to understand the internal structures of the Oracle Coherence caches used in Cache object management. This information is not required for configuration tasks.

For each entity in working memory, a corresponding cache exists in the cache cluster. Internal entities also have caches for various purposes, explained in this section.

Entity Cache Names Format

Each entity cache has a name, which uses the following format:

cache-type.cluster-name.AgentClassName.entity-name

The elements of the above name are explained below

Cache Type (Caching Scheme)

Cache type is the type of caching scheme (as defined by its cache name in the `coherence-cache-config.xml` descriptor), for example, `dist-unlimited-bs`.

Cluster Name

Cluster name is the value of the following property:

`java.property.tangosol.coherence.cluster`

Agent Name

This field of the cache name is blank because TIBCO BusinessEvents does not support agent-specific entity caches.

All entities are globally scoped and available to all agents.

Entity Name

Two types of entities have caches:

- Internal entities
- Ontology entities

Internal entity names and caches are listed and described in Table 73, Internal Entity Caches, on page 726.

The ontology entity field of the entity cache name uses the entity's generated class name, which is similar to its design-time folder path and name, prefixed by `be.gen`. For example:

```
be.gen.Concepts.LargeConcepts.ThisLargeConcept
```

Caches for Ontology Objects

These caches are used to store the objects of types defined in the ontology of the project.

The types of caches created for ontology objects depend on the caching scheme used. If the `dist-unlimited-bs` caching scheme is used, then the cache names look like this:

```
dist-unlimited-bs$foo$$be.gen.Order
```

Where foo is the cluster name.

Caches for Internal Entities

The following internal caches use a pre-defined scheme in the cache configuration file. Do not change this scheme. This information is provided for reference only.

Table 73 Internal Entity Caches

Entity (Cache) name	Purpose of the Cache
`Master`	Maintains the cluster state and is shared by all nodes.
`Catalog`	Maintains a cached copy of all ontology definitions shared by all nodes.
`TypeIDs`	Stores the mapping between type IDs and class names. All ontology objects are tagged with a unique integer ID. Use of IDs avoids the need to serialize and send class name strings between nodes.
`ObjectTableIDs`	Stores the key mapping for all objects in the cluster. The objects themselves are stored in their respective caches.
`ObjectTableExtIDs`	Stores the external key mapping for all objects that have an external ID (`extId`).
`AgentTable`	Stores all the agents and their respective states across all cluster nodes and identifies the currently active and standby nodes.

Table 73 Internal Entity Caches

Entity (Cache) name	Purpose of the Cache
`AgentTxn-`*agentId*	Each agent in the cluster has an `AgentTxn-`*agentId* cache. The *agentId* is internally generated. It stores the change list for the agent. The change list is used to replicate changes between active-active and active-passive sets of agents in the cluster so that they stay synchronized.
`TimeQueue`	Maintains all entries that are time bound, for example, state machines that can have timeouts at a state machine level or at a state level. This cache maintains an index to all objects that must be re-evaluated after a certain period of time.

Appendix B Handling Null Properties

This appendix explains three cases for special handling of null concept property values.

Topics

Handling Null Concept Property Values

This appendix explains three cases for special handling of null concept property values:

- Enabling Use of the Nillable Attribute on page 730
- Enabling Null Property Values to Appear When Serializing Concepts to XML on page 730
- Enabling and Setting Special Treatment of Numeric and Boolean Null Values on page 732

This section also provides related procedures and a reference to the properties used (Table 74, Properties for Null Property Handling, on page 734).

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 (`""`).

Effect on schema generation tool The setting for this property affects the concept XSD files generated using the Generate Schema utility. See Exporting (Generating) Concept and Event Schema (XSD) Files on page 5 for details.

Enabling Null Property Values to Appear When Serializing Concepts to XML

By default concept properties with null values are excluded when concept objects (instances) are serialized to XML. 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
```

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/>
```

Enabling and Setting 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 on page 730), 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 Table 74, Properties for Null Property Handling, on page 734).

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 on page 733.

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 datatypes in TIBCO BusinessEvents (shown in Property Reference for Null Property Handling on page 734).

To override the default values do the following:

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.

Table 74 Properties for Null Property Handling

Property	Notes
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 on page 730 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. When set to false, null-valued concept properties are output to XML. See Enabling Null Property Values to Appear When Serializing Concepts to XML on page 730. 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. See Enabling and Setting Special Treatment of Numeric and Boolean Null Values on page 732. Possible values are true and false. Default is false.

Table 74 Properties for Null Property Handling (Cont'd)

Property	Notes
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).