

TIBCO BusinessEvents® Developer's Guide

*Software Release 5.4
January 2017*

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, The Power of Now, TIBCO ActiveMatrix, TIBCO ActiveMatrix BusinessWorks, TIBCO Administrator, TIBCO ActiveSpaces, TIBCO BusinessEvents, TIBCO Designer, TIBCO Enterprise Message Service, TIBCO Enterprise Administrator, TIBCO Hawk, TIBCO Live Datamart, TIBCO LiveView Web, TIBCO Runtime Agent, TIBCO Rendezvous, TIBCO StreamBase, and Two-Second Advantage are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

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 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.

Copyright © 2004-2017 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Figures	16
TIBCO Documentation and Support Services	17
Project Development	18
Creating a Project	18
Importing Projects in TIBCO BusinessEvents Studio	19
Finding a Project Element	20
Exporting (Generating) Concept and Event Schema (XSD) Files	20
Generating an XML Schema (XSD File)	20
Validating a Project or Project Resource	21
Working with External Library and Custom Function Paths	21
Adding an External Library or Custom Function Path	22
Specifying an External Library Location Using Build Path Variable	23
Working with Project Libraries	23
Creating (Exporting) a Project Library	24
Adding (and Removing) Project Libraries in a Project	25
Including a Project Library (Adding it to the Build Path)	25
Importing a Project Library	26
Removing a Project Library	26
Changing XPath Version of The Project	27
Working with Global Variables	27
Setting and Overriding Global Variables from a Project Library	28
Adding and Managing Global Variables	28
Add and Manage Global Variables	29
Using Global Variables	30
Global Variable Reference	30
Overriding Global Variables at Deploy Time	31
Order of Precedence of Global Variable Overrides	32
Storing Trusted Certificates Outside of Your Project	32
Store Trusted Certificates Outside of the Project	32
Remote Terminal	33
TIBCO BusinessEvents Studio Tools Utility	34
Building an EAR File at the Command Line	34
Importing an Older Studio Project at the Command Line	35
Import an Existing 4.x Project at the Command Line	35
TIBCO BusinessEvents Studio Tools Options	36
Open the Imported Project in TIBCO BusinessEvents Studio	36

Working With Project Libraries at the Command Line	37
Work with Project Libraries at the Command Line	37
Examples of Creating a Project Library	37
Migrating Core Coherence Functions at the Command Line	39
Migrate Core Coherence Functions at the Command Line	39
Migrating Mapper Functions to XPath 2.0 Using Command Line	39
Generating All Project Class Files at the Command Line	40
Generate Class Files at the Command Line	40
TIBCO BusinessEvents Studio Tools for Generating Class Files	40
Generating Encrypted Passwords	41
Running the Tester on the Command Line	42
The Generate Command	43
The Import Command	43
The Assert Command	44
Element Refactoring Operations	46
Renaming Moving Deleting and Copy-Pasting Elements	46
Renaming Moving and Deleting Elements	46
Renaming a Project Element	47
Moving an Element to a Different Project Folder	47
Delete an Element or Folder	48
Deleting a Project	48
Working with the Preview Page	49
Copy-Pasting an Element	50
Migrating Core Coherence Functions	50
Migrate Coherence Function Calls in TIBCO BusinessEvents Studio	50
Mapping of Coherence Functions to DataGrid Functions	51
Automatic Refactoring Actions and Limitations	52
Refactoring Limitations	52
Refactoring for Move and Rename Operations	52
Refactoring for Delete Operations	53
Channels and Destinations	55
Channel Serializers	55
Mapping Incoming Messages to Non-default Events	56
Working with Rendezvous Channels	56
Deserializing from Rendezvous Message to Event	56
Serializing from Event to Rendezvous Message	57
Avoiding REGISTRATION COLLISION RVCM Advisory Messages	57
Setting UTF-8 Message Encoding for the Rendezvous Channel	57
Working with Local Channels	58

Adding a Channel	58
Editing a Channel	59
Adding a Destination to a Channel	59
Communicating with Other Sources using TCP	60
Example TCP Rule Function to Start a Local TCP Server	61
Example TCP Rule Function to Connect to a Remote TCP Server	61
APIs for TCP Communication	61
Channel Resource Reference	62
Destination Resource Reference	67
JMS Channels	73
Selecting a JMS Serializer	73
Payload Handling	73
Creating Unique JMS DurableSubscriber Name Properties	74
Variables to Use with DurableSubscriberName	75
JMS Message Acknowledgement Modes	75
Using CLIENT_ACKNOWLEDGE Mode with Websphere MQ and Cache-Aside	76
When JMS Messages are Acknowledged	77
Using JMS Header Properties in Incoming and Outgoing Messages	77
Setting Certain Header Properties in Destinations	77
Setting Header Properties Using Header Properties from Incoming JMS Messages	77
Setting JMS Header Properties in Outgoing JMS Messages Using Event Properties	77
How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages	78
JMS Header Field Names	78
JMS Sender Session Pooling	80
SOAP over JMS Support in WSDL Import and Export	80
HTTP Channels	83
SOAP Channels	83
Adding an HTTP Connection	83
Adding an Identity Resource — For SSL Only	84
Adding an HTTP Channel	84
Adding a Destination	84
Creating Events as Needed and Setting Default Destinations	84
Configuring Rules and Rule Functions	85
In the CDD Configure the Processing Unit	85
Working with HTTP Requests	85
HTTP Requests and Events Mapping with the REST Serializer	86
JSON Payload	87
Action Rule Function Based Approach	87
Sample Code for Action Rule Function	88

Setting Up Fault Tolerance for the HTTP Channel	89
HTTP Channel Advanced Configuration Settings	90
Defining Event Properties for Standard HTTP Header Properties	95
Configuring the Retry Count for HTTP Client Connection	96
HTTP Functions for HTTP Request Messages Configuration	96
Generating a Self-Signed SSL Certificate (Keystore)	96
Getting POST Data	97
Loading Trusted Certificates	97
Sending an Event	98
Sending an Asynchronous Request (Not Secure)	99
Sending a Secure Asynchronous Request	100
Sending a Secure Synchronous Request	101
Sending a Request via Proxy Server	102
Disabling HTTP Methods at Channel and Destination Level	103
Configuring TIBCO BusinessEvents as a SOAP Server and Client	104
Overview of SOAP Related Resources	104
Manually Creating Resources to Work with SOAP Services	105
Creating Resources Using the WSDL Import Utility	106
Importing a WSDL File	106
Re-importing a WSDL File	107
Exporting Resources as a WSDL File	107
Exporting a Rule or Rule Function as a WSDL	108
Parsing and Building SOAP Messages	109
Working with Incoming SOAP Messages (Event Payloads)	109
Parsing (and Optionally Removing) Headers and Header Attributes	109
Parsing the SOAP Body (SOAPBodyParts)	110
Parsing Attachments	110
Parsing SOAP Fault XML Nodes	111
Working with Outgoing SOAP Messages (Event Payloads)	111
Understanding the WSDL to Project Resource Mapping	111
Example WSDL	112
Example Project Folder Structure	112
How Project Artifacts are Named	114
Hawk Channels	116
Working with Hawk Channels	116
Setting Up the Runtime Configuration	117
Using the Hawk Destination and Event Wizard	117
Creating a Destination and Event Using the Wizard	117
Creating Event for Hawk Catalog Function Using Wizard	118

StreamBase Channel	120
StreamBase Key Terms	120
Creating a Destination and SimpleEvent from a StreamBase Schema	120
Message Processing	122
Custom Channel	124
Creating a New Custom BusinessEvents Channel	124
Key Java API Components for Custom Channel	126
ActiveSpaces Channels	128
Metaspace	128
Space	128
Members and Member Roles	128
Seeders	129
Leeches	129
ActiveSpacesSerializer	129
Distribution role	129
Consumption mode	129
Tuple	130
Filters	130
Operators Supported in Filters	130
Formats for Filter Values	131
Working with ActiveSpaces Channels	132
Creating a Destination and Event for ActiveSpaces Channel Using the Wizard	132
Configuring the Destination and Default Event Manually	136
Assigning a Default Event to the Destination	136
Configuring Events to Work with ActiveSpaces Channel	137
Configuring Security and Authentication For ActiveSpaces Channel	137
Events	138
Overview of Simple Events	138
Using Inheritance	138
Working with Events in Rules	138
Adding a Simple Event	139
Defining Payloads Using XML Schema Files	139
Defining an XML Schema File in TIBCO BusinessEvents Studio	139
Importing an XML Schema File	140
Using an XML Element in a Schema File in an Event Payload	140
Simple Event Reference	140
Wizard and Configuration (Standard Tab)	141
Properties (Standard Tab)	142
Declaration and Expiry Action (Advanced Tab)	142

Payload (Advanced Tab)	142
Simple Event Attributes Reference	144
Working With Time Events	145
Scheduled Time Events	145
Rule Based Time Events	145
Adding a Time Event	146
Configuring a Rule-Based Time Event in a Rule or Rule Function	146
TimeEvent Resource Reference	147
Wizard and Configuration Tab	147
TimeEvent Attributes Reference	148
Rule-Based TimeEvent Function Reference	149
Using Scheduler Functions (Requires Cache OM)	149
Creating a Scheduler	150
Working with Advisory Events	151
Advisory Event Attributes Reference	151
Event Property	152
Concepts	153
Adding a Concept	153
Adding Concept Relationships	153
Creating a Containment Relationship	153
Creating a Reference Relationship	154
Enabling the Concept Reference Serialization in Concept to JSON	154
Editing or Deleting Concept Relationships	154
Concept Resource Reference	154
Wizard and Configuration Tab	155
Properties	156
Metadata	157
Concept Attributes Reference	157
Adding a Scorecard	158
Using a Scorecard in Rules	158
Scorecard Resource Reference	159
Accessing a Concept PropertyAtom	159
Get and Set PropertyAtom Value With User-Specified Time	159
Concept Property Arrays	160
Rules and Functions	162
Adding a Rule	163
Rule Editor Reference	164
Adding a Rule Function	165
Adding a Rule Function	165

Rule Function Resource Reference	167
Using Variables and Functions in the Rule Editor	168
Using Catalog Functions in the Rule Editor	168
Using Global Variables in the Rule Editor	169
Using the Function Argument Mapping Wizard	169
Using Priority and Rank to Control Order of Rule Execution	171
Examples	172
Using the Quick Fix Feature in the Rule Editor	172
Using the Quick Fix Feature	173
Tips for Working in the Rule Editor	175
Event Preprocessors	176
Transaction Error Handler Rule Function	177
Transaction Error Handler Rule Function Reference	178
Rule Template and Rule Template View	180
Adding a Rule Template View	180
The Rule Template Editor	181
Action Context Section	181
Bindings Tab	182
Tips for Working in the Rule Editor	183
Rule Template Views	184
Rule Template Editor Reference	184
Functions	186
Catalog Functions	186
Built-in Functions	186
Standard Functions	186
Custom Functions	190
Adding Custom Java Functions	190
Adding Libraries	190
Importing Old Projects	191
Compiling Project and Building an EAR File	191
Debugging	191
Annotations Reference	191
Considerations when Defining the Functions	192
Ontology Functions	193
Enabling Extended Functions	194
Function Tooltips and Decorations	194
Temporal Functions and Their Parameters	195
Virtual Rule Functions and VRF Catalog Functions	196
Adding a Virtual Rule Function	197

VRF Function Arguments	198
Cache Related Functions	199
Indexing for More Efficient Cache Queries	200
Create Indexes Using a Coherence Function	200
Creating an Index Using a Domain Object Override Setting	201
Enabling Explicit Tuple Format for DataGrid Fields	201
Query the Cache Using BQL Queries	202
Oracle Coherence Cache Query Functions	203
Structure of a Function Catalog	204
Example Function Catalog	204
Elements in the Function Catalog	205
Using the Reevaluate Element	206
Rule Language Grammar	208
Whitespace	208
Comments	208
Separators	208
Local Variables	209
Primitive Arrays	209
Literals	209
Escape Sequences	210
Operators	211
Keywords and Other Reserved Words	213
Attributes	221
Exception Handling	223
Syntax	223
Examples	223
Flow Control	225
Rule Language Datatypes	226
Concept Properties to XML Datatype Conversions	226
Compatibility of Operators with Types	226
Correcting Inconsistencies of Type	228
Mapping and Transforming Data	230
Function Argument Mapping Wizard for XPath 1.0	230
Toolbar and Right-Click Menu on the Input Section	231
Icons for Schema Element Datatypes	233
Qualifier Icons	234
Function Argument Mapping Wizard for XPath 2.0	235
Mapping and Transforming Data to Function Input	238
Statements Hints and Errors	239

Buttons Menus and Icons	239
Toolbar and Right-Click Menu on the Input Section	239
Icons for Schema Element Datatypes	241
Qualifier Icons	242
Specifying Constants	243
Date and Datetime Strings in Constants	244
Data Validation	244
Incorrect Mappings	244
Repair Incorrect Mappings	245
Migrating Mapper Functions from XPath 1.0 to XPath 2.0	245
Shortcuts	246
Statement Menu Options	247
Dragging to the Left	247
Cutting and Pasting	247
Automatic Testing (at Runtime)	247
Examples of Mappings	248
Setting an Element Explicitly to Nil	248
Merging Input from Multiple Sources	249
Converting a List Into a Grouped List	253
Merging Two Corresponding Lists	255
Coercions	257
XSLT Statements	260
Attribute	260
Choose	260
Comment	261
Copy	261
Copy-Contents-Of	261
Copy-Of	261
Element	261
For-Each	261
For-Each-Group	262
Generate Comment	262
Generate PI	262
If	262
Value-Of	262
Variable	262
XPath Formula Builder For XPath 1.0	263
Addressing Schema Elements	263
Evaluation Context	264

Namespaces	264
Search Predicates	264
Testing for Nil	264
Comments	264
The XPath Formula Builder	265
String Representations of Datatypes	266
Date and Time Functions	267
Cluster Deployment Descriptor	270
TIBCO BusinessEvents and TIBCO Live Datamart Integration	271
Displaying BusinessEvents Project Entities in LiveView Web	272
Generating Live DataMart Configuration Files Using BusinessEvents Studio Tools Utility	273
LiveView Configuration File Reference	274
TIBCO ActiveMatrix BusinessWorks 6 Integration	276
TIBCO Enterprise Runtime for R Integration	279
Configuring BusinessEvents for TIBCO Enterprise Runtime for R	280
BusinessEvents and TIBCO Enterprise Runtime for R Mapping Reference	280
Performance Profiler	282
The Delimiter Character	282
Turning Profiler On and Off	283
Using TIBCO BusinessEvents Monitoring and Management	283
Using Properties	283
Using Functions	285
Using TIBCO Hawk Methods	285
Profiler Reference	286
Testing and Debugging Projects	291
Preparing to Run (Test) or Debug a Project	291
Build an EAR File	291
Create Test Data	291
For Remote Debugging Only Configure Java Debug Interface (JDI)	291
Configure Java Debug Interface (JDI)	291
Adding and Working with Launch (Debug or Run) Configurations	292
Add and Work with Launch Configurations	292
Launch Configurations Reference	293
For Testing and Local Debugging	294
For Remote Debugging	294
Test Data	295
Working with Concept and Event Test Data	295
Creating Concept and Event Instance Test Data	295
Edit Test Data for Concepts and Events in BusinessEvents Studio Explorer	296

Edit Concept and Instance Test Data in the Rule Input View	296
Working with Rule Data	297
Setting Breakpoints in Rules and Rule Functions	297
Running Debugger	298
Running Tester	299
Run the Tester	299
Asserting Rule Input Data	299
Assert Tester Data	300
Viewing the Results	300
Viewing and Understanding Working Memory Contents	301
Unit Testing Using JUnit	302
Adding Unit Test Suite for the BusinessEvents Project	303
Shared Resources	307
Adding a Shared Resource	307
Enabling the Test Connection Feature	307
ActiveSpaces Connection Reference	308
Wizard and Configuration Tab	308
Hawk Connection Reference	311
Wizard and Configuration Tab	311
HTTP Connection Reference	312
Wizard and Configuration Tab	313
SSL Configuration	314
Identity Resource Reference	315
Wizard and Configuration Tab	315
JDBC Connection Reference	316
JDBC Connection Wizard and Configuration Tab	316
JDBC SSL Configuration Reference	319
Configuring SSL for JDBC Connection	320
Connection Pooling	321
Test Connection Button	321
JMS Application Properties	321
Wizard and Configuration Tab	321
JMS Connection Reference	322
Wizard and Configuration Tab	322
Test Connection Button	325
Advanced Tab	325
Configure SSL	325
JNDI Configuration Reference	327
Wizard and Configuration Tab	327

Advanced Section	328
Rendezvous Transport Reference	328
Wizard and Configuration Tab	329
Configure SSL	329
Advanced Section	330
StreamBase Connection Reference	331
Domain Models	333
Set Up a Domain Model	333
Domain Model Value Descriptions for User Friendly Presentation	333
Adding a Domain Model	333
Add a Domain Model	334
Adding Domain Entries	334
Importing and Exporting Domain Models	335
Exporting To and Import From Excel	336
Importing Domain Model Entries from Excel	336
Export Domain Values to Excel	336
Importing Domain Model Entries from a Database Table	337
Associating Domain Models with Properties	338
Associate Domain Models with a Property	338
Validating Data in Domain Models	338
Display Models	339
Creating a Display Model	340
Editing a Display Model	340
Understanding Entity Caches	342
Caches for Ontology Objects	342
Caches for Internal Entities	342
Handling Null Properties	344
Enabling Use of the Nillable Attribute	344
Enabling Null Property Values to Appear When Serializing Concepts to XML	344
Examples of Nillable Attribute and Null Properties Settings	344
Special Treatment of Numeric and Boolean Null Values	345
Setting Runtime Properties for Special Treatment of Null Values	346
Property Reference for Null Property Handling	346
Diagrams	348
Working with Diagrams	349
Common Tasks	350
Diagram Tools	350
Interactive Tools	351
Layout	351

Context Menu Diagram Tools	351
Exporting a Diagram to an Image	352
Printing a Diagram	353
Setting Diagram Preferences	354
Project Analyzer and Selected Entity Project Diagrams	354
Advantages of Project Analyzer and Selected Entity Project Diagrams	355
Project Analyzer and Selected Entity Project Diagrams	355
Create a Selected Entity Project Diagram	355
Dependency Diagrams	356
Create a Dependency Diagram	357
Sequence Diagrams	358
Create a Sequence Diagram	358
Concept Model Diagrams	358
Create a Concept Model Diagram	359
Event Model Diagrams	359
Create an Event Model Diagram	360
Diagram Options and Tools Reference	360
Layout Options	362
Preferences	363
Setting Preferences	364
Decision Table Related Preferences	365
Global Diagram Preferences	366
Tester Preferences	370

Figures

Mapper Function Migration Wizard Confirmation	27
Serializer and Deserializer Behavior	55
Kafka Channel BusinessEvents Studio UI	126
Rule Form Editor	162
Rule Source Editor	162
The Function Argument Mapping Wizard in XPath 1.0 Project	170
The Function Argument Mapping Wizard in XPath 2.0 Project	171
Temporal Functions Parameters	196
The Function Argument Mapping Wizard in XPath 2.0 Project	236
XPath Formula Builder for XPath 2.0	238
Mapper Function Migration Wizard Preview Window	246
Example Schema of Merging Input From Multiple Sources	249
Example Schema of List Conversion to a Grouped List	253
Example Schema of Merging of Two List	255
Example Schema of Coersion	257
Creating an XPath formula	266
LiveView Wev Sample Dashboard	273
RTC and TIBCO Enterprise Runtime for R Engine	279

TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, visit:

<https://docs.tibco.com>

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site. To directly access documentation for this product, double-click the following file:

`TIBCO_HOME/release_notes/TIB_businessevents-standard_version_docinfo.html` where `TIBCO_HOME` is the top-level directory in which TIBCO products are installed. On Windows, the default `TIBCO_HOME` is `C:\tibco`. On UNIX systems, the default `TIBCO_HOME` is `/opt/tibco`.

The following documents for this product can be found in the TIBCO Documentation site:

- *TIBCO BusinessEvents Installation*
- *TIBCO BusinessEvents Getting Started*
- *TIBCO BusinessEvents Architect's Guide*
- *TIBCO BusinessEvents Developer's Guide*
- *TIBCO BusinessEvents Configuration Guide*
- *TIBCO BusinessEvents WebStudio User's Guide*
- *TIBCO BusinessEvents Administration*
- Online References:
 - *TIBCO BusinessEvents Java API Reference*
 - *TIBCO BusinessEvents Functions Reference*
- *TIBCO BusinessEvents Release Notes*

How to Contact TIBCO Support

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

- 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.

How to Join TIBCO Community

TIBCO Community is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCO Community offers forums, blogs, and access to a variety of resources. To register, go to the following web address:

<https://community.tibco.com>

Project Development

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

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

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

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

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

Before you begin to use *TIBCO BusinessEvents Developer'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 *TIBCO BusinessEvents Configuration Guide* provides information on how to configure the project's cluster deployment descriptor.

For in-depth explanations about designing a TIBCO BusinessEvents project and other topics, read *TIBCO BusinessEvents Architect'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 *TIBCO BusinessEvents Installation*.

Procedure

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

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

Names follow Java variable naming restrictions. Do not use any reserved words.

5. Accept the default location or clear the **Use default location** check box and specify the desired location.
6. Click **Finish**.

Result

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.

Importing Projects in TIBCO BusinessEvents Studio

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

Procedure

1. Select **File > Import** .
The Import wizard is displayed.
2. In the Import wizard, select **TIBCO BusinessEvents > Existing TIBCO BusinessEvents Studio Project** and click **Next**.
The Existing TIBCO BusinessEvents Studio Project Import Wizard is displayed.
3. In the **Existing project root directory** field, click **Browse** and select the project root directory of the project you are importing.
4. Select the XPath version to be used for the project in the **XPath Version** dropdown.
5. Do one of the following:
 - Select the **Copy project into workspace** checkbox. This option copies the project into your current workspace.
 - Clear the **Copy project into workspace** checkbox and specify an import location.
6. Do one of the following:
 - If your project has no HTTP channels, click **Finish** (this is the only option).
 - If your project has one or more HTTP channels, click **Next**.
7. If you clicked **Next** then in the Select Processing Unit window select the project CDD in the **Cluster Deployment Descriptor** dropdown.
8. Select the processing unit in the **Processing Unit** dropdown that contains the HTTP properties to be migrated.
An informational panel displays the settings that will be migrated.
9. Click **Finish**.
The imported project is displayed in the TIBCO BusinessEvents Studio.

Finding a Project Element

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

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

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

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

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

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

In the XSD files, concepts are represented as follows:

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

Generating an XML Schema (XSD File)

Procedure

1. Select the project 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 and select the folder where you want to put the schema files.
4. Select the **Override TIBCO BusinessEvents Namespace** check box to specify a different namespace.

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

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

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

Result

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



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

Procedure

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

See also [Using the Quick Fix Feature in the Rule Editor](#) for a related use of this feature.

4. In addition to validating for internal consistency, you can run a project analyzer feature. It helps you to understand your project better, and find ways to improve it.

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

Working with External Library and Custom Function Paths

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

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



To make the Test Connection feature work for JMS Connection and JDBC Shared Connection shared resources, see [Enabling the Test Connection Feature](#).

Adding an External Library or Custom Function Path

This functionality is also available in the **Debug Configurations > ClassPath**, and the **Run Configurations > ClassPath** tab, for your convenience.

Procedure

1. Open the project in TIBCO BusinessEvents Studio.
2. In BusinessEvents Studio Explorer, right-click the project and click **Properties** (or press Alt-Enter). You see the properties dialog for the project.
3. In the left panel, select **Build Path** 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 select the JAR file.
5. If a library or custom function depends on additional third-party Java archives, follow these steps:
 - 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 complete [step 5](#), a dialog is displayed when you click **OK**. It lists the name of the class or classes that could not be loaded. To resolve the problem, complete [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.

Result

Deploymenttime Action Required

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

- Either copy the JAR or JARs to the `lib/ext/tpcl` directory, or other directory in the class path; or update the class path in the TRA file to point to the location of the JAR or 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 *TIBCO BusinessEvents Administration* for more details.

To Remove an External Library or Custom Functions JAR Path, follow the instructions in [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.

Specifying an External Library Location Using Build Path Variable

You can use variables (aliases) to specify an external library location in project build path to build a project using studio-tools in TIBCO BusinessEvents. You can store the third party JARs, custom function JARs, and `.projlib` files in different locations on your local machines and refer to them through aliases. To perform common builds on a build machine, the aliases are used to create the full path to common JAR files.

Procedure

1. Open the project in TIBCO BusinessEvents Studio.
2. In BusinessEvents Studio Explorer, right-click the project and click **Properties** (or press Alt-Enter). You see the Properties dialog for the project.
3. In the left panel, select **Build Path** and select **Custom Functions** tab.
4. Click **Add Variable**.
The New Variable Classpath Entry window is displayed.
5. Click **Configure Variables**.
The Preferences window is displayed to add a new variable.
6. Click **Add**.
The New Variable Entry window is displayed.
7. In the **Name** field, enter the new variable name (for example, `CUSTOM_HOME`) and in the **Path** field enter the path and click **OK** three times.
The new variable is now added to the build path.

Result

The variable is used as a placeholder in the build path. Define the value of that variable on your local machine (for example, `C:\dev\mycustomjars\`), and similarly define the value on the build machine. Thus, there are no hard-coded paths to the custom function, or third-party project library entries, but only the relative ones.

The two methods using which you can specify where to find the JAR files during deploy time are:

- **Using classpath (-cp) option** - In the command-line specify the path of the JAR files in the `-cp` command line option for the classpath. For example:

```
studio-tools.exe -cp c:\logs\677138\cf-logger51.jar -core buildEar -o c:\temp\hello.ear -p C:\Logs\677138\HelloWorld_with_CustomFunction
```

- **Using the studio-tools.tra file** - In the `studio-tools.tra` file, which is used to build the EAR file on the build machine, add the following property for each defined variable:

```
java.property.CUSTOM_HOME=C:/Logs/677138/
```

where, `CUSTOM_HOME` is the name of the variable.

Now you can build EAR without specifying classpath in the command, for example:

```
studio-tools.exe -core buildEar -o c:\temp\hello.ear -p C:\Logs\677138\HelloWorld_with_CustomFunction
```



Do not put the name of the JAR file at the end when specifying the value of the property. In the `studio-tools.tra` path, forward slashes are used.

Working with Project Libraries

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

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

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

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

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



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

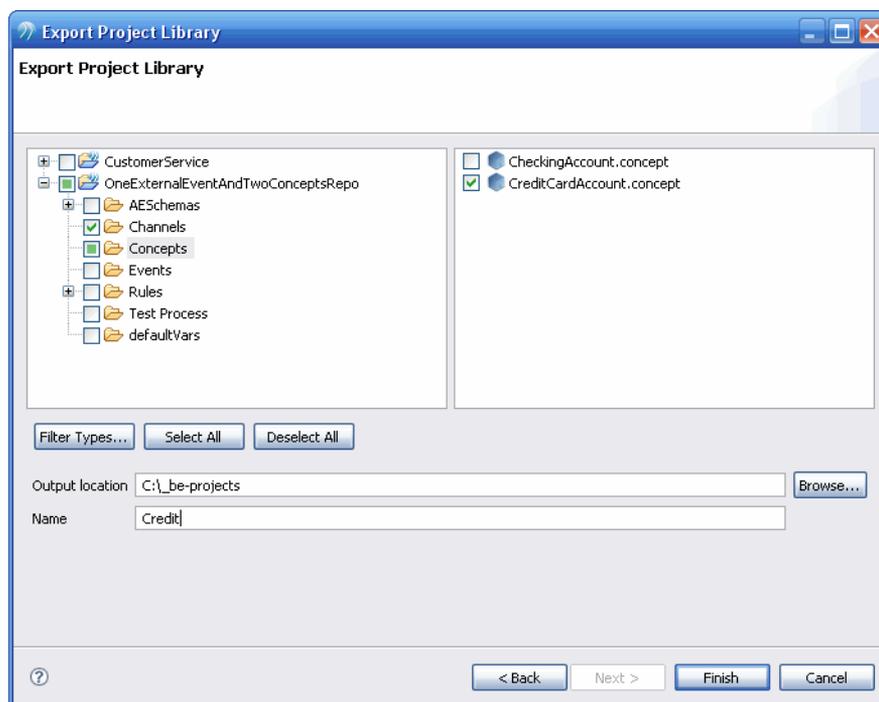
If multiple project libraries have the same element, then the first one on the build path wins, and the other ones are ignored.

Creating (Exporting) a Project Library

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

Procedure

1. Optionally, select a project or any of its resources in TIBCO BusinessEvents Studio.
2. Do one of the following:
 - From the **File** menu select **Export**.
 - Right-click anywhere in BusinessEvents Studio Explorer and select **Export**.
In the Select dialog, select **TIBCO BusinessEvents > Project Library** and click **Next**. You see the Export Studio Project Library dialog.
3. In the Export Studio Project Library dialog, select the resources you want to export. If you selected resources in BusinessEvents Studio Explorer they are selected automatically.



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

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



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

Adding (and Removing) Project Libraries in a Project

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

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

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

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



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

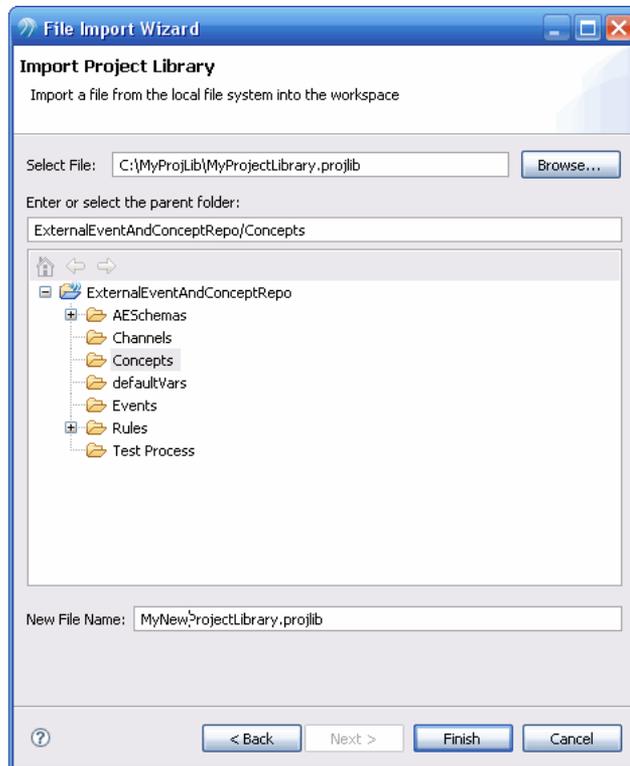
Procedure

1. Open the project in TIBCO BusinessEvents Studio.
2. In BusinessEvents Studio Explorer, right-click the project name and click **Properties** (or press Alt-Enter, or select ProjectProperties). You see the properties dialog for the project.
3. In the left panel, select **Build Path** and then select the **Project Libraries** tab.
4. Click **Add Library** and browse to and select the desired project library (.projlib) file.
5. Click **OK**. You are prompted to rebuild the project.
6. Save the resource. The project library appears at the root of the project tree as a Project Dependencies node.

Importing a Project Library

Procedure

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



- Select **File** field, browse to and select the project library (.projlib) file.
- 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.
- Click **Finish**. The project library appears at the root of the project tree as a Project Dependencies node.

Removing a Project Library

Procedure

- Open the project in TIBCO BusinessEvents Studio.
- In BusinessEvents Studio Explorer, right-click the project name and click **Properties** (or press Alt-Enter, or select ProjectProperties). You see the properties dialog for the project.
- In the left panel, select **Build Path** and then select the **Project Libraries** tab.

4. Browse to and select the desired project library (.projlib) file.
5. Click **Remove Library**.

Changing XPath Version of The Project

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

Procedure

1. In BusinessEvents Studio select the project for which you want to change the XPath version.
2. Right click the project and select **Properties**, or click **File > Properties**.
The property window for the project is displayed.
3. Select **Build Path** and select the appropriate XPath version in the **XPath Version** field.
4. Click **OK**

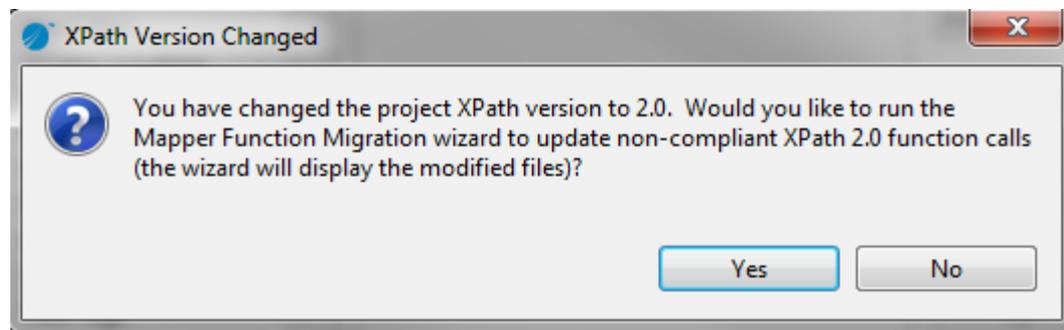
Alternatively, you can edit the .beproject file and update the value of the `xpathVersion` property to 1.0 or 2.0.

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



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

Mapper Function Migration Wizard Confirmation



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

What to do next

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

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](#)).

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.

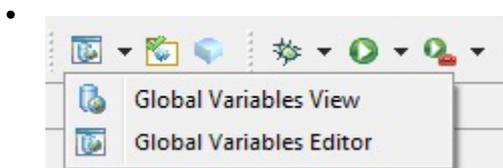
To see all global variables, including overridden ones, open the global variable editor and expand **Project Dependencies** and project libraries. Global variables from project libraries are not editable.

Adding and Managing Global Variables

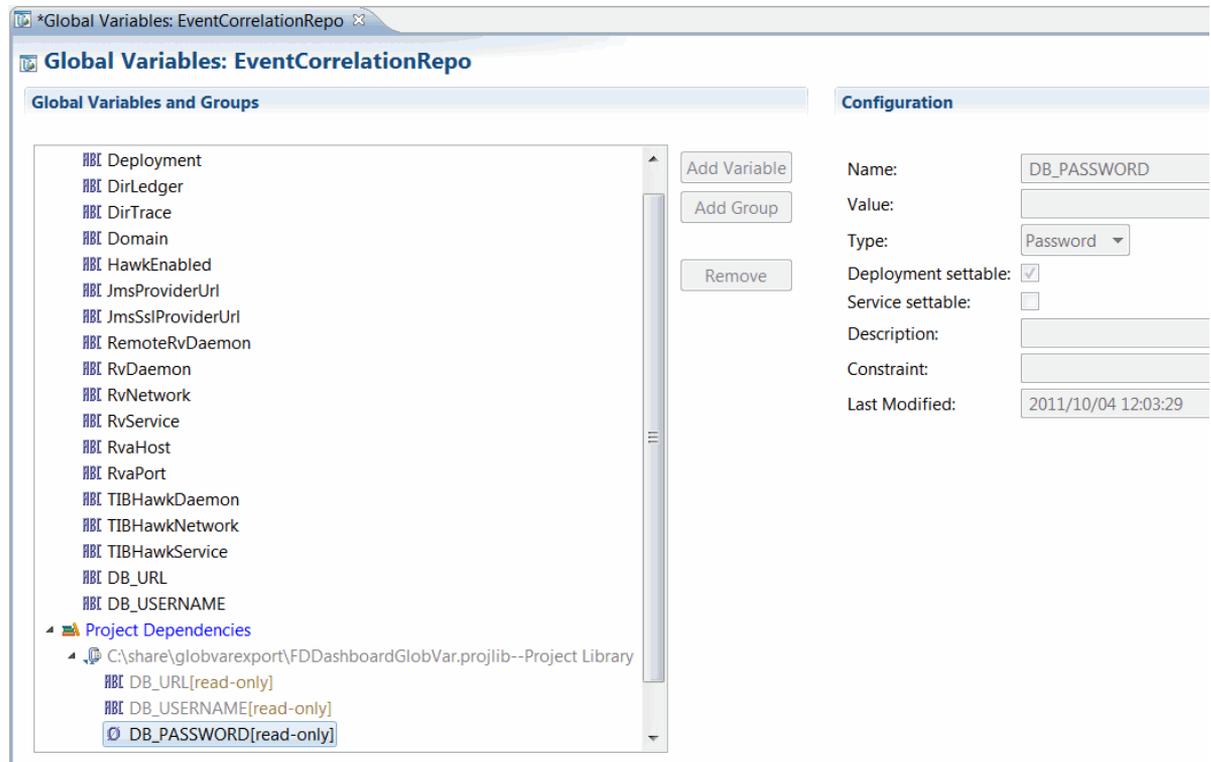
In a new TIBCO BusinessEvents Studio project no global variables are predefined. When you import a project from an earlier version of TIBCO BusinessEvents, however, you see predefined global variables as well as any others defined in the project.

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**.



Add and Manage Global Variables

Procedure

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

You see the global variables editor listing the variables available, if any.
3. Do any of the following. (See [Global Variable Reference](#) 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**.



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.

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 shown 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:

```
-VVariable_Group/Variable_Name=
```

Global Variable Reference

Global Variable Reference

Field	Description
Name	The variable name. Different groups can have the same-named variable, because the group name is included when you use the global variable.
Value	The variable value. Varies according to type.
Type	Data type of the global variable. The values are: <ul style="list-style-type: none"> • String • Integer • Boolean • Long • Password <p>The type must match the type of the field where the global variable is used, or errors result.</p>

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

Overriding Global Variables at Deploy Time

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

- For starting at the command line:
 - In the design time CDD file.
 - At the command line, using the `--propVar` option, or using the `-p` option to specify a property file where the override properties are defined. See *TIBCO BusinessEvents Administration*.
- For deployment using TIBCO BusinessEvents Monitoring and Management component, set values in the CDD or MM Console. CDD values can be overridden in MM Console. See *TIBCO BusinessEvents Administration*.

- For deployment using TIBCO Administrator, set overrides in the TIBCO Administrator UI. You can override variables at the deployment level or at the service level. Values set at the service level are used for the specific engine you are deploying. Service settable global variables are only available if the **Include All Service Level Global Variables** check box in the Build Enterprise Archive dialog is selected (see *TIBCO BusinessEvents Administration*).

Order of Precedence of Global Variable Overrides

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

1. Command-line arguments at engine startup (highest priority).
2. Property files specified at command-line engine startup
3. 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 Reference](#) or [JMS Connection Reference](#).

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

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

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

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

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

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

Store Trusted Certificates Outside of the Project

Procedure

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

The location can be the same for all deployed engines (that is, you copied it to the same location on each machine or it is a shared network drive). Alternatively you can change the value of the global variable as needed when you deploy the project

4. Specify a value in the Trusted Certificates field in the SSL Configuration dialog. When the project runs, the value of BE_GLOBAL_CA_STORE is used, and not the value you specify in the Trusted Certificates field.
5. Save the resource.

Remote Terminal

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

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

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

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

TIBCO BusinessEvents Studio Tools Utility

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



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

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

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

Building an EAR File at the Command Line

Procedure

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] [-lc] [-o outputEarFile>] -p
studioProjectDir [-pl projectLibrariesFilePath] [-cp extendedClasspath]
```

For example:

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

[TIBCO BusinessEvents Studio Tools Options for Building an EAR File](#), provides detailed information about the options.

TIBCO BusinessEvents Studio Tools Options for Building an EAR File

Option	Description
<code>-core buildEar</code>	Within the core category of operations, specifies the <code>buildEar</code> operation for building EAR files.
<code>-h</code>	Optional. Displays help.
<code>-x</code>	Optional. Overwrites the specified output file if it exists.
<code>-lc</code>	Optional. Specifies that the file-based legacy compiler must be used to build the EAR file. By default, the EAR files are built in memory.
<code>-o</code>	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 <code>projectDir</code> path.
<code>-p</code>	Absolute path to the TIBCO BusinessEvents Studio project directory. The EAR file is built using this project.

Option	Description
<code>-pl</code>	Optional. Specifies list of project library file paths to be used, separated by a path separator.
<code>-cp</code>	Optional. Specifies the extended classpath to be used.

Result



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

Importing an Older 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 Studio Project**. It is used to migrate older (4.x and 5.x) projects to 5.4.

When you run this command-line utility, various migration actions are done, as explained in *TIBCO BusinessEvents Installation*.

For migration of version 3.x projects, see *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 [Open the Imported Project in TIBCO BusinessEvents Studio](#).

Import an Existing 4.x Project at the Command Line

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

Procedure

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

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

For example:

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

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.
```

TIBCO BusinessEvents Studio Tools Options

TIBCO BusinessEvents Studio Tools Options for Importing an Existing Project

Option	Description
<code>-core importExistingProject</code>	Specifies the <code>importExistingProject</code> operation for importing a TIBCO BusinessEvents Studio project into the workspace.
<code>-h</code>	Optional. Displays help.
<code>-p</code>	Source project: absolute path to the project directory of the TIBCO BusinessEvents Studio project to be imported.
<code>-o</code>	<p>Optional. Absolute path to the target project directory, where the project is imported to.</p> <p>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.</p> <p>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.</p> <p>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.</p>
<code>-c</code>	Optional. The CDD to use for migration actions. Project path of the CDD (path relative to the root directory of the source project).
<code>-u</code>	<p>Optional. If specified, then the <code>-c</code> option must also be specified. Specifies the name of the PU (within the specified CDD) that contains settings to be migrated.</p> <p>HTTP channel settings from this PU are migrated to all HTTP channel resources in the project.</p>
<code>-xp</code>	<p>Optional. The XPath version to be compatible with the project. The values are:</p> <ul style="list-style-type: none"> • (default) 1.0 • 2.0

Open the Imported Project in TIBCO BusinessEvents Studio

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

Procedure

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

Working With Project Libraries at the Command Line

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

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

- Create a project library (and optionally overwrite any existing library at the same location with the same name). This action creates the `.projlib` file. The corresponding action in TIBCO BusinessEvents Studio is exporting a project library.
- Add an existing project library to a TIBCO BusinessEvents Studio project. The corresponding action in TIBCO BusinessEvents Studio is importing a project library (or adding it to the build path property page).
- Remove an existing project library from a TIBCO BusinessEvents Studio project. The corresponding TIBCO BusinessEvents Studio action is removing the library from the Build Path property page.

Work with Project Libraries at the Command Line

Procedure

1. Navigate to `BE_HOME/studio/bin/` 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 -core buildLibrary [-h] -p studioProjDir [-a] [-r] [-x] -n
projectLib [-f resources]
```

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

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

Examples of Creating a Project Library

Example 1

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

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

Example 2

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

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

Examples for Adding and Removing a Project Library

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

Example 3

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

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

Example 4

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

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

[Table 29](#) provides detailed information about the options.

TIBCO BusinessEvents Studio Tools Options for Working with Project Libraries

Option	Description
<code>-core buildLibrary</code>	Specifies the <code>buildLibrary</code> tool for adding, removing, and creating (overwriting as needed) project libraries.
<code>-h</code>	Optional. Displays help.
<code>-p</code>	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.
<code>[-r -a]</code>	Use one of these arguments as needed. <code>-r</code> : removes the specified project library from the project specified by <code>-p</code> . <code>-a</code> : adds the specified project library to the specified project.
<code>-x</code>	Optional. Overwrites any existing project library at the location specified by <code>-n</code> .
<code>-n</code>	The absolute path of the project library to be created or added or removed

Option	Description
<code>-f</code>	Optional. A comma-separated list of resources to include in the project library, using a path relative to the <code>projectDir</code> . 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.

Migrate Core Coherence Functions at the Command Line

Procedure

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:

Result

```
studio-tools -core migrateCoherenceCalls -p studioProjectDir
```

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

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

Migrating Mapper Functions to XPath 2.0 Using Command Line

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

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



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

Procedure

1. Navigate to `BE_HOME/studio/bin/` and open a command prompt.

- Execute a command with the following format (all in one line) at a command prompt:

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

Where:

Studio Tools Options for Migrating Mapper Functions

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



You cannot undo the changes after executing this command.

For example:

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

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.

Generate Class Files at the Command Line

Procedure

- Navigate to `BE_HOME/studio/bin/` and open a command prompt.
- 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}] [-lc] [-pl projectLibraryFilePath] [-cp
extendedClasspath]
```

For example:

```
studio-tools -core generateClass -p D:\Workspace\FraudDetection -o c:\temp -x
true -cp c:\tibco\be\5.4\lib\myjar.jar
```

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

TIBCO BusinessEvents Studio Tools for Generating Class Files

This table provides detailed information about the options.

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.

Option	Description
-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.
-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 <code>true</code> , overwrites any existing class file with the same name.
-lc	Optional. Specifies that the file-based legacy compiler must be used to build the EAR file. By default, the EAR files are built in memory.
-pl	Optional. Specifies list of project library file paths to be used, separated by a path separator.
-cp	Optional. Extended classpath. Use as needed. Provide separate JAR file paths for each JAR file required for project compilation. For example, additional classpath information is needed if the decision table uses custom functions or third-party JAR files. Separate entries by the appropriate path separator. For example, if the separator is semicolon (;) you might add the following: C:\customjars\custom.jar;C:\customjars\custom2.jar

Generating Encrypted Passwords

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

Syntax

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

Definition

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

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

[TIBCO BusinessEvents Studio Tools Options for Generating Encrypted Passwords](#) provide more detailed information about the option.

TIBCO BusinessEvents Studio Tools Options for Generating Encrypted Passwords

Option	Description
-h	Optional. Displays help.
-i	Input text which needs to be encrypted

Running the Tester on the Command Line

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

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

Procedure

1. Run the command

```
-tester generate
```

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

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



Blank columns should be filled with NA.

To generate a file in a specific format, see [Command Line Options](#).

2. Enter the data manually in the generated template files.
3. Run the command

```
-tester import
```

This will import the data entered into test data files

```
*.concepttestdata
```

and

```
*.eventtestdata.
```

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

4. Run the command

```
-tester assert
```

This will start an instance of the TIBCO BusinessEvents engine and assert the test data generated in step 2 to the instance.

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

- **Default Format:**
It is either the XML format (default) or HTML format (defined through the command line option).
- **Custom Format:**
Output is based on Types or Operation (such as Created, Modified, and Deleted).

The default output is generated based on Operation.

Command Line Options

Result

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

- [generate](#)
- [import](#)
- [assert](#)

The Generate Command

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

Syntax

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

Options

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

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

The Import Command

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

Syntax

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

Options

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

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

The Assert Command

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

Syntax

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

Options

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

Option	Description
-c	(required). Specifies the CDD file path.
-f	(optional). Specifies the output format type. Acceptable values are <code>ops</code> and <code>type</code> . If this option is not specified, the default output is generated.
-h	(optional). Prints usage help.

Option	Description
-i	(optional). Specifies the input directory path. If not specified, it will attempt to use the TestData directory under the project directory.
-n	(optional). Name of the resultFile. If not specified, it creates a file with the name Run-#.resultdata.
-o	(optional). Specifies the output directory. If not specified, the output directory will be ProjectDirectory/results.
-p	(required). Specifies the project path.
-r	(required). Specifies the EAR file path.
-u	(optional). Specifies the processing unit name. If not specified, it will use the default processing unit.
-w	(required). Specifies the working directory path.
-x	(optional). Overwrites the output file if it already exists.

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.

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](#)).

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

Updating All References is Strongly Recommended

To ensure the integrity of the project, ensure that you make all changes to all locations where a renamed or moved items is referenced. Disable such changes only 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

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

Project Level Actions

You can rename a project (select **File > Rename**) 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.

Renaming a Project Element

Procedure

1. Rename the element using one of the following methods:
 - Right-click a project element in BusinessEvents Studio Explorer and select **Refactor > Rename** .
 - Highlight the element, 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, clear the **Update References** check box; however, you should update all references to ensure the integrity of the project (see [Updating All References is Strongly Recommended](#)).
4. Click **Preview**. One of the following occurs:
 - The preview page displays so you can examine the effect of this change. See [Working with the Preview Page](#) for details.
 - The problems page displays if renaming cannot be done, for example, because a new element name is used by an existing element. Click **Back** to fix the problem, or **Cancel** to cancel the rename.
5. In the preview page, clear check boxes if you do not want the change to be made in some referenced locations; however, you should update all references to ensure the integrity of the project (see [Updating All References is Strongly Recommended](#)).
6. To complete the change, click **OK**.

Moving an Element to a Different Project Folder

Procedure

1. Move the element using one of the following methods:
 - Right-click a project element name in BusinessEvents Studio Explorer and select **Refactor > Move** .
 - Highlight the element, then select **File > Move** .
 - Drag the element to the target folder (go to [step 3](#)).
2. If you opened the Move Element wizard using menus, navigate the project tree in the first page of the wizard to select a destination folder, then click **Preview**.
3. One of the following occurs:
 - The preview page displays so you can examine the effect of this change. See [Working with the Preview Page](#) for details.
 - The problem page displays if the move cannot be made. Click **Back** to fix the problem, or **Cancel** to cancel the move.
4. In the preview page, clear check boxes if you do not want the change to be made in some referenced locations; however, you should update all references to ensure the integrity of the project (see [Updating All References is Strongly Recommended](#)).

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

Delete an Element or Folder

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

Procedure

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

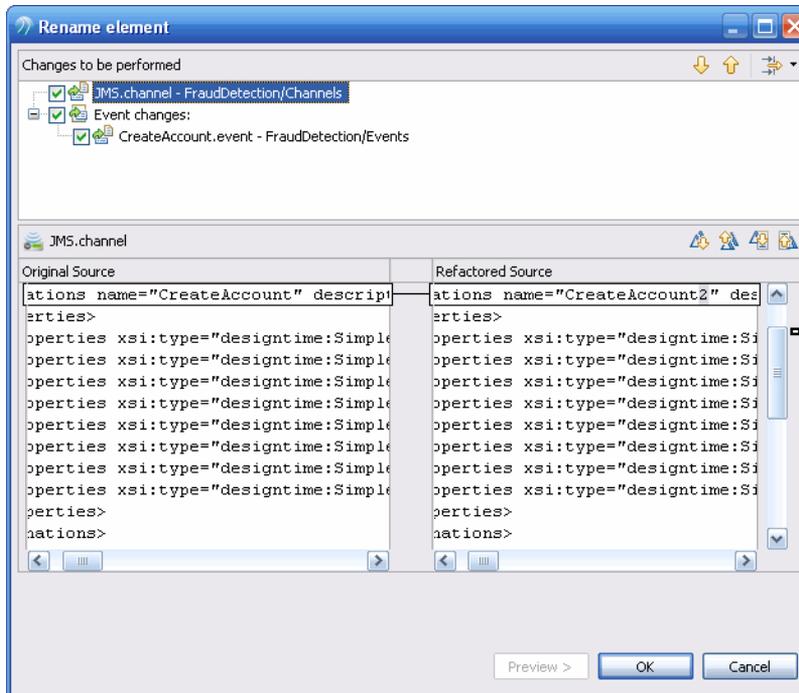
Deleting a Project

Procedure

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

Working with the Preview Page

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



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, clear the check boxes 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.



Accept all changes to be performed

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

Copy-Pasting an Element

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

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

```
rule SomeRules.Application_Rule
```

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

```
rule OtherRules.Application_Rule
```

Procedure

1. In BusinessEvents Studio Explorer, highlight the element (or folder) and perform one of the following actions.
 - Right-click the element and select **Copy**.
 - Press **Ctrl+C**.
 - Select **Edit > Copy**.
2. Highlight the project folder where you want to paste the element, and perform one of the following actions.
 - Right-click the folder and select **Paste**.
 - Press **Ctrl+V**.
 - Select **Edit > Paste**.

Result

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

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. See [Mapping of Coherence Functions to DataGrid Functions](#) 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](#).

Migrate Coherence Function Calls in TIBCO BusinessEvents Studio

See [Mapping of Coherence Functions to DataGrid Functions](#) for details of the functions that are mapped using this feature.

Procedure

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 is displayed 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 clear the check boxes 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 DataGrid Functions

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()

4.x Coherence Category Function Name	DataGrid Category Function Name
C_CacheName()	CacheName()
C_CacheReevaluate()	CacheReevaluate()
C_ClassName()	ClassName()
C_EnableCacheUpdate()	EnableCacheUpdate()
C_Index()	Index()
C_Lock()	Lock()
C_TransactionProperties	TransactionProperties()
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 36](#). Projects can be complex; this list covers the main cases.

Refactoring for Move and Rename Operations

Renaming this...	Updates references in these places...
Concept	<ul style="list-style-type: none"> Concepts that inherit from this one Event expiry actions Property definitions for contained or referenced concepts State models Rules and rule functions
Event	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> Event expiry actions State models Rules and rule functions
Domain Model	<ul style="list-style-type: none"> Domain models that inherit from this one Associated properties
Channel	<ul style="list-style-type: none"> Event default destination paths
Destination	<ul style="list-style-type: none"> Event default destination paths
Rule	<ul style="list-style-type: none"> The rule source Rules and rule functions
Rule Function	<ul style="list-style-type: none"> Event expiry actions State models The rule function source Rules and rule functions
Folder	<p>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 .</p>

Refactoring for Delete Operations

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

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

- References to a deleted domain model are removed from concepts that refer to it.

- References to a deleted state model are removed from the owning concept.

Channels and Destinations

One project can have multiple channels of different types with multiple destinations as needed. For more information, see *TIBCO BusinessEvents Architect's Guide*.

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

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

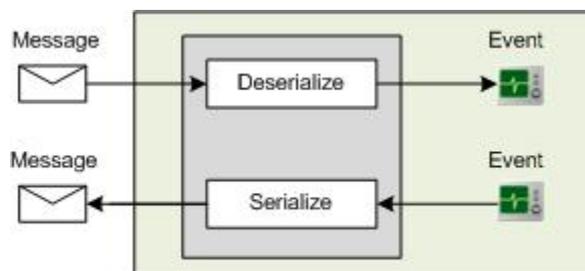
The screenshot displays the configuration window for a 'Channel: RV'. The 'Configuration' section includes a 'Description' field, a 'Driver' dropdown set to 'RENDZVOUS', and 'Methods of Configuration' set to 'Properties'. Below this is the 'Extended Configuration' section with 'Properties' expanded, showing fields for 'Service', 'Network', and 'Daemon'. The 'Destinations' section features an 'Add' (+) and 'Remove' (-) button, a list containing 'input', and a detailed configuration form for the 'input' destination. This form includes fields for 'Name', 'Description', 'Default Event' (with a 'Browse..' button), 'Serializer/Deserializer' (a dropdown menu), 'Subject', 'RVC Pre Registration', 'Limit Policy' (a dropdown menu), 'Max Events', and 'Discard Amount'.

Channel Serializers

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

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

Serializer and Deserializer Behavior



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

Mapping Incoming Messages to Non-default Events

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

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

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

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

Working with Rendezvous Channels

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).

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

For TRA file updates required for Rendezvous and JMS channels that use TIBCO Enterprise Message Service, see TIBCO BusinessEvents Administration.

Rendezvous Message Header

For Rendezvous messages, the only header that TIBCO BusinessEvents interprets is `_sendsubject_` 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 `TibRvMsgSerializer` serializer is used for efficient handling of events and messages that do not have payloads. It ignores payloads in messages and in events if any exist.

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	<code>TibrvMsg.STRING</code>
TibrvXML	<code>TibrvMsg.XML</code>

Data Type	Wire Format Type
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.

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

Avoiding REGISTRATION COLLISION RVCM Advisory Messages

REGISTRATION COLLISION RVCM Advisory Messages should be avoided.

When you are configuring RVCM using a Rendezvous transport shared resource, an advisory message named REGISTRATION_COLLISION RVCM can be thrown if the same CM name (or CMQ name) is used on all engines.

To prevent name collisions that can result in thrashing, 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. 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.

Setting UTF-8 Message Encoding for the Rendezvous Channel

Use the UTF-8 message encoding, if data that is sent through the Rendezvous channel contains characters outside its default character set - ISO8859-1, such as CJK (Chinese, Japanese, and Korean) characters.

Procedure

1. In TIBCO BusinessEvents Studio, select the BusinessEvents project.
2. Click **Project** on the menu bar and select **Edit Global Variables**.
3. Create a global variable named `MessageEncoding` and set its value to UTF-8.
4. Save the project.

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*.

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.

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

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

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

The signature of this function is as follows:

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

Adding a Channel

Procedure

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



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 [Element Refactoring Operations](#) for more details.

3. In the **Driver type** field, select the appropriate driver:

- Rendezvous
- Local
- HTTP
- JMS
- StreamBase
- ActiveSpaces
- Hawk

If you select Local, channel configuration is complete.

4. Click **Finish**. You see the Channel editor.

Editing a Channel

Procedure

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



The path to the resource and the resource name should not contain any of the keywords or other reserved words listed in *TIBCO BusinessEvents Architect's Guide*.
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.
7. Continue to the section [Adding a Destination to a Channel](#) 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.

Procedure

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

5. In the **Serializer/Deserializer** field, select the appropriate class. See the following sections for more details:
 - For JMS destinations: see *TIBCO BusinessEvents Architect's Guide*.
 - For Rendezvous destinations: see *TIBCO BusinessEvents Architect's Guide*.
 - For HTTP and SOAP destinations: [Add a Destination](#) (Destination configuration is the same for SOAP and HTTP destinations). See [Manually Creating Resources to Work with SOAP Services](#) and [Creating Resources Using the WSDL Import Utility](#) for more information about creating SOAP channels and destinations)
 - For Hawk destinations: The Hawk channel allows TIBCO BusinessEvents to receive events from the Hawk monitor and transform them to events in TIBCO BusinessEvents.
 - For StreamBase destinations: see [StreamBase Channels](#).
 - For ActiveSpaces destinations: [Working with ActiveSpaces Channels](#).

Various fields appear, depending on your selection.

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

See [Destination Resource Common Properties](#) for more details on this property.

7. Complete the rest of the properties for the type of destination you are creating. See [Destination Resource Reference](#) for details.
8. 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, follow these steps to communicate with TCP servers:

Procedure

1. Create a local TCP server in a startup function using `TCP.createLocalServer()`.
2. Connect to a Remote TCP server as a client using.
3. `TCP.connectToRemoteServer()`.
4. Register the session listener using `TCP.registerSessionListener()`.
5. Start the local server using `TCP.startLocalServer()`.
6. 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 *TIBCO BusinessEvents Architect's Guide*.

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.

Channel Configuration Reference

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.
Description	No	Short description of the resource.
Driver	No	Select the driver for the type of channel you are configuring: <ul style="list-style-type: none"> • TIBCO Rendezvous • Local • HTTP • JMS • StreamBase • ActiveSpaces • Hawk

Field	Global Var?	Description
Method of Configuration	No	<p>Properties - Select Properties to configure this channel resource using properties. The properties configuration is depended on the value of Driver. If Driver is:</p> <ul style="list-style-type: none"> • Rendezvous - see Configuration for TIBCO Rendezvous Channels • JMS - see Configuration for JMS Channels • StreamBase - see Configuration for StreamBase Channels • ActiveSpaces - see Configuration for ActiveSpaces Channels <p>Resource - Select Resource if you have a shared resource in your project whose properties you want to reuse for this channel.</p> <p>Note: The path to the resource and the resource name cannot contain any of the words listed in <i>TIBCO BusinessEvents Architect's Guide</i>.</p>
Resource	No	If you choose Resource as the method of configuration, the Resource field appears. Browse to and select the resource you want to use. For convenience, you can open the selected resource by clicking the underlined label.

Configuration for TIBCO Rendezvous Channels

TIBCO Rendezvous Channel Properties

Field	Global Var?	Description
Service	Yes	<p>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.</p> <p>For more information about the Rendezvous service parameter, see <i>TIBCO Rendezvous Concepts</i> or <i>TIBCO Rendezvous Administration</i>.</p> <p>Default is 7500 (defined in Global Variables).</p>
Network	Yes	<p>The network over which Rendezvous sends messages. In most cases you can leave this field empty. For more information about the network parameter, see <i>TIBCO Rendezvous Administration</i>.</p> <p>Default is an empty string (defined in Global Variables).</p>
Daemon	Yes	<p>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 <i>TIBCO Rendezvous Concepts</i>.</p> <p>Default is <code>tcp:7500</code> (defined in Global Variables).</p>

Configuration for JMS Channels

JMS Channel Properties

Field	Global Var?	Description
ProviderURL	Yes	The URL at which TIBCO BusinessEvents can contact the Enterprise Message Service server. Example: <code>tcp://localhost:7222</code>
UserName	Yes	A valid user name for the Enterprise Message Service server.
Password	Yes	The password assigned to the user name specified in UserName , for accessing the Enterprise Message Service server.
IsTransacted	Yes	Accepts <code>true</code> or <code>false</code> . Specify <code>true</code> if the session has transaction semantics. Specify <code>false</code> if it has non-transaction semantics. For more information about the <code>IsTransacted</code> property, see TIBCO Enterprise Message Service documentation.
ClientID	Yes	The unique client ID of the connection.

Configuration for StreamBase Channels

StreamBase Channel Properties

Field	Global Var?	Description
StreamBase Server URI	Yes	The URI at which TIBCO BusinessEvents can contact the StreamBase server Example: <code>sb://localhost:10000/</code>
UserName	Yes	A valid user name for the StreamBase server. Note: This field is required only when the basic authentication is enabled at the StreamBase server.
Password	Yes	The password assigned to the user name specified in UserName , for accessing the StreamBase server. Note: This field is required only when the basic authentication is enabled at the StreamBase server.

Configuration for ActiveSpaces Channels

ActiveSpaces Channel Properties

Field	Global Var?	Description
Metaspace Name	No	<p>The name of a particular metaspace instance in TIBCO ActiveSpaces that the channel must connect to.</p> <p>The metaspace must be created and initialized before the channel can use it at run time.</p>
Discovery URL	No	<p>Specifies how a metaspace instance discovers the current metaspace members. Multicast discovery can use either PGM - Pragmatic General Multicast or RV - TIBCO Rendezvous protocol.</p> <p>If using PGM protocol, the multicast URL is expressed in the following format:</p> <pre>tibpgm://destination port/interface;discovery group address/optional transport arguments, where</pre> <ul style="list-style-type: none"> <i>destination port</i> specifies the destination port used by the PGM transport. If not specified, the default value of 7888 is used. <i>interface;discovery group address</i> specifies the address of the interface to be used for sending discovery packets, and the discovery group address to be used. If not specified, it defaults to the default interface and discovery address, 0.0.0.0;239.8.8.8. <i>optional transport arguments</i> specifies a semicolon-separated list of optional PGM transport arguments. By default, the PGM transport is tuned to provide the best performance according to the most common deployment architectures. The values of those optional arguments should be changed only when necessary and with care, since any inappropriate values could easily result in degraded performance of the product. <p>If using TIBCO Rendezvous, the multicast URL is expressed in the following format:</p> <pre>tibrv://service/network/daemon, where</pre> <ul style="list-style-type: none"> <i>service</i> specifies the TIBCO Rendezvous service number (UDP port) that will be used. If not specified, the value defaults to 7889. <i>network</i> specifies the interface and discovery group address that will be used to send Rendezvous discovery packets. The format is: <code>interface; discovery_group_address</code> If not specified, the default interface and discovery group address will be used. (<code>tibrv://7889/;239.8.8.9/</code>). <i>daemon</i> specifies where to find the Rendezvous daemon. If not specified, it will try to connect to a local daemon on port 7500.

Field	Global Var?	Description
ListenUrl	No	<p>The discovery mechanism is based on pure TCP.</p> <p>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.</p> <p>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).</p>
RemoteListenUrl	No	<p>This field is used to configure TIBCO ActiveSpaces Channel as a remote-discovery proxy. In this case, any remote client can connect to an ActiveSpaces metaspace via the ActiveSpaces Channel node.</p>
EnableSecurity	No	<p>Enables security for the ActiveSpaces channel when selected.</p> <p>Note: Some fields are activated only for the specific security role or authorization policy.</p>
SecurityRole	No	<p>Security role of a node for the secure ActiveSpaces channel in the metaspace. The values are:</p> <ul style="list-style-type: none"> • Controller is dedicated to enforcing a security domain's defined security behavior for a metaspace associated with the security domain. Security domain controllers are the only discovery nodes in a metaspace. • Requestor just requires access to the data in the data grid, such as a seeder or a leech, and which need to be authorized by a controller. Requesters can never be used a discovery nodes. <p>The controller nodes are configured with a security policy file. The requester nodes provide a token file and additional credentials to the controller for authentication. The controller performs authentication as defined in its policy file and using the credentials provided by the requester.</p> <p>If the Controller option is selected, then the following fields become active:</p> <ul style="list-style-type: none"> • Identity Password • PolicyFile <p>If the Requestor option is selected, then the following fields become active:</p> <ul style="list-style-type: none"> • Identity Password • TokenFile • Credential
Identity Password	No	<p>The password for the identity key in the security policy file.</p>

Field	Global Var?	Description
PolicyFile	No	Absolute path to the policy file which contains the security settings that the controller node enforces. It is generated using the as-admin utility.
TokenFile	No	Absolute path to the token file which is used by requestor to connect to a metaspace whose security is defined in the policy file.
Credential	No	<p>Authentication policy to be used for authentication as specified in the policy file. The values are:</p> <ul style="list-style-type: none"> • USERPWD - A user name and password based authentication is used. It activates the following fields: <ul style="list-style-type: none"> – Domain – Username – Password • X509V3 - The authentication source is an LDAP configured with certificate based authentication. It activates the following fields: <ul style="list-style-type: none"> – KeyFile – PrivateKey
Domain	No	Domain name for system based user authentication.
Username	No	User name for LDAP and system based authentication.
Password	No	Password for LDAP and system based authentication.
KeyFile	No	The absolute path for a file containing the key to use for LDAP with the certificate based authentication.
PrivateKey	No	The password for the identity key in the LDAP identity file specified in KeyFile

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.

Destination Resource Common Properties

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.
Description	No	Short description of the resource.
Default Event	No	The event to be created from incoming messages unless otherwise specified. For 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.
Include Event Type	Yes	<i>(Rendezvous, JMS, and HTTP only)</i> Specifies when to suppress the original behaviour of including <code>_ns_</code> and <code>_nm_</code> fields during serialization and deserialization. The values are: <ul style="list-style-type: none"> • When Serializing and Deserializing: always include <code>_ns_</code> and <code>_nm_</code> fields. • Only when Serializing: for outgoing JMS messages <code>_ns_</code> and <code>_nm_</code> fields are set in the JMS message headers. For incoming JMS messages <code>_ns_</code> and <code>_nm_</code> fields are ignored. • Only when Deserializing: the <code>_ns_</code> and <code>_nm_</code> fields are suppressed in the outgoing message but are used in incoming messages. • Never: the <code>_ns_</code> and <code>_nm_</code> fields are suppressed in incoming as well as outgoing messages.

Configuration for TIBCO Rendezvous Destinations

See TIBCO Rendezvous documentation for more details on these settings.

TIBCO Rendezvous Destinations Configuration Properties

Field	Global Var?	Description
Subject	Yes	The TIBCO Rendezvous subject for incoming and outgoing messages.
RVCM Pre Registration	Yes	For TIBCO Rendezvous certified message publishers, specify pre-registered listener names as a comma separated list.

Field	Global Var?	Description
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.

JMS Destinations Configuration Properties

Field	Global Var?	Description
Queue	Yes	Specifies whether the destination is a queue or a topic. Select the check box if the destination is a queue. If the destination is a topic, do not select it.
Name	Yes	Required. The name of the queue or topic. TIBCO BusinessEvents ignores JMS destinations with null or empty-string queue or topic names. It logs an error message for the ignored destinations. If a JMS message is sent out through an ignored destination, TIBCO BusinessEvents 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.

Field	Global Var?	Description
DeliveryMode	No	<p>The delivery mode property instructs the server concerning persistent storage for the message. Select one of the following:</p> <ul style="list-style-type: none"> <code>PERSISTENT</code> (default) — In JMS message headers this is represented by the code 2. <code>NON-PERSISTENT</code> — In JMS message headers this is represented by the code 1. <code>RELIABLE</code> — This value is an extension to the standard, used in TIBCO Enterprise Message Service. In message headers this is represented by the code 22. <p>You can also set a delivery mode in an event.</p>
AckMode	Yes	<p>The acknowledgement mode. See JMS Message Acknowledgement Modes for more details on the various modes.</p> <p>The default value is <code>EXPLICIT_CLIENT_ACKNOWLEDGE</code>.</p>
Priority	No	<p>The message priority. Takes a numerical value between 0 and 9. Larger numbers represent higher priority.</p> <p>You can also set a priority in an event.</p> <p>The default value is 4.</p>
TTL	Yes	<p>The length of time that the message will live (in milliseconds) before expiration. If set to 0, the message does not expire.</p> <p>You can also set a TTL (<code>JMSExpiration</code>) in an event.</p> <p>The default value is 0.</p>
DurableSubscriberName	Yes	<p>For destinations that are JMS Topics, if you provide a <code>DurableSubscriberName</code>, 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.</p> <p>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 <code>DurableSubscriberName</code> string.</p> <p>The default value is: <code>%%EngineName%%:%%SessionName%%:%%ChannelURI%%:%%DestinationName%%</code>.</p>

Configuration for Local Destinations

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

Local Destinations Configuration Properties

Field	Global Var?	Description
Size	No	<p>The maximum number of events to be held in the queue. The default is zero (0), which allows unlimited events in the queue.</p>

Field	Global Var?	Description
TimeOut	No	<p>Time to wait when sending an event to this local destination. The values are:</p> <ul style="list-style-type: none"> • -1 - Waits indefinitely • 0 - Does not wait • >0 - Waits for the specified number of milliseconds <p>The default value is -1.</p>

Configuration for HTTP Destinations

HTTP Destinations Configuration Properties

Field	Global Var?	Description
Is JSON Payload	No	Specifies whether payload is JSON
Is Page Flow	No	Enables Action Rule Function based approach. If selected, 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 check box is selected.
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 check box is selected.

Configuration for StreamBase Destinations

StreamBase Destinations Configuration Properties

Field	Global Var?	Description
Stream Name	No	The name of the input or output stream from the StreamBase application
Client Type	No	<p>Specifies whether the connection is used as an input (dequeue) client or an output (enqueue) client. The values are:</p> <ul style="list-style-type: none"> • Dequeueer • Enqueueer

Field	Global Var?	Description
Filter Predicate	No	A StreamBase expression that is used to filter the incoming messages. For instance, <code>BidPrice > 100</code> . This field is active only if Client Type is Dequeueer.
Enable Buffering	No	Specifies whether to activate buffering for an enqueue client. Activating buffering can improve performance when there are a large number of enqueue operations. This field is active only if Client Type is Enqueueer.
Buffer Size	No	The number of tuples to buffer before the enqueue operation. This field is active only if Enable Buffering is selected.
Flush Interval (ms)	No	Time interval (in milliseconds) to wait before flushing the enqueue buffer. This field is active only if Enable Buffering is selected.

JMS Channels

This chapter provides additional information about working with JMS channels.

Selecting a JMS Serializer

When working with the JMS channels, choose the serializer that handles the JMS message types that will be sent to the destination you are configuring.

Which Serializers to Use for JMS Message Types

Message Type	BytesMessage Serializer/ UtfBytesMessage Serializer	TextMessage Serializer	MessageWithNoBody (1)
Message	Supported	Supported	Supported
BytesMessage	Supported	(N/A)	Supported
MapMessage	Supported	Supported	Supported
ObjectMessage	Not Supported	Not Supported	Not Supported
StreamMessage	Not Supported	Not Supported	Not Supported
TextMessage	(N/A)	Supported	Supported

(1) MessageWithNoBody corresponds to the BasicMessageSerializer option in TIBCO BusinessEvents Studio.

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

Payload Handling

All serializers support reading and writing application header properties and JMS header properties.

The difference between the serializers is in how they handle payloads.

BytesMessageSerializer

The BytesMessageSerializer decodes the body as a sequence of bytes and parses them to create an XML structure according to the payload definition in the event.

The BytesMessageSerializer is used to receive JMS messages coming in as a stream of uninterrupted bytes in the message body.

Typically, BytesMessageSerializer is a low level serializer and can have performance implications. Choose the serializer for a defined message type (MapMessage, TextMessage, and so on) that most closely matches the expected usage.

For incoming messages of type JMS BytesMessage, the serializer converts the message bodies to event payloads. The payloads are XML String type, but are not human-readable. However, you can access them using XPath functions. For outgoing events, the serializer converts XML payloads to JMS BytesMessage message bodies.

The `BytesMessageSerializer` class is the default serializer.

UtfBytesMessageSerializer

The `UtfBytesMessageSerializer` is similar to the `BytesMessageSerializer` except that it serializes the payload using `writeUTF()` instead of `writeBytes()`, and deserializes the payload using `readUTF()` instead of `readBytes()`.

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

TextMessageSerializer

The `TextMessageSerializer` decodes the text from the message as an XML string.

For incoming messages, the `TextMessageSerializer` serializer converts JMS `TextMessage` messages using XPath functions. For outgoing events, the serializer converts XML payloads to JMS `TextMessage` messages.

MessageWithNoBody

The `MessageWithNoBody` serializer does not serialize or deserialize the payload.

This serializer corresponds to the `BasicMessageSerializer` option in TIBCO BusinessEvents Studio.

For outgoing events, the serializer converts the payload to messages of type `Message`.

See JMS Header Properties in Incoming and Outgoing Messages in *TIBCO BusinessEvents Developer's Guide*.

Creating Unique JMS DurableSubscriber Name Properties

For destinations that are JMS Topics, if you provide a `DurableSubscriberName` 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 `DurableSubscriberName` 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 > AgentClassName > Max Active** is set to 1). Instead, use queue destinations.

The value of the `DurableSubscriberName` 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. For details see [Variables to Use with DurableSubscriberName](#).

Variables to Use with DurableSubscriberName

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

DurableSubscriberName Variables

Variable	Description
<code>%%EngineName%%</code>	The name of the TIBCO BusinessEvents engine. The name used is established using a series of checks. See <i>Determining the Engine Name in TIBCO BusinessEvents Administration</i> for details.
<code>%%SessionName%%</code>	The name of the agent class that is associated with the durable subscriber. Agent classes are defined in the CDD resource. See <i>Collections, Agent Classes, and Processing Units in TIBCO BusinessEvents Configuration Guide</i> for details.
<code>%%ChannelURI%%</code>	The path to the channel within the TIBCO BusinessEvents project: <code>/folder/.../channel_name</code>
<code>%%DestinationName%%</code>	The name of the TIBCO BusinessEvents destination, within the channel specified in <code>%%ChannelURI%%</code> .

JMS Message Acknowledgement Modes

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

JMS Message Acknowledgement Modes

Mode	Description
<code>AUTO_ACKNOWLEDGE</code>	Specifies that the session is to automatically acknowledge consumer receipt of messages when message processing has finished.
<code>CLIENT_ACKNOWLEDGE</code>	Specifies that the consumer is to acknowledge all messages that have been delivered so far by the session. When using this mode, it is possible for a consumer to fall behind in its message processing and build up a large number of unacknowledged messages. See Using CLIENT_ACKNOWLEDGE Mode with Websphere MQ and Cache-Aside for required configuration for Websphere MQ when cache-aside database write strategy is used.
<code>DUPS_OK_ACKNOWLEDGE</code>	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.

Mode	Description
EXPLICIT_CLIENT_ACKNOWLEDGE (TIBCO Proprietary)	<p>TIBCO Enterprise Message Service extension to JMS acknowledge modes. This is the default acknowledgement mode.</p> <p>EXPLICIT_CLIENT_ACKNOWLEDGE is like CLIENT_ACKNOWLEDGE except it acknowledges only the individual message, rather than all messages received so far on the session.</p> <p>One example of when EXPLICIT_CLIENT_ACKNOWLEDGE would be used is when receiving messages and putting the information in a database. If the database insert operation is slow, you might want to use multiple application threads, all doing simultaneous inserts. As each thread finishes its insert, it can use EXPLICIT_CLIENT_ACKNOWLEDGE to acknowledge only the message that it is currently working on.</p>
EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE (TIBCO Proprietary)	<p>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.</p>
NO_ACKNOWLEDGE (TIBCO Proprietary)	<p>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.</p> <p>Note:</p> <ul style="list-style-type: none"> • Sessions created in NO_ACKNOWLEDGE receipt mode cannot be used to create durable subscribers. • Also, queue receivers on a queue that is routed from another server are not permitted to specify NO_ACKNOWLEDGE mode.

Using CLIENT_ACKNOWLEDGE Mode with Websphere MQ and Cache-Aside

The cache-aside database write strategy is multi-threaded. However, when Websphere MQ messages are sent using CLIENT_ACKNOWLEDGE_MODE, each message must be handled from start to finish using a single thread. To address this issue, follow these steps:

Procedure

1. Define the destination using Caller's Thread in the Threading Model setting (in the **CDD Collections** tab or **Agent Classes** tab).
2. To ensure sequential operations set the following property in the CDD file at the appropriate level:

```
Agent.agentClassName.enableParallelOps=false
```

Setting this property to false means that all post-RTC operations are done on a single thread.

When JMS Messages are Acknowledged

When TIBCO BusinessEvents acknowledges JMS messages depends on the JMS acknowledgement mode, time to live (TTL) setting, and object management (OM) type, as shown in the table below.

When JMS Messages are Acknowledged

JMS Acknowledgement Mode	OM Type	Acknowledged
AUTO_ACKNOWLEDGE DUPS_OK_ACKNOWLEDGE		On receipt
NO_ACKNOWLEDGE		Never
CLIENT_ACKNOWLEDGE EXPLICIT_CLIENT_ACKNOWLEDGE EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE	Cache	Post RTC
	In Memory	When retracted (when the event is deleted (using the <code>Event.consumeEvent</code> 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 can 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 [JMS Header Field Names](#). You only have to configure event properties for those fields that you want to use. Incoming JMS message header properties will then populate the corresponding TIBCO BusinessEvents event properties.

Setting JMS Header Properties in Outgoing JMS Messages Using Event Properties

Similarly outgoing JMS message header properties will be populated by the corresponding TIBCO BusinessEvents event properties.

Note that the `JMSMessageID` and `JMSTimeStamp` properties are generated when the message is sent. You cannot set these properties manually.

See [How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages](#) for special handling of the `JMSReplyTo` header.

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



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

See [JMS Header Field Names](#) for details 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.



You can use global variables in all the advanced properties. For more details, see [Global Variables](#).

JMS Header Field Names

Field name	Type	Description
<code>JMSDestination</code>	Object	The destination (queue or topic) to which the message is sent.
<code>JMSDeliveryMode</code>	Integer	<p>The delivery mode specified by the sender. This property instructs the server concerning persistent storage for the message. Value can be:</p> <ul style="list-style-type: none"> 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. <p>The integer values are interpreted as the text names of delivery modes.</p> <p>You can also set a delivery mode for a destination. See Setting Certain Header Properties in Destinations.</p>

Field name	Type	Description
JMSExpiration	Long	<p>The length of time that the message will live (in milliseconds) before expiration. If set to 0, the message does not expire.</p> <p>You can also set an expiration (TTL) for a destination. See Setting Certain Header Properties in Destinations .</p>
JMSPriority	Integer	<p>The message priority, a numerical value between 0 and 9. Larger numbers represent higher priority.</p> <p>You can also set a priority for a destination. See Setting Certain Header Properties in Destinations .</p>
JMSMessageID	String	<p>An ID that uniquely identifies each message sent by a provider.</p> <p>A generated value overrides any value set in the corresponding event property.</p>
JMSTimestamp	Long	<p>The time when the message was handed off to a provider to be sent. The message may be sent later than this timestamp value.</p> <p>A generated value overrides any value set in the corresponding event property.</p>
JMSCorrelationID	String	<p>A correlation ID that can be used to link messages. For example, you can link a response message to a request message. Optional.</p>
JMSReplyTo	String	<p>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.</p> <p>Optional.</p> <p>Note :</p> <p>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.</p> <p>See How TIBCO BusinessEvents Sets the JMSReplyTo Header in Outgoing Messages .</p>
JMSType	String	<p>The message type identifier, if used by the provider.</p>
JMSRedelivered	Boolean	<p>If this field is set, it is possible that the message was delivered to the client earlier but not acknowledged at that time.</p>

JMS Sender Session Pooling

When working in the non-transacted mode, you can create a pool of JMS sender sessions for each JMS channel in the project.

Each thread, that executes any of the JMS sending functions (such as, `sendEvent`, `replyEvent`, `routeTo`, and so on), takes a session from the pool. After utilizing the session and on completion of the send task, the thread returns the session to the pool. If max pool size is reached, threads wait for a session to become available and till then, do not create any new session.

By default, the pool is disabled and the sending functions share a single session per channel. To activate the pool set the value of the `be.channel.jms.sender.session.pool.maxsize` property, which identifies the maximum pool size, with minimum allowed value as 1.

SOAP over JMS Support in WSDL Import and Export

SOAP and JMS are supported in WSDL Import and Export for events, channels, rules, rulefunctions and catalog functions.

WSDL Import

The WSDL Import wizard generates resources for handling SOAP and JMS bindings.

Events

One Event type named `/Events/SoapJms`. This event inherits from `SOAPEvent` and provides a set of properties that may be useful for SOAP and JMS. Events are operation specific events and they are imported similarly as for the SOAP and HTTP WSDL Import.

Channels

A JMS channel by default is located at `/Channels/SoapJms` for each different set of parameters, with one shared JMS connection resource.

By default the resource is in the Transport folder of the imported service folder. One destinations for each different set of destination parameters is designated.

By default the destination is named `destinationName`, which uses the `serializerSoapMessageSerializer` and the default event type `/Event/SoapJms`. See the `RegisterSoapEventUri` function below, which allows the deserialer know how to deserialize message into operation specific events instead of using the destination default.

Rules

There is one rule for each operation, which is imported similarly as for the SOAP and HTTP WSDL Import.

By default, the rule replies with a SOAP fault stating that the operation is not implemented.

RuleFunctions

There is one RuleFunction for each operation, which is imported similarly as for the SOAP and HTTP WSDL Import.

If desired, the RuleFunction can be used as a preprocessor function.

The one RuleFunction is named, `RegisterSoapEventUri`.

By default, it is avaiable in the Transport folder of the imported service folder. This registers in the JMS `SoapMessageSerializer`, and event type for each operation that is reachable by a SOAP and JMS binding. It should be invoked as a startup function.

Catalog Functions

A serializer and deserializer for SOAP and JMS has been added to the JMS destinations. This serializer handles the conversion between the JMS messages and SoapEvent events. Two help catalog functions are provided as well:

Catalog Function	Description
<code>SOAP.registerEventUri()</code>	<p data-bbox="898 348 1458 443">Voids <code>registerEventUri(String event Uri, String destinationUri, String serviceName, String soapAction, String Preprocessor)</code></p> <p data-bbox="898 464 1485 646">Associates an event type to a given input destination, target service, and SOAP action. This is used by the SOAP and JMS deserializer to generate events of specific types (eventUri) instead of using the default event type resolution mechanism.</p> <ul data-bbox="898 680 1477 1182" style="list-style-type: none"> • eventUri: String path of an event type in the project. • destinationUri: String path of the TIBCO BusinessEvent destination receiving the message. If this is empty, it matches all of the destinations. • serviceName: String name of the target service declared in the message received. If this is empty, it matches all of the service names. • soapAction: String value of the soapAction parameter in the message received. • Preprocessor: String path of the preprocessor in the project.
<code>SOAP.newCorrelationId()</code>	<p data-bbox="898 1226 1270 1289">String <code>newCorrelationId(String responseEventUri)</code></p> <p data-bbox="898 1310 1458 1493">Creates a new correlation ID for use when sending a message in a request and response case. This is used by the SOAP and JMS deserializer to generate a response event of a specific type, instead of using the default event type resolution mechanism.</p> <ul data-bbox="898 1526 1445 1633" style="list-style-type: none"> • responseEventUri: String path of an event type in the project. • returns: String correlation ID.

In the context of TIBCO BusinessEvents, the two tasks of a message serializer are deserialization from the transport-level message to an event and serialization from an event to the transport-level message.

WSDL Export

The WSDL Export was implemented to support SOAP and JMS, but it has some limitations. The following include its limitations:

1. It uses the TIBCO format and not W3C.
2. It relies on the presence of an event registration start up function with a body that looks like something that is generated during WSDL Import.
3. It relies on a folder structure that is similar to what is generated during import.
4. It does not export all of the possible information, for example, the reply-to names.



The import of a WSDL, then an export to another WSDL should result in similar WSDLs, though not identical.

HTTP Channels

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

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

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



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

SOAP Channels

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

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

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

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

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

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

See [TIBCO BusinessEvents for SOAP Server and Client Configuration](#) for more details.

Adding an HTTP Connection

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

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

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

To configure an HTTPS (Secure) Connection

Select the **SSL** check box, 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 [For SSL Only — Add an Identity Resource](#)).



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

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

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

Adding an Identity Resource — For SSL Only

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

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

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

Adding an HTTP Channel

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

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

Adding a Destination

Add a destination to the channel in the usual way. The fields for destination differs based on different approaches.

To follow the action rule function based approach

Select the **Is Page Flow** check box 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`.

Creating Events as Needed and Setting Default Destinations

Skip this step for action rule function based approach.

In the event based approach

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

Configuring Rules and Rule Functions

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

In the action rule function based approach

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

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

In the event based approach

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

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

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

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

In the CDD Configure the Processing Unit

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

Working with HTTP Requests

An HTTP request is mapped to an event.

HTTP requests are parsed and 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.

For instructions on setting up a channel, see *TIBCO BusinessEvents Developer's Guide*, Adding a Channel.

HTTP Requests and Events Mapping with the REST Serializer

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

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

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

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

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

JSON Payload

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

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

- Select the **Is JSON Payload** while creating a new destination. See [Adding Destination to a Channel](#) and [Destination Resource Reference](#).
- The event mapped to the destination, should have the destination URL in its default destination field.
- Regular payload validation rules apply as they would for default XML based events., that is, if the namespace check fails, the payload is not set. Namespace for an XML element is part of its root element, while for JSON it is part of "attributes" node under "type".
- After a JSON payload is deserialized to an event, it is used the way an XML based event is used.



XML and JSON Payload Examples

XML Request

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

JSON Request

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

XML Response

```
<Address>
  <street>243 Buena Vista Avenue</street>
  <city>Sunnyvale</city>
  <zip>94086</zip>
</Address>
```

JSON Response

```
{"Address" :
  { "street" : "243 Buena Vista Avenue",
    "city" : "Sunnyvale",
    "zip" : "94086" }
}
```

Payload Without Wrapping Elements

You can set the payload schema to send the JSON document with no wrapping elements. To exclude the wrapper and the internal attribute elements, set the `be.http.json.rootElement.ignore` property to `true`. The default value of the property is `false`.

Action Rule Function Based Approach

HTTP request can be processed in TIBCO BusinessEvents Studio without mapping to an event.

An HTTP request can be processed in two ways:

HTTP Request Processing

The TIBCO BusinessEvents server parses the HTTP request received from client. The server extracts the request header and identifies the request URI of the request. The request URI should map to the context specified in the destination defined for the channel. In this case TIBCO BusinessEvents server looks for the destination having the same context path as the requestURI.

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

See [Sample Code for Action Rule Function](#).

Deploying Multiple Web Applications

You can deploy multiple web applications on single channel. Specify the *.WAR file or a valid J2EE web application folder under the Web Applications section in the advanced tab. Specify the context URI and resource path for the web application. Resource path identifies the actual path of web application resources. In case of a *.WAR file resource path is the location of the war file, and in case of the J2EE web application folder the resource path is the location of the base folder of the web application.

Sample Code for Action Rule Function

Action rule function defined in the destination uses catalog functions to get required data and parameters from the HTTP request. The rule function processes these parameters and creates a response, similar to a preprocessor. TIBCO BusinessEvents supports all HTTP methods stated in the HTTP 1.1 specification for Action Rule Function based approach.

```
void rulefunction RuleFunctions.Callback {
    attribute {
        validity = ACTION;
    }
    scope {
        Object asyncContextObject;
    }
    body {
        //getting servlet request and response objects
        Object servletRequest = HTTP.Servlet.getServletRequest(asyncContextObject);
        Object servletResponse = HTTP.Servlet.getServletResponse(asyncContextObject);
        System.debugOut("##Servlet request method : " +
HTTP.Servlet.Request.getMethod(servletRequest));
        System.debugOut("##Servlet request content : " +
HTTP.Servlet.Request.getRequestContent(servletRequest));
        System.debugOut("##Servlet request Requester Address : " +
HTTP.Servlet.Request.getRequestorAddress(servletRequest));
        System.debugOut("##Servlet request Request URI : " +
HTTP.Servlet.Request.getRequestURI(servletRequest));
        //getting parameters
        String[] params = HTTP.Servlet.Request.getRequestParameters(servletRequest);
        for(int i=0;i<params@length;i++)
        {
            System.debugOut("## Servlet request Parameters : " +
HTTP.Servlet.Request.getRequestParameter(servletRequest,params[i]));
        }
        //getting headers
        System.debugOut("## Servlet request Header Accept : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept"));
        System.debugOut("## Servlet request Header Accept-Encoding : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept-Encoding"));
        System.debugOut("## Servlet request Header Accept-Language : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept-Language"));
        System.debugOut("## Servlet request Header Accept-Charset : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Accept-Charset"));
        System.debugOut("## Servlet request Header Connection : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Connection"));
    }
}
```

```

        System.debugOut("## Servlet request Header User-Agent: " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"User-Agent"));
        System.debugOut("## Servlet request Header Content-Length : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Content-Length"));
        System.debugOut("## Servlet request Header Content-Type : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Content-Type"));
        System.debugOut("## Servlet request Header Host : " +
HTTP.Servlet.Request.getRequestHeader(servletRequest,"Host"));
        HTTP.Servlet.Response.setResponseHeader(servletResponse, "Content-type",
"text/plain");
        HTTP.Servlet.Response.setResponseContent(asyncContextObject, "response",
true);
    }
}

```

Setting Up Fault Tolerance for the HTTP Channel

Use the Apache HTTP server (“httpd”) and mod_jk to setup fault tolerance to the TIBCO BusinessEvents HTTP channel and a third party fronting server. These properties are used in addition to sharing the cache server between the nodes in the cluster.

Procedure

1. Create two new property files with suitable names. For example, `Ajp_1.properties` and `Ajp_2.properties`.
2. Add the ajp connector port property in the two newly created property files.
In `Ajp_1.properties`:
`<channel path>.ajp.connector.port=8011`
In `Ajp_2.properties`:
`<channel path>.ajp.connector.port=8012`
3. Specify the new property files as part of the TRA file of individual engines. Ensure that there is one properties file per engine.
`java.property.be.channel.external.config.file=<location of properties file>`
For example,
`java.property.be.channel.external.file=/tibco/app/conf/Ajp_1.properties`
`java.property.be.channel.external.file=/tibco/app/conf/Ajp_2.properties`
4. Configure the Apache HTTP server (“httpd”) as a fronting webserver.
 - a) Add the `worker.properties` file in Apache httpd under the `conf` folder.
 - b) Set the Apache http for load balancing in the `worker.properties` file.
 - c) Configure two workers to listen to the above specified ajp ports.
 - d) Disable one worker (for example, worker 2) by default, so that no requests are sent to it.
 - e) Configure another worker (worker1), so that in case of a failure, it redirects all incoming requests to the disabled worker (worker2).

The `worker.properties` file should contain the following properties:

```

worker.list=balancer
worker.balancer.type=lb
worker.balancer.sticky_session=0
worker.balancer.balance_workers=worker1,worker2
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8011
worker.worker1.redirect=worker2
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8012
worker.worker2.activation=disabled

```

Worker1, in case of a failure, now becomes the active worker.

5. Add Tomcat connectors `mod_jk.so` under the `modules` folder.

You can download `mod_jk` from the following location:

<http://tomcat.apache.org/download-connectors.cgi>

6. Configure Apache `httpd` to load this module.
 - a) Redirect all requests to the load balancer defined earlier.
 - b) Add the following snippet to `httpd.conf`, after completion of all the basic module load statements:

```
LoadModule jk_module modules/mod_jk.so
<IfModule jk_module>
JkWorkersFile conf/worker.properties
JkLogFile logs/mod_jk.log
JkLogStampFormat "[%b %d %Y - %H:%M:%S] "
JkRequestLogFormat "%w %V %T"
JkLogLevel info
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
<Location//WEB-INF/>
deny from all
</Location>
JkMount /* balancer
</IfModule>
```

7. Start Apache server; Start both the instances of be-engines running the HTTP based project.
8. Open the browser and client server.
 - a) Enter the url `http://localhost/<context-path>`, without any port.
 The request is now routed to the active worker (`worker1`) only, since the other worker (`worker2`) is disabled. You can verify through logs that logs are being created for one be-engine only.
9. Shutdown the active be-engine associated with the active worker (`worker1`).
 All subsequent requests from the client now route to other worker (`worker2`). To verify, check the logs in the corresponding be-engine instance.

HTTP Channel Advanced Configuration Settings

The following settings configure the internal Tomcat HTTP server used by the channel. They are set in the **Advanced** tab of the HTTP channel resource editor.



You can use global variables in all the advanced properties. For more details, see [Global Variables](#).

HTTP Channel Advanced Configuration Settings (Sheet of)

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.
Access Log Valve	

Property	Notes
	<p>Enables access logging on HTTP channels.</p> <p>In the advanced properties of the channel, if <code>DebugRequestInfo</code> is enabled, it enables other two debug fields <code>DebugLogFolder</code> and <code>DebugLogPattern</code>. <code>DebugLogFolder</code> specifies the location of generating the log file. Default is a new logs folder in the currently running folder. It uses the default log pattern. You can change the log pattern using Tomcat documentation.</p>
Connection Timeout (msec)	
	<p>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.</p> <p>The value 0 means no timeout.</p> <p>The default is 60000.</p>
Accept Count	
	<p>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.</p> <p>The value -1 means that a default of 10 is set.</p> <p>The default is -1.</p>
Socket Output Buffer Size	
	<p>The size in bytes of the buffer to be provided for socket output buffering.</p> <p>The value -1 means that the use of a buffer is disabled.</p> <p>The default is 9000.</p>
Max Processors	
	<p>The maximum number of simultaneous requests that can be handled by the HTTP server.</p> <p>Set to one of these:</p> <ul style="list-style-type: none"> • 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. <p>The default is -1.</p>
Connection Linger	
	<p>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.</p> <p>To disable use of socket linger, set to -1.</p> <p>The default is -1.</p>
Enable DNS Lookups	

Property	Notes
	<p>Set to <code>true</code> if you want the calls to <code>request.getRemoteHost()</code> to perform DNS lookups and return the actual host name of the remote client. Set to <code>false</code> to skip the DNS lookup and return the IP address as a string instead (thereby improving performance).</p> <p>By default, DNS lookups are disabled.</p>
TCP No Delay	
	<p>If the check box is selected, the <code>TCP_NO_DELAY</code> option is not set on the server socket.</p> <p>If the check box is selected, the <code>TCP_NO_DELAY</code> option is set on the server socket. Using <code>TCP_NO_DELAY</code> improves performance under most circumstances.</p> <p>The default is selected.</p>
Compression	
	<p>Compression can be used to save server bandwidth. Uses HTTP/1.1 GZIP compression.</p> <p>The compression option set here is used together with the MIME types shown in the Compressible Mime Types setting. Allowable values are:</p> <p>off Disables compression. Values in the Compressible Mime Types setting are ignored.</p> <p>on Enables compression</p> <p>force Forces compression.</p> <p>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).</p> <p>The default is off.</p>
Use Body Encoding for URI	
	<p>If selected, the encoding specified in the <code>contentType</code> HTTP header is used to decode the request URI.</p> <p>If not selected, the value in URI Encoding field (or its default value) is used.</p> <p>If the URI Encoding setting is specified, then the Use Body Encoding for URI checkbox is ignored.</p> <p>For an example showing <code>contentType</code>, see Working with Outgoing SOAP Messages (Event Payloads).</p> <p>The default this option is not selected.</p>

Property	Notes
URI Encoding	<p>Specifies the character encoding used to decode the request URI. If not specified, UTF-8 is used.</p> <p>If specified, the Use Body Encoding for URI setting is ignored.</p> <p>The default is UTF-8.</p>
Max KeepAlive Requests	<p>The maximum number of HTTP requests that can be pipelined until the connection is closed by the server.</p> <p>A value of 1 disables HTTP/1.0 keep-alive, as well as HTTP/1.1 keep-alive and pipelining.</p> <p>A value of -1 allows an unlimited amount of pipelined or keep-alive HTTP requests.</p> <p>The default value is -1.</p>
Max HTTP Header Size	<p>The maximum size of the request and response HTTP header, specified in bytes.</p> <p>The default is 4096, that is 4 KB.</p>
Max HTTP Post Size	<p>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.</p> <p>The default is 2097152, that is, 2 MB.</p>
Max HTTP Save Post Size	<p>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.</p> <p>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.</p> <p>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.</p> <p>A value of -1 means no limit.</p> <p>A value of 0 disables the saving of POST data during authentication.</p> <p>The default is 4096, that is 4 KB.</p>
Protocol	

Property	Notes
	<p>Specifies the HTTP protocol to use. Options are as follows:</p> <p><code>http</code> HTTP protocol.</p> <p><code>https</code> For secure communications (SSL).</p> <p><code>memory</code> Memory protocol.</p> <p><code>ajp</code> Apache JServ Protocol, a binary protocol.</p> <p>Default is <code>http</code>.</p>
Max Spare Threads	
	<p>The maximum number of unused request processing threads that are allowed to exist until the thread pool starts stopping the unnecessary threads.</p> <p>The default is 50.</p>
Min Spare Threads	
	<p>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.</p> <p>The default is 4.</p>
Compressible Mime Types	
	<p>This is a comma-separated list of MIME types for which HTTP compression may be used.</p> <p>This setting works with the Compression setting See notes for that setting for more details.</p> <p>The default value is <code>text/html , text/xml , text/plain</code>.</p>
Restricted User Agents	
	<p>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, <code>Mozilla/4.0</code>.</p>
Document Root	
	<p>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.</p> <p>Tip</p> <p>To enable clients of a TIBCO BusinessEvents HTTP server to view the imported concrete WSDL, provide the URL to the document root folder.</p> <p>No default value.</p>
Document Page	

Property	Notes
	The name of the default static HTML file stored in the document root. This setting is used by internal TIBCO BusinessEvents applications and may have limited application for other purposes.
SSL Server: Key Manager Algorithm	
	The key manager algorithm for the SSL Service provider. The default is SunX509.
SSL Server: Trust Manager Algorithm	
	The trust manager algorithm for the SSL Service provider. Ensure that a provider is available for any algorithm other than PKIX. The default is PKIX.
SSL Server: Protocols	
	The SSL protocols that can be enabled on the server. Add each protocol using a comma-separated list. The default protocols are the ones supported by the SSL Provider. (SSLv3 and TLSv1 are the most widely supported.) Leave blank to use the default protocols for your SSL provider.
SSL Server: Ciphers	
	The cipher suites (SSL protocols) used by the server. Add each suite name on a separate line. The following cipher suites are supported: <pre> 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 </pre> 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
```

Defining the `HttpStatusCode` property to the event properties explicitly sets the HTTP status code. `HttpStatusCode` can be an integer or a string, and is not case-sensitive.

Configuring the Retry Count for HTTP Client Connection

You can configure the number of retry attempts to connect to the HTTP client.

Procedure

- Add the following property in the CDD file and specify the number of retries for connecting to the HTTP client.

```
com.tibco.be.http.client.retryCount
```

By default, retry attempts are disabled.

HTTP Functions for HTTP Request Messages Configuration

This section explains how HTTP functions are used to send secure and non-secure HTTP requests to other servers, and work with responses received.

HTTP functions are located in the HTTP section of the Standard function catalog.



For configuring SSL over HTTP when the Identity Resource is of the type Certificate/KeyURL, you need to setup Native Tomcat libraries. For more information see the *TIBCO BusinessEvents Installation*.

Generating a Self-Signed SSL Certificate (Keystore)

HTTPS requires use of an SSL certificate. It creates a keystore file and password for the specified type of SSL certificate (keystore).



Avoid using self-signed certificates for production.

If you use trusted certificates, see [Loading Trusted Certificates](#).

Signature

```
Object createKeystore(String ksFilePath, String ksType, String ksPassword)
```

Description

Creates and returns a keystore object, using the given parameters.

Parameters

Name	Type	Description
ksFilePath	String	The absolute path of the keystore file.
ksType	String	The type of keystore. JKS and PKCS12 are supported.
ksPassword	String	Obfuscated password for the keystore.

Returns

Type	Description
Object	The keystore object

Getting POST Data

Signature

```
Object getPostData(SimpleEvent event)
```

Description

Returns the POST data sent in an HTTP POST request (event).

Parameters

Name	Type	Description
event	SimpleEvent	Request event sent on the HTTP destination. Must not be null.

Returns

Type	Description
Object	The POST data

Loading Trusted Certificates

As explained in [Add an HTTP Connection](#), ensure that certificates from trusted certificate authorities are stored in the project, using an Identity resource in the TIBCO Shared Resources folder. Alternatively

use the `BE_GLOBAL_TRUSTED_CA_STORE` global variable to store a location of the certificates outside of the project.

Signature

```
Object loadTrustedCertificates(String trustedCertsFolder, String passwordToSet)
```

Description

HTTPS requires 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
<code>trustedCertsFolder</code>	<code>String</code>	The project path to the folder containing the certificates. If trusted certificates are stored outside the project, use the following construct in the function: <code>System.getGlobalVariableAsString("BE_GLOBAL_TRUSTED_CA_STORE")</code>
<code>passwordToSet</code>	<code>String</code>	Obfuscated password for the keystore.

Returns

Type	Description
<code>Object</code>	The certificate's keystore object

Sending an Event

To send an event as a request encapsulating request headers as properties.

Signature

```
Event sendRequest(String url, SimpleEvent requestEvent, String responseEventURI, long timeoutMillis, Object httpConnectionInfo)
```

Description

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

You can disable the cookies using the `disableCookies()` function. To ensure that the entire message along with the response header and the request body is sent at once, use `disableExpectContinueHeader()`.

Parameters

Name	Type	Description
<code>url</code>	<code>String</code>	The URL for the endpoint that will receive this request.
<code>requestEvent</code>	<code>SimpleEvent</code>	The event to serialize and send to the server.

Name	Type	Description
<code>responseEventURI</code>	String	The fully qualified path of an event. This event is created when the response is received.
<code>timeoutMillis</code>	Long	The timeout interval for the operation. If the value is -1, the server waits forever.
<code>httpConnectionInfo</code>	Object	HTTP Connection Info object.

Returns

Type	Description
Event	An event as a response to the request

Sending an Asynchronous Request (Not Secure)

To send requests to an HTTP server asynchronously, use this function.

Signature

```
String sendAsynchronousRequest(String url, SimpleEvent requestEvent, String correlationId, String successCallbackRuleFunctionURL, String errorCallbackRuleFunctionURL, String methodType, Object httpConnectionInfo)
```

Description

Sends a request to the server specified by the `url` parameter. When TIBCO BusinessEvents receives a response, the callback function is called.

Returns a correlation ID, which is either passed as input, or is generated from the server if the parameter is null. This ID enables you to correlate a request with its response.

You can disable the cookies using the `disableCookies()` function. To ensure that the entire message along with the response header and the request body is sent at once, use `disableExpectContinueHeader()`.

For handling timeout case, you can set the java property `com.tibco.be.http.client.socketTimeout` with the required timeout period.

Set the property `be.engine.http.client.async.responseEventOnError.enabled` to true to pass an additional parameter (output event) to the `errorCallback` rulefunction specified in the `HTTP.sendAsynchronousRequest()` function. The default value of the property is false.

Parameters

Name	Type	Description
<code>url</code>	String	The URL for the server that will receive this request.
<code>requestEvent</code>	SimpleEvent	The event to serialize and send to the server.
<code>correlationID</code>	String	An optional ID to correlate the request and the response. If not specified, the ID is generated by the server.

Name	Type	Description
successCallbackRuleFunctionURL	String	The fully qualified path of a rule function to be invoked for success case. This rule function is called when a successful response is received. The response event would contain the correlation ID. The rule function must have correlation ID, RequestEvent, and ResponseEvent as parameters.
errorCallbackRuleFunctionURL	String	The fully qualified path of a rule function to be called in case of an error. This rule function is called when the response received indicates an error. The response event would contain the correlation ID. The parameter can be null. The rule function must have correlation ID, RequestEvent, and ResponseEvent as parameters.
methodType	String	The HTTP method type. Valid values are: GET or POST.
httpConnectionInfo	Object	HTTP Connection Info object.

Returns

Type	Description
String	A correlation ID

Sending a Secure Asynchronous Request

To send asynchronous requests using SSL, use this function.

Description

This function is the same as the `sendAsynchronousRequest()` function.

For secure communication, use `sendAsynchronousRequest()` function along with the `setSecureInfo()` catalog function with the addition of the SSL-related parameters shown below:

```
Object connectionInfo= HTTP.ConnectionInfo.createHTTPConnectionInfo(true);
HTTP.setSecureInfo(Object connectionInfo, String sslProtocol, Object
clientIdKeyStore, String clientIdPassword, Object trustedCertsKeystore, String
trustedCertsPassword, boolean verifyHostName)
```

For handling timeout case, you can set the java property `com.tibco.be.http.client.socketTimeout` with the required timeout period.

Parameters

All of the parameters for `sendAsynchronousRequest()` plus the following:

Name	Type	Description
clientIdKeystore	Object	The keystore object for client identity.

Name	Type	Description
clientIdPassword	String	Password for the client ID keystore.
trustedCertsKeystore	Object	Keystore Object for trusted certificates.
trustedCertsPassword	String	Password for the trusted certificates keystore.
verifyHostName	Boolean	Flag for checking if a host name matches the names stored in the server's certificates.
httpConnectionInfo	Object	HTTP Connection Info object.

Returns

Type	Description
String	A correlation ID

Sending a Secure Synchronous Request

To send synchronous requests using SSL, use this function.

Description

This function is the same as the `sendRequest()` function.

For secure communication, use `sendRequest()` function along with the `setSecureInfo()` catalog function for SSL-related parameters :

```
Object connectionInfo= HTTP.ConnectionInfo.createHTTPConnectionInfo(true);
HTTP.setSecureInfo(Object connectionInfo, String sslProtocol, Object
clientIdKeyStore, String clientIdPassword, Object trustedCertsKeystore, String
trustedCertsPassword, boolean verifyHostName)
```

Parameters

Name	Type	Description
url	String	The URL for the server that will receive this request.
requestEvent	SimpleEvent	The event to serialize and send to the server.
responseEventURI	String	The fully qualified path of an event. This event is created when the response is received.
clientIdKeystore	Object	The keystore object for client identity.
clientIdPassword	String	Password for the client ID keystore.
trustedCertsKeystore	Object	Keystore Object for trusted certificates.
trustedCertsPassword	String	Password for the trusted certificates keystore.

Name	Type	Description
<code>verifyHostName</code>	Boolean	Flag for checking if a host name matches the names stored in the server's certificates.
<code>timeoutMillis</code>	Long	The timeout interval for the operation. If the value is -1, the server waits forever.
<code>httpConnectionInfo</code>	Object	HTTP Connection Info object.

Returns

Type	Description
Event	An event as a response to the request

Sending a Request via Proxy Server

To send requests via a proxy server, use this function.

Signature

```
Object setProxy(Object connectionInfo, String proxyHost, int proxyPort)
```

Description

This function updates the `HttpConnectionInfo` object with the proxy server details.

For secure communication, use the function as follows for both Synchronous and Asynchronous requests:

```
Object connectionInfo= HTTP.ConnectionInfo.createHTTPConnectionInfo(true);
HTTP.setSecureInfo(Object connectionInfo, String sslProtocol, Object
clientIdKeyStore, String clientIdPassword, Object trustedCertsKeystore, String
trustedCertsPassword, boolean verifyHostName)
HTTP.setProxy(Object connectionInfo, String proxyHost, int proxyPort)
```

Parameters

Name	Type	Description
<code>connectionInfo</code>	Object	HTTP Connection Info object.
<code>proxyHost</code>	String	Proxy host.
<code>proxyPort</code>	Integer	Proxy port.

Returns

Type	Description
Object	Connection info object.

Disabling HTTP Methods at Channel and Destination Level

If required, you can disable specific HTTP methods on the native HTTP channel.

The HTTP methods can be disabled at the channel level and the destination level. You can also disable them at both levels and use the combination according to your requirement. For example, you can disable the GET method for a channel (TestChannel) and for the same channel (TestChannel) disable the POST method for a destination (getData). Then for that destination (getData) the GET and POST methods do not work, while for other destinations only the GET method is disabled.

```
Channels.TestChannel.disableHttpMethods=GET
Channels.TestChannel.getData.disableHttpMethods=POST
```

Procedure

1. Add the `disableHttpMethods` system property at the channel level, in the system TRA file or the project's CDD file, to disable the specified HTTP methods for all destinations of the channel.

`<Channel Folder>.<Channel Name>.disableHttpMethods`

For example,

```
Channels.TestChannel.disableHttpMethods=GET
```

2. Add the `disableHttpMethods` system property at the destination level, in the system TRA file or the project's CDD file, to disable the specified HTTP methods for that destination of the channel.

- Regular destination - `<Channel Folder>.<Channel Name>.<Destination Name>.disableHttpMethods`

- Pageflow destination - `<Channel Folder>.<Channel Name>.<Context Path>.disableHttpMethods`

For example,

```
Channels.TestChannel.getData.disableHttpMethods=POST
```

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.



When TIBCO BusinessEvents acts as a web service server, clients of the service can view the exported concrete WSDL for the web service using the URL to the document root folder. You must place the WSDL file in that location.

Overview of SOAP Related Resources

Project resources must be configured so that TIBCO BusinessEvents can send and receive SOAP messages.

The resources to configure are as follows:

SOAP Destinations

When a correctly configured HTTP destination receives a SOAP request, the SOAP serializer deserializes the SOAP message to its corresponding event. That event has to be inherited from a `SOAPEvent`. The event payload contains the SOAP envelope.

SOAP Events

To create a SOAP event, create a `SimpleEvent` that inherits a `SOAPEvent`. It makes event configuration easier. Its payload has a message root element having an `Envelope` child element. The root element contains `Header` and `Body` elements, and the `Body` element has a `Fault` element. You can further configure these elements using the payload editor.

SOAP messages (and events) can have attachments.



For outbound SOAP events, ensure that the default destination is specified. If it is not specified, an incorrect SOAP message, with "Message" as its root instead of "Envelope", is sent out.

Also, ensure that the default destination for the SOAP event uses `com.tibco.cep.driver.http.serializer.SOAPMessageSerializer`.

SOAP Encoding

Only the SOAP document/literal form of encoding is supported.

Rules and Rule Functions

A rule in the project accesses the SOAP event payload as needed using the mapper or SOAP catalog functions. The rule puts the SOAP response into the payload of a `SOAPEvent` that is sent to the client. The SOAP serializer sends the SOAP response to the client.

SOAP Catalog Functions

SOAP catalog functions enable you to access and process the contents of the following elements in the event payload of an incoming `SOAPEvent`, and add elements to the outbound `SOAPEvent`:

- Header
- Body
- Fault

- Attachment

Mapping of SOAP Request URI to Destination

For SOAP requests, the header property `SOAPAction` represents the destination name, and the `requestURI` represents the channel URI. TIBCO BusinessEvents combines the `SOAPAction` and `requestURI` to create a destination URI. It maps the request with a destination having same URI as the newly created URI.

For example, if the `requestURI` of a request is `/QueryBooks` and the value of `SOAPAction` is `QueryBooksByAuthor`, then TIBCO BusinessEvents engine maps the request with a destination having `/QueryBooks/QueryBooksByAuthor` URI if it exists.

Manually Creating Resources to Work with SOAP Services

This section explains how you can manually create and configure project resources to work with SOAP services.

The section [Creating Resources Using the WSDL Import Utility](#) explains how to use the WSDL import feature to create the resources.

Procedure

1. Configure the HTTP Channel and SOAP Destination

Configure an HTTP channel, an HTTP connection resource for the channel. Connection configuration is the same for SOAP and HTTP channels, except for the serializer. For SOAP destinations, use the following serializer:

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

After you complete [Add a SOAPEvent \(and Other Ontology as Needed\)](#), set that `SOAPEvent` as the default event for the destination.

2. Add a SOAPEvent (and Other Ontology as Needed)

Configure the `SOAPEvents` that 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:

- Add a simple event in the usual way.
- In the `Inherits From` field, select **SOAPEvent**.
- Set the default destination to the destination you created in [Configure the HTTP Channel and SOAP Destination](#).

The payload of a `SOAPEvent` is automatically configured with the structure of a SOAP message. It has `Header` and `Body` elements, and within the `Body` element, a `Fault` element. You can further configure the `Header` and `Body` elements using the payload editor.

3. Configure Rules and Rule Functions using SOAP Functions

In general the procedure of serving requests and sending requests to other servers is the same for SOAP services as for other HTTP interactions. One rule function is used as the event preprocessor.

In addition, you must configure rules and rule functions to access information from the SOAP messages in the inbound events, and to populate outbound events with SOAP message details. Set two `SOAPEvents` you configured in [Add a SOAPEvent \(and Other Ontology as Needed\)](#) as input and return arguments of the preprocessor. See [Parsing and Building SOAP Messages](#) for details.

4. Configure an Event Preprocessor

Open the project CDD for editing, and configure an event preprocessor for the destination you configured in [Configure the HTTP Channel and SOAP Destination](#). Use the rule function you configured in [Configure Rules and Rule Functions using SOAP Functions](#).

Creating Resources Using the WSDL Import Utility

WSDL (Web Services Description Language) is an XML format for describing web services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.



- Only document style WSDLs with literal encoding are currently supported.
- Only In-Only and In-Out Message Exchange Patterns (MEPs) are currently supported.

TIBCO BusinessEvents can parse an imported WSDL file and create the required project artifacts based on its contents: the channels and destinations (concrete WSDLs only), events, rules, and rule functions.

Both abstract and concrete WSDL files can be used. When you import an abstract WSDL, channel and transport information is not available and you must create those resources manually. Concrete WSDLs contain information that is used to create these resources automatically.

You then implement the newly created rules and rule functions to provide the expected behavior for the web service described in the WSDL file.

Importing a WSDL File

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

Procedure

1. Import the WSDL file into the project, as follows.
 - a) In Studio Explorer, right-click the project folder (or the project root) where you want to store the WSDL file and select **Import**. You see the Import wizard.
 - b) At the Select dialog, select **General > From File System**. Click **Next**.
 - c) At the File system dialog, click **Browse**, select the directory that contains the WSDL file you want to use, and click **OK**.
 - d) The File system dialog displays the directory in the left panel and its files in the right panel. Select the check box 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.

Procedure

1. Navigate to the project directory and rename the WSDL file.
2. Copy the updated WSDL file to the same directory. Ensure that the name of the WSDL file is the same.
3. In Studio Explorer, select the WSDL file and then select **Edit > Delete** from the top 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/>

[Exporting Project Artifacts as WSDL](#) shows how related project resources are exported as a WSDL file.

Exporting Project Artifacts as WSDL

Project Folders and Resources	WSDL Artifact after Exporting
<p>RuleFunction</p> <p>Use the rule function specified as the input event's default destination as the SOAPEvent's preprocessor.</p> <p>That rule function must have the "SOAPEvent for input" as the input and the "SOAPEvent for output" as the return type.</p>	Forms the <wsdl:operation> operation.
<p>SOAPEvent</p> <ul style="list-style-type: none"> • SOAPEvent for the WSDL input argument OR • SOAPEvent for the WSDL output argument <p>Use the event you configured in Add a SOAPEvent (and Other Ontology as Needed) .</p>	Creates a message. The contents of Envelope > Header > Body become message parts.
<p>HTTP Channel</p> <p>Channel with HTTP as the Driver, Resource as the Method of Configuration, with a pointer to the HTTP Connection.</p>	The URI of the channel forms the <soap:address> URI.
<p>HTTP Connection for HTTP Channel</p> <p>Host and port are used.</p>	The name of the connection forms the <wsdl:port>. Host and port of the connection form the <soap:address> host and port.
<p>Destination for HTTP Channel</p> <p>With SOAPMessageSerializer as the default serializer.</p> <p>The destination used is the default destination of the SOAPEvent.</p>	Forms the soapAction attribute of the operation. The destination name becomes soapAction specified for that particular operation.
<p>CDD</p> <p>In the Agent Classes tab, Input Destination Collections list, the Preprocessor specified for the input SOAPEvent's default destination.</p> <p>Place the input destination directly under the agent class input destination collections, and not under Collections > Input destination.</p>	Associates soapAction to the operation in the HTTP binding.

Exporting a Rule or Rule Function as a WSDL

Procedure

1. Right-click the project, and click **Export**.
2. In the Export dialog, 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 information out of the incoming event payload, which contains the SOAP message.

To Parse the SOAP Envelope

```
String getEnvelope(SimpleEvent inSoapEvent)
```

Given a request SOAPEvent, this function returns the SOAP envelope in the request event payload, for example:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sch="http://www.tibco.com/schemas/SoapOverHttp/Schema/Schema.xsd" >
  <soapenv:Header/>
  <soapenv:Body>
    <sch:root>
      <sch:First>1</sch:First>
      <sch:Second>2</sch:Second>
    </sch:root>
  </soapenv:Body>
</soapenv:Envelope>
</message>
```

Parsing (and Optionally Removing) Headers and Header Attributes

```
String[] getHeaders(SimpleEvent inSoapEvent, String actor, Boolean removeHeaders)
```

If the `actor` parameter has a null value then all the immediate children of the Header element are retrieved:

```
getHeaders(inSoapEvent, null, false)
```

Otherwise, the header specified by the `actor` attribute is retrieved. For example, given this Headers element in a SOAP event payload:

```
<soapenv:Header>
  <t:user xmlns:t="http://schemas.xml.com" soapenv:mustUnderstand="true"
    soapenv:actor="http://localhost:9090/Service">jon</t:user>
  <t:user_surname xmlns:t="http://schemas.xml.com"
    soapenv:mustUnderstand="true"
    soapenv:actor="http://localhost:9090">smith</t:user_surname>
</soapenv:Header>
```

If you specify the following:

```
getHeaders(inSoapEvent, "http://localhost:9090/Service", false)
```

To remove the specified header part or parts, set the final parameter to `true`. (The SOAP specification states that if a header is processed it should be removed. You would remove a header if TIBCO BusinessEvents is acting as an intermediary node and the request created using the SOAP functions 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()
```

Parsing the SOAP Body (SOAPBodyParts)

Two functions are available for getting SOAP body parts.

```
String[] getAllSOAPBodyParts(SimpleEvent inSoapEvent)
String[] getSOAPBodyParts(SimpleEvent inSoapEvent, String name, String namespace)
```

The `getAllSOAPBodyParts()` function simply returns all SOAP body parts.

Using the `getSOAPBodyParts()` function you can specify which parts are of interest. Given a body part name and a namespace for a specified SOAPEvent, it returns a String array of matching SOAP body parts in serialized form. Name and namespace parameters cannot be null.

Example

Given this function:

```
String[] body_part= getSOAPBodyParts
(soapeventin, "root", http://www.tibco.com/schemas/SoapOverHttp/Schema/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>
```

Parsing Attachments

The following functions enable you to work with SOAP attachments in the request message:

```
getNumberOfAttachments(SimpleEvent inSOAPEvent)
getAttachmentContentID(SimpleEvent inSOAPEvent, int Index)
getAttachmentContentType(SimpleEvent inSOAPEvent, int Index)
getAttachmentContent(SimpleEvent inSOAPEvent, int Index)
getAttachmentContentByContentID(SimpleEvent inSOAPEvent, string contentID)
```

The content ID is the attachment identifier. You can select which attachment to work with using its index position. First get the count of the attachments using `getNumberOfAttachments()`. Then using the index, you can get the content ID and content type, as well as the attachment content itself.

The content is returned in byte form, so after you get the content, you must then use other functions to make the content human-readable.

For an example, see <http://www.w3.org/TR/SOAP-attachments#SOAPReferenceToAttachments>.

Parsing SOAP Fault XML Nodes

The following functions enable you to work with the standard SOAP Fault XML nodes from the payload of a SOAP event:

```
getFault (SimpleEvent, soapEvent)
getFaultActor (SimpleEvent, soapEvent)
getFaultCode (SimpleEvent, soapEvent)
getFaultString (SimpleEvent, soapEvent)
```

Working with Outgoing SOAP Messages (Event Payloads)

You can add header parts, body parts, fault parts and attachments to the outgoing SOAP message (whether it is a response or a request).

The signatures of the relevant functions are as follows:

```
addHeaderPart(SimpleEvent outSOAPEvent, String headerPartXml)
addSOAPBodyPart(SimpleEvent outSOAPEvent, String bodyXML)
addSOAPHeaderAttribute(SimpleEvent outSOAPEvent, int index, String attribute,
String value)
addFaultPart(SimpleEvent outSOAPEvent, String faultCode, String faultMessage,
String faultActor, String faultDetailString)
addAttachment(SimpleEvent outSOAPEvent, String contentID, String content, String
contentType, String contentEncoding)
```

TIBCO BusinessEvents adds each type of fragment to the appropriate part of the event payload: header, body, or fault. The fragments must be well-formed XML. You can also add attachments.

For example, to add a body part containing information for a response you would include all the required details including any namespace information:

```
SOAP.addSOAPBodyPart(outSOAPEvent, "<ns0:BookStore xmlns:ns0=\"http://
www.abc.com/xsd/books\"><ns0:Book><ns0:Author>J.K.Rowling</ns0:Author></ns0:Book></
ns0:BookStore>");
```

The specified body part is added to the correct place in the outline structure of the SOAP message, which is provided by the SOAPEvent. The resulting payload would look similar to the following:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns0:BookStore xmlns:ns0="HTTP://www.abc.com/xsd/books">
      <ns0:Book>
        <ns0:Author>J.K.Rowling</ns0:Author></ns0:BookStore>
      </ns0:Book>
    </ns0:BookStore>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Understanding the WSDL to Project Resource Mapping

A WSDL file describes a web service. The WSDL Import utility imports a WSDL file and generates TIBCO BusinessEvents project artifacts using elements in the WSDL. TIBCO BusinessEvents can import abstract and concrete WSDL files. The source of the WSDL could be, for example, an ActiveMatrix BusinessWorks SOAPRequestReply activity.



For WSDL import to be successful, all names in the WSDL must conform to TIBCO BusinessEvents folder and entity naming requirements.

Example WSDL

The table following this example shows which WSDL elements and attributes are used to create TIBCO BusinessEvents project artifacts.

Elements and attributes used in the import are highlighted in bold text. Differences between import from abstract and concrete WSDL files are also highlighted. See [Imported WSDL Project Artifacts](#) 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.

Imported WSDL Project Artifacts

Project Folders and Resources	Project Resource Type	WSDL Source
Library	Project root folder	N/A
getBook/	Folder	For concrete WSDLs: <wsdl:service> For abstract WSDLs: <wsdl:service> is not present, so the folder structure starts from the folder created from <wsdl:portType>.
GetBookPortType/	Folder	For abstract and concrete WSDLs: <wsdl:portType>
getBook/Events/	TIBCO BusinessEvents folder	
GetBookRequestMessage	Event	<wsdl:input> For abstract WSDLs, in <wsdl:portType> section. For concrete WSDLs, in <wsdl:binding> section.
GetBookResponseMessage	Event	<wsdl:output> For abstract WSDLs, in <wsdl:portType> section. For concrete WSDLs, in <wsdl:binding> section.
getBook/RuleFunctions/	TIBCO BusinessEvents folder	
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>

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 .	<soap:address location> (from the last part of the location URL)
Service_getBook	Destination See Destination Names	<soap:operation soapAction>

How Project Artifacts are Named

Channel Folders

Given a concrete WSDL, folders are created for all the elements after the port, up to the last forward slash of the location URL. The text after the last forward slash is the channel name. For example, given the following location URL:

```
http://ACME:9090/Service/Trial/getBook
```

The folder structure would be /Service/Trial and the channel name would be getBook.

Destination Names

In a concrete WSDL, the SOAPAction attribute of a <soap:operation> element specifies the URL of a destination. It also becomes the destination name. Forward slashes (/), colons (:), and periods (.) are converted to underscore characters (_) to form the name. For example:

```
http://www.acme.com/TNT/webservices/getByteField
```

Becomes:

```
http://www.acme.com/TNT/webservices/getByteField
```

Rules and Rule Functions

For each operation, the import utility creates a rule and a rule function. The rule has no body. The rule functions have `SoapEventOut` as the return type. Null value is returned by default.

For example, the `GetBook` operation becomes a `GetBook` rule in the `GetBookPortType` folder which is in the `Rules` folder, and also a `GetBook` rule function in the `Rulefunctions` folder.

You implement the rules and rule functions in your project according to the web service you want to implement.

Events

The `<wsdl:input>` element becomes a request event and the `<wsdl:output>` element becomes a response event. Each event type inherits from the `SoapEvent` event type.

Event names come from the message attributes. In the example, the request event is `GetBookRequestMessage` and the response event is `GetBookResponseMessage`.

Faults

Faults specified in a WSDL are used in the outbound SOAP event, as the `Fault` element.

Hawk Channels

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

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

AlertMonitor

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

RuleBaseListMonitor

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

Agent Monitor

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

MicroagentListMonitor

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

ErrorMonitor

It is used to listen on the event for errors.

WarningMonitor

It is used to listen on the event for warnings.

Microagent Method Subscription

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

You need to create a channel with a driver type Hawk as described in *TIBCO BusinessEvents Developer's Guide*, Adding a Channel.

Working with Hawk Channels

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

Perform the following steps:

1. Add the following environment variables:

```
tibco.env.HAWK_HOME=<absolute_path_where_TIBCO_Hawk_is_installed>
tibco.env.TRA_HOME=<absolute_path_where_TIBCO_Rendezvous_is_installed>
```

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

```
tibco.env.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/tibrvj.jar;%RV_HOME%/lib/;%TRA_HOME%/lib;%EMS_HOME%/lib
```

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

Setting Up the Runtime Configuration

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

Procedure

1. Add or edit the environment variable `HAWK_HOME`:
2. Depending on the transport used by the Hawk Connection shared resource, set one of the following:

```
tibco.env.HAWK_HOME=<absolute_path_where_TIBCO_Hawk_is_installed>
tibco.env.RV_HOME=<absolute_path_where_TIBCO_Rendezvous_is_installed>
tibco.env.EMS_HOME=<absolute_path_where_TIBCO_Enterprise_Message_Service_is_installed>
```

Using the Hawk Destination and Event Wizard

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

You can choose one of the wizard types:

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

Creating a Destination and Event Using the Wizard

Procedure

1. Start the Hawk Destination and Event Wizard as described in [Using the Hawk Destination and Event Wizard](#) and select the wizard type `Create Destination and Event`. Click **Next**.
2. On the `Select Channel` dialog, select the Hawk channel from the list for which you need to create a destination and event. Click **Next**.
3. On the `Select Monitor Type` dialog, select the monitor type for the Hawk channel. The monitor type determines the type of Hawk events that will be monitored. Click **Next**.
If you selected `MicroAgent Method Subscription`, proceed to [step 4](#). Otherwise go to [step 7](#).
4. The wizard detects the available Hawk agents and displays them on the `Select Hawk Agent` dialog. Select a Hawk agent from the list and click **Next**.
5. The `Select Hawk MicroAgent` dialog is displayed. Select a microagent from the list of available Hawk microagents and click **Next**.
6. The `Select Hawk MicroAgent Method` dialog is displayed. Select a method from the list of available methods of the microagent. The tooltip displays the arguments for the selected method along with their datatypes.
7. The `Set File and Folder name` dialog is displayed. Enter the name of the destination to be created, the folder name where the event is to be stored, and the name of the event to be created.
8. Click **Finish**. A message 'Created the resources of Hawk Channel successfully.' indicates that the Hawk destination and the event have been created.

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

Creating Event for Hawk Catalog Function Using Wizard

Procedure

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

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

Hawk Channel Destination Reference

Field Name	Description
Name	Name of the destination to be created. The default name on the screen is of the format NewDestination_n.
Description	Description of the destination that is to be created.
Default Event	The default event for the destination. You can browse and select an existing event from the project.
Serializer/Deserializer	The default serializer used to deserialize a Hawk event to a simple event in TIBCO BusinessEvents and to serialize an event in TIBCO BusinessEvents to a Hawk event. You must configure the serializer (<code>com.tibco.cep.driver.hawk.serializer.HawkSerializer</code>) for every destination created on the Hawk channel.

Field Name	Description
Monitor Type	<p>Type of monitor listening to different Hawk events. Choose one of:</p> <ul style="list-style-type: none"> • AlertMonitor • RuleBaseListMonitor • AgentMonitor • MicroAgentListMonitor • ErrorMonitor • WarningMonitor • MicroAgentMethodSubscription
Subscription Method URI	<p>URI for the Hawk microagent subscription method.</p> <p>When you use the TIBCO Channel Wizard to create a destination, you can select a subscription method from the list of available methods.</p>
TimeInterval (seconds)	<p>Time interval (specified in seconds) between successive calls to a subscription method.</p>
Arguments	<p>Optional. The subscription method selected determines if the arguments are required. Some methods do not require any arguments.</p> <p>Enter the arguments required by the selected subscription method, if any.</p> <p>If you create the destination using the TIBCO Channel wizard, the wizard automatically enters the arguments for the selected subscription method, if any.</p>

StreamBase Channel

Using the StreamBase channel, you can send messages back and forth between TIBCO StreamBase and TIBCO BusinessEvents. For example, you can use StreamBase for aggregation and filtering data, and then use BusinessEvents to process context based rules.

A BusinessEvents application can read from (*dequeue*) and write to (*enqueue*) any StreamBase stream using the StreamBase channel. Thus, an existing StreamBase application requires no modification to exchange messages with a BusinessEvents application. The configurations for communicating with the StreamBase application is performed in the BusinessEvents application.

StreamBase Key Terms

StreamBase is a new type of computing platform, designed specifically to meet the performance requirements of high-volume, real-time streaming applications. At its core, StreamBase implements a unique Stream Processing Engine™ named StreamBase Server. StreamBase processes the inbound data while it is in flight, as it streams through the server.

For more detail information regarding StreamBase, refer to the *TIBCO StreamBase Documentation*.

The following are the key components and terms of StreamBase:

StreamBase Server

A server process that listens for, and acts upon, client requests (commands and data) in a StreamBase application. Also known as the StreamBase Daemon. This process is the primary server for StreamBase, accepting the incoming streaming data, managing the execution queues, and performing the real-time processing work as defined in the operators that comprise your application.

Stream

An ordered and potentially unlimited collection of tuples that share the same schema and arrive at the same processing input. A stream has one producer, which is either an operator in the application or an external data source. A stream may also have zero or more consumers, which may be operators or output streams.

Tuple

A set of data, associated with an event message that is allowed to flow through a stream. A tuple is composed of one or more fields; each field is a name-value pair. Each tuple must conform to the stream's schema, which specifies permissible names and data types.

Schema

A description of valid fields in a tuple, defining acceptable names and data types.

Creating a Destination and SimpleEvent from a StreamBase Schema

The StreamBase Channel in BusinessEvents requires a Simple Event definition in the BusinessEvents Project to match the corresponding schema definition in the StreamBase application.

Procedure

1. In BusinessEvents Studio Explorer, select **File > New > Other > TIBCO Channel Wizards > StreamBase Destination and Event** and click **Next**.
2. In the StreamBase Event Details window, select the parent folder for the simple event, enter its name in the **Name** field and description in the **Description** field, and click **Next**.
The StreamBase Schema Location Info window is displayed to specify the source location of the StreamBase Schema.
3. Select the StreamBase schema source location:

- **StreamBase Server** - Select this option if you want to select the schema available on the StreamBase server. You can either enter the StreamBase server details or import it from an existing StreamBase channel.
 - Select the **Populate server details from existing StreamBase Channel** check box, if the BusinessEvents project already has a StreamBase channel with the StreamBase server details. Click **Browse** and select the required StreamBase channel. The server details are automatically populated from the StreamBase channel. Click **Next**.
 - Enter the StreamBase server details, see [StreamBase Channel Properties](#).
Optionally, select the **Create new StreamBase Channel from Server details** check box, if you want to create a new StreamBase Channel using the entered StreamBase server details. Click **Next**.
- **StreamBase Named Schema** - Select this option if you want to load the StreamBase schema from an SBAPP or SBINT file. The SBAPP file is an XML source file for an EventFlow application. Click **Browse** and select the .sbappor .sbint file from the local system. Click **Next**.



If the **StreamBase Named Schema** option is selected then only named schema and input stream schema are listed under StreamBase Scheme Selection page.

If you have selected the **Create new StreamBase Channel from Server details** check box, the StreamBase Channel Details page is displayed; otherwise, the StreamBase Schema selection page is displayed.

4. (Optional) In the StreamBase Channel Details page enter the channel details and click **Next**.

Option	Description
Enter or select the parent folder	Enter the name of the parent folder in the project or select the parent folder from the project tree.
Channel name	Name of the channel.
Description	Short description for the channel.
Destination Name	<p>Optional. Name of the destination to be created in the channel.</p> <p>The Client Type property of the destination is depended on the schema type.</p> <ul style="list-style-type: none"> • Input schema type - Enquirer client type • Output schema type - Dequirer client type <p>See Configuration for StreamBase Destinations for more details on StreamBase destination properties.</p>

The StreamBase Schema selection page is displayed listing the available schemas from the source location.

5. Select the StreamBase schema from the **Available schemas** list and click **Finish**.

For a **StreamBase Server** source location, the schemas are retrieved from all input and output streams from the application. For a **StreamBase Named Schema** source location, the source SBAPP file is parsed and the named schema definitions are displayed.

A new Simple Event is created, with properties defined from the **Schema Structure** of the selected schema. See [Datatype Conversion](#) for more details on mapping of StreamBase datatype to BusinessEvents datatype.

Message Processing

BusinessEvents uses StreamBaseSerializer to processes the incoming and outgoing messages in StreamBase channel.

Incoming Messages

StreamBaseSerializer takes the incoming message from the StreamBase server, which is cast as a Tuple. The Tuple is transformed to the corresponding BusinessEvents event. The BusinessEvents event is then asserted into working memory.

When a Tuple is transformed into the BusinessEvents event, the following steps are taken:

1. The schema is retrieved from the Tuple
2. All fields are then retrieved from the schema
3. For each field from the Tuple, the event is checked for a matching property (field name).
4. If a matching property is found, the property value is set to the field value from the Tuple.
5. If no matching property is found, a warning is logged and the process moves to the next field.

Outgoing Messages

The two functions in BusinessEvents that you can use to send events out to other application are:

- `Event.sendEvent()` that automatically sends the event to the default destination.
- `Event.routeTo()` that takes a destination as an argument.

For the StreamBase channel, use these functions to send a tuple back to StreamBase. Using StreamBaseSerializer these functions transform the event into a tuple, and enqueue the tuple to the stream specified in the destination.

Datatype Conversion

Some of the BusinessEvents simple event property types are directly mapped to the datatype of the StreamBase tuple fields. The StreamBase datatypes that cannot be directly mapped to the BusinessEvents property type are converted to *String* representation.

The following table lists the StreamBase datatypes and their respective BusinessEvents datatypes (if present).

StreamBase to BusinessEvents Datatype Conversion

StreamBase Datatype	BusinessEvents Datatype
<code>DataType.BOOL</code>	<code>boolean</code>
<code>DataType.DOUBLE</code>	<code>double</code>
<code>DataType.INT</code>	<code>int</code>
<code>DataType.LONG</code>	<code>long</code>
<code>DataType.STRING</code>	<code>String</code>
<code>DataType.TIMESTAMP</code>	<code>DateTime</code>

StreamBase Datatype	BusinessEvents Datatype
<code>DataType.BLOB</code>	String
<code>DataType.LIST</code>	String
<code>DataType.TUPLE</code>	JSON String Tuple value is converted to JSON string, which can be used later on. For example, you can use JSON function to extract object values.
<code>DataType.FUNCTION</code>	String Note: The Function type is available only when client type is dequeuer. It holds value as string representation of the function definition which can be used for testing or debugging purposes.

Payload Mapping

To get the event payload in StreamBase, add the `_payload_` field to the StreamBase stream schema. This field holds the XML event payload set in BusinessEvents.

Custom Channel

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

The Java API for custom channel (`com.tibco.be.custom.channel` package) is bundled with TIBCO BusinessEvents. The `com.tibco.be.custom.channel` package consists of interfaces and classes that you can implement and extend to create a custom channel. Implement these classes to initialize the channel, create serializer and deserializer for the custom channel, to store and access event data, and to perform various other operations. For more information about the classes and interfaces in the `com.tibco.be.custom.channel` package see *Java API Reference*. See [Key Java API Components for Custom Channel](#) on page 126 for key components of the custom channel API.

TIBCO BusinessEvents bundles a sample Kafka channel and its source code to showcase the channel API usage. The Kafka channel example is located at `BE_HOME\examples\standard\KafkaChannel` and the source code is located at `BE_HOME\api\channel-api\examples\kafka\src`.

Creating a New Custom BusinessEvents Channel

You can use the custom channel API to create a custom channel according to your requirement for your project.

The Java API for custom channel (`com.tibco.be.custom.channel` package) is bundled with TIBCO BusinessEvents. For more information about the classes in the `com.tibco.be.custom.channel` package see *Java API Reference*.

Procedure

1. Create a XML file and save it with the name `drivers.xml`.
2. In the XML file, define the drivers for the channel in the following format.

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

Where,

- `CHANNEL_TYPE` - The channel type that you have defined. This is displayed in the BusinessEvents Studio under the **Driver Type** field in the New Channel Wizard.
- `CHANNEL_LABEL` - Display name of the channel.
- `DRIVER_CLASS` - The URI of the BaseDriver class that you have implemented.
- `CHANNEL_DESCRIPTION` - Short description of the channel.

- `CHANNEL_VERSION` - Version number of the channel.
 - `<!--Channel Properties -->` - Define the configuration properties for the channel. Specify the name, type, and default value of the properties. These properties are displayed as channel properties in BusinessEvents Studio.
 - `<!--Destination Properties -->` - Define the configuration properties for the destination of the custom channel. Specify the name, type, and default value of the properties. These properties are displayed as destination properties in BusinessEvents Studio.
 - `SERIALIZER_TYPE` - Type of the serializer for the channel. You can define multiple serializers.
 - `SERIALIZER_CLASS` - Class of the serializer that you have defined. Specify at least one serializer for the channel. The class is listed in the BusinessEvents Studio under the **Serializer/Deserializer** field while adding destination for the custom channel.
3. Create all the required Java class files using the Java API for custom channel. Archive all the Java class files for the custom channel with the `drivers.xml` file as a JAR file.
Refer to the Kafka channel example and its source code for the custom channel Java API usage. The Kafka channel example is located at `BE_HOME\examples\standard\KafkaChannel` and the source code is located at `BE_HOME\api\channel-api\examples\kafka\src`. See [Key Java API Components for Custom Channel](#) on page 126 for key components of the custom channel API.
Refer to the *Java API Reference* for complete details on Java API for custom channel (`com.tibco.be.custom.channel` package).
 4. Copy this JAR file at `BE_HOME\lib\ext\tpcl` and restart BusinessEvent Studio.
New custom channel is displayed under the **Driver Type** in the New Channel Wizard.

Sample Kafka Channel Drivers.xml File

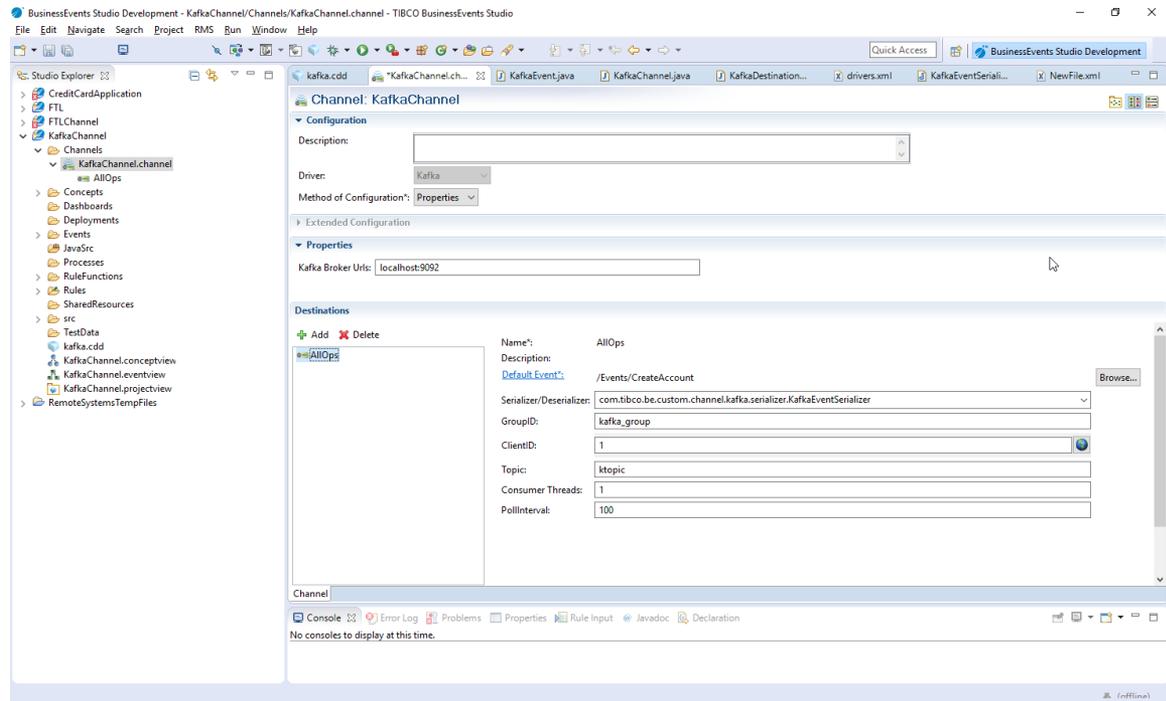
The following sample code is of the `drivers.xml` file for the Kafka channel example:

```
<?xml version="1.0" encoding="UTF-8"?>
<drivers>
  <driver>
    <type>Kafka</type>
    <label>Kafka</label>
    <class>com.tibco.be.custom.channel.kafka.KafkaDriver</class>
    <description>Apache Kafka is an open-source message broker project
developed by the Apache Software Foundation written in Scala. The project
aims to provide a unified, high-throughput, low-latency platform for handling
real-time data feeds.</description>
    <version>1.0.0.0</version>
    <properties>
      <property name="BootstrapServers" type="String"
default="localhost:9095"/>
    </properties>
    <destinations>
      <property name="GroupID" type="String" default="kafka_group"/>
      <property name="ClientID" type="String" default="1"/>
      <property name="Topic" type="String" default="ktopic"/>
      <property name="Threads" type="Integer" default="1"/>
      <property name="PollInterval" type="Integer" default="100"/>
    </destinations>
    <serializers userdefined="true">
      <serializer type="Map"
class="com.tibco.be.custom.channel.kafka.serializer.KafkaEventSerializer"/>
    </serializers>
  </driver>
</drivers>
```

Sample Kafka Channel BusinessEvents Studio UI

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

Kafka Channel BusinessEvents Studio UI



Key Java API Components for Custom Channel

Using the Java API, you can create your own custom channel according to your requirement.

Refer to the *Java API Reference* for complete list of Java classes and interfaces for the custom channel. However, few of the key Java classes, interfaces and their key components are mentioned in the following section. You can implement these methods to create your custom channel.

class `com.tibco.be.custom.channel.BaseDriver`

The following methods of the `BaseDriver` class are called by BusinessEvents runtime, during startup for initialization of channel and destination.

- `getChannel`
- `getDestination`

class `com.tibco.be.custom.channel.BaseChannel`

The following methods of the `BaseChannel` class are called by BusinessEvents runtime during startup, for initializing, starting, connecting and closing of channels.

- `init`
- `start`
- `connect`
- `close`

class com.tibco.be.custom.channel.BaseDestination

The following methods of the BaseDestination class are called by BusinessEvents runtime for initializing, starting, connecting, and closing destinations.

- init
- start
- connect
- close

The following *send* and *requestEvent* methods corresponds to the `Event.sendEvent` and `Event.requestEvent` catalog functions, respectively.

- send
- requestEvent

interface com.tibco.be.custom.channel.Event

You need to implement the Event interface which holds the underlying event related data. An instance of this interface is to be returned through the custom deserializer.



A default implementation of the event interface is provided in the class: `com.tibco.be.custom.channel.DefaultEventImpl`. You can reuse or extend this class or provide your own implementation.

class com.tibco.be.custom.channel.BaseEventSerializer

You need to extend the BaseEventSerializer class. The following methods are used during serialization and deserialization of the custom channel message.

- `initUserEventSerializer` - initializes the serializer
- `serializeUserEvent`
- `deserializeUserEvent`

You must specify the fully qualified Java class name of the serializer in the `drivers.xml` file.

interface com.tibco.be.custom.channel.EventContext

A new instance of this interface is to be returned through the `BaseDriver.getDestination().getEventContext` method. Implement the following EventContext interface methods.

- `reply` - This corresponds to the catalog function `Event.replyEvent`.
- `acknowledge` - For some messaging systems, this method specifies an indication that this message can be removed from the messaging system. This method is called by BusinessEvents when message processing is completed.
- `rollback` - For messaging systems that support transactions, rollback is invoked by BusinessEvents when there is an internal failure during the transaction. The implementation handles rollback as applicable to the custom channel.

ActiveSpaces Channels

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

This section provides a brief introduction to the ActiveSpaces concepts and terminology that are useful when working with the ActiveSpaces channel. See the *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 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 might join or leave the group at any time without requiring any user intervention. A space automatically scales up as the number of processes in the group increases, and scales down when processes suddenly disappear from the group or network. There is no negative impact on the data contained in the space when processes leave the space.

System spaces are a set of administrative spaces that are created and maintained by 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: the best solution might be 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, Entry Browser, or Router.

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

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

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



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

Tuple

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

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

Filters

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

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

Examples

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

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

Operators Supported in Filters

Table 18 shows the operators that are supported in the ActiveSpaces filters:

Operators for ActiveSpaces Filters

Operator	Meaning
>, >=	greater than
NOT or ! or <>	not
*	multiply
=	equal
!=	not equal
ABS	absolute value
MOD or %	modulo, as in (MOD (x,y) or x % y)
NOT BETWEEN	not between
BETWEEN	between
	string concatenation
NOT LIKE	not like (regex)

Operator	Meaning
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 19 shows the formats for values used in filters.

Formats for Filter Values

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

Working with ActiveSpaces Channels

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 for ActiveSpaces Channel 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.

Procedure

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.

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 <code><spaceName>Event</code> .
Destination	
Name	Name of the destination to be created. The default name on the screen is of the format <code><spaceName>Dest</code> .
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 <code>com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer</code> . See ActiveSpacesSerializer for more information.

Field Name	Description
Space Name	<p>Name of the space which the destination connects to. This space is selected in the Select space dialog.</p> <p>See Space for more information about spaces in TIBCO ActiveSpaces.</p>
Distribution Role	<p>The level of participation of the space member: seeder or leech.</p> <p>See Distribution role for more information.</p>
Filter	<p>String specified to evaluate tuples and refine the set of tuples to work. A filter string can be seen as what would follow the <i>where</i> clause in a <code>select * from Space where...</code> statement.</p> <p>See Filters for more information.</p>
Consumption Mode	<p>Specifies the consumption mode for the ActiveSpaces event as one of:</p> <ul style="list-style-type: none"> • Event Listener - listens for specific events to occur, and invokes a callback function from TIBCO ActiveSpaces. • Entry Browser - can listen and retrieve tuples from the space using the Get, Take, or Lock & Take methods. • Router - feeds the events to only the specified listener. This way, the application benefits from having multiple listener processes running. <p>See Consumption mode for more information.</p>
Browser Type	<p>Available only when the consumption mode is Entry Browser.</p> <p>Specifies the browser type used to retrieve tuples from a space. You can choose either Get, Take, or Lock & Take.</p> <p>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.</p> <p>Using the Lock & Take option, you can acquire a lock on the tuple key. The event is consumed or taken in post-RTC ensuring successful processing of the event. The Take action does not occur if <code>abortRtc()</code> is called in preprocessor or rules.</p> <p>Note: The lock owner is the process. Make sure the Lock Scope is defined as a process in the ActiveSpaces space definition.</p>

Field Name	Description
DistributionScope	<p>The distribution scope for the ActiveSpaces event.</p> <p>This field is active only when the Consumption Mode is Event Listener or Entry Browser.</p> <p>When Consumption Mode is Entry Browser, the field specifies the possible ways to browse for events in a Space based upon where entries are stored in the space. The values are:</p> <ul style="list-style-type: none"> • ALL - The browser iterates through all of the events in a space regardless of where the entries, for which those events occurred, are stored. • SEEDED - The browser only iterate through the events in the space that occurred for entries stored on the member. <p>The values when Consumption Mode is Event Listener are:</p> <ul style="list-style-type: none"> • ALL - The listener listens to event related to all the entries in space. • SEEDED - The listener listens only to events associated with the entries in Space that are seeded by the member (will be empty unless the member is a seeder on the space).

Field Name	Description
TimeScope	<p>The time scope for the ActiveSpaces event.</p> <p>This field is active only when the Consumption Mode is Event Listener or Entry Browser.</p> <p>When Consumption Mode is Entry Browser, the field specifies the possible ways to browse for entries in a Space based upon the time at which entries were stored in the space relative to the Browser creation time. The values are:</p> <ul style="list-style-type: none"> • ALL - The browser iterates through all of the entries stored in the space at creation time and continues to iterate as changes occur in the space. • NEW - The browser skips iterating through the entries stored in the space at creation time and only iterates through new changes that occur in the space. • SNAPSHOT - The browser only iterates through entries stored in the space at creation time. <p>The values when Consumption Mode is Event Listener are:</p> <ul style="list-style-type: none"> • ALL - The listener starts with all of the tuples currently in the space at creation time, which will be presented as an initial site of PUT events and then is continuously updated according to changes in the space. • NEW - The listener starts empty and is updated only with events related to new or updated tuples in the space. • SNAPSHOT - The listener contains only PUT events corresponding to the tuples stored in the space at creation time. • NEW_EVENTS - The listener starts empty and is updated with all of the events that occur in the space after creation time.
Put Event	<p>Available only when the consumption mode is Event Listener.</p> <p>When selected, the Event Listener listens for any Put events on the space and invokes a callback function when such an event occurs.</p>
Take Event	<p>Available only when the consumption mode is Event Listener.</p> <p>When selected, the Event Listener listens for any Take events on the space and invokes a callback function when such an event occurs.</p>
Expire Event	<p>Available only when the consumption mode is Event Listener.</p> <p>When selected, the Event Listener listens for any Expire events on the space and invokes a callback function when such an event occurs.</p>

Field Name	Description
Prefetch	Available only when the consumption mode is Entry Browser. Set the value in field to attain optimum performance. The default value is -1 (prefetch all).

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

Result

The wizard adds the following properties to the simple event:

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



Configuring the Destination and Default Event Manually

You can add a destination to the 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.

Procedure

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

If you have not yet created the event, you can select the default event later. See [Assigning a Default Event to the Destination](#) for details about the simple event.

- Select the serializer/deserializer to be used:
`com.tibco.cep.driver.as.serializers.ActiveSpacesSerializer.`
- Enter the values for the rest of the fields as described in [Configuring an ActiveSpaces Destination and Event](#).

Assigning a Default Event to the Destination

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

Procedure

- 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.

Configuring Events to Work with ActiveSpaces Channel

You can configure events to interact with external ActiveSpaces metaspace.

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

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

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

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

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

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

Configuring Security and Authentication For ActiveSpaces Channel

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

Procedure

1. Open the ActiveSpaces channel file in the channel editor.
2. Select **Properties** or **Resources** as **Method of Configuration**.
The channel properties are displayed.
3. If **Properties** is selected, then enter the values for the following security fields (see [Configuration for ActiveSpaces Channels](#) for more details on the properties).
 - **EnableSecurity** (activates the other security fields)
 - **SecurityRole**
 - **Identity Password**
 - **PolicyFile**
 - **TokenFile**
 - **Credential**
 - **Domain**
 - **Username**
 - **Password**
 - **KeyFile**
 - **PrivateKey**



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

4. If **Resources** is selected, then configure the ActiveSpaces connection resource for the security. The security fields are same as mentioned in Step 3. See [ActiveSpaces Connection Reference](#) for more details on the properties.

Events

TIBCO BusinessEvents supports three sorts of events: Simple events (usually referred to just as events); time events, which are timers; and advisory events. This chapter explains how to use simple events.

A simple event defines an object that represents an occurrence of something. When the term "event" is used without the qualifier advisory or time, it refers to a simple event.

- *TIBCO BusinessEvents Getting Started* provides a practical introduction to events, including use of default events and default destinations.
- See *TIBCO BusinessEvents Architect's Guide*, [Channels, Destinations, and Events](#) for more detailed information .

Overview of Simple Events

A simple event defines an object that represents an occurrence of something.

When the term "event" is used without the qualifier advisory or time, event refers to a simple event.

For more information, see the following:

- *TIBCO BusinessEvents Getting Started* guide provides a practical introduction to events, including the use of default events and default destinations.
- *TIBCO BusinessEvents Architect's Guide* provides a detailed overview and information about Message Acknowledgment, Default Destinations and Default Events, Simple Events -- Time to Live and Expiry Actions.

Using Inheritance

Use of inheritance can simplify event configuration.

A child event inherits:

- All the parent event's properties.
- The parent event's expiry action (if set). However, an expiry action set in the child event overrides the parent event expiry action.



A parent event cannot have a payload. Also, all child events of an event with TTL=0 must also use TTL setting TTL=0 .

Events that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore the following restrictions apply:

- If two events are related by inheritance, you cannot create a new property in one with a name that is already used in the other.
- If two unrelated events have properties that share a name, you cannot create an inheritance relationship between the two events.

Working with Events in Rules

At runtime, event instances that are created using rules are not automatically asserted.

You must explicitly assert such events, for example using the `Event.assertEvent()` function.

Events that are created from incoming messages, on the other hand, are automatically asserted.

STIBCO BusinessEvents includes two functions that allow you to send simple events out to another application: `Event.sendEvent()` and `Event.routeTo()`.

- `Event.sendEvent()` automatically sends the event to its default destination.



The `Event.sendEvent()` function does not recognize whether the event is Put or Take. Use ActiveSpaces catalog function for Put or Take operations.

- `Event.routeTo()` takes a destination as an argument, ignoring the event's default destination. With `routeTo` you can direct an event to a destination on a different channel from the event's default destination. You can also override the properties of the destination, for example, the subject. You cannot, however, override the properties of the channel itself, for example, the network field in a Rendezvous channel.

You can use two methods to schedule simple events:

- Rule-based time events schedule the assertion of simple events.
- Scheduler functions schedule the sending of simple events to their default channels.

Adding a Simple Event

This section provides summary steps for adding a simple event. See [Simple Event Reference](#) for details on how to complete the values.

Procedure

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**.

3. Click **Finish**. You see the Simple Event Editor.
4. Complete the **Standard** tab and **Advanced** tab sections as explained in [Simple Event Reference](#).
5. Save the resource.



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.

Defining an XML Schema File in TIBCO BusinessEvents Studio

An Eclipse resource is available for this task.

Procedure

1. Right-click the folder where you want to add the resource and select **NewOtherXMLXML Schema** and click **Next**.
2. Enter a name and click **Finish**.
3. In the XML schema editor use the **Design** or **Source** tabs as preferred to define the schema. Refer to *Eclipse help* for details.

Importing an XML Schema File

An Eclipse resource is available for this task.

Procedure

1. From the **File** menu select ImportFile System.
2. At the Import from File System wizard, browse to a folder and select the desired schema file within that folder.
3. In the **Into Folder** field, specify the project folder where you want to put the file.
4. Set the overwrite and folder options as desired, then click **Finish**.

Using an XML Element in a Schema File in an Event Payload

To use an XML element in a schema file as the root element of the event payload, you will have to perform the following actions.

Procedure

1. In the event's **Advanced** tab, click the **Add (+)** button. You see a root node in the left panel.
2. In the **Content** field, select **XML Element Reference**.
3. In the **Schema** field click the **Browse** (binocular) button.
4. You see the Select a Resource dialog. Depending on the XML schema files available in the project, multiple entries may be grouped according to location (on the By Location tab) and namespace (on the **By Namespace** tab). Select one XML schema resource from either of these tabs.
5. In the Element panel, select the element you want to use as the root element for the payload. (You can also change this in the Payload section of the Event's **Advanced** tab.)
6. 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 might be able to see metadata properties. However, do not use them. Instead, use the settings and properties in the Domain Objects section of the CDD file as needed.

Wizard and Configuration (Standard Tab)

The Wizard and the Configuration section have the following fields.

Field	Global Var?	Description
Name	No	<p>Not shown as a field because it cannot be changed. The name appears in the Wizard, and in the title of the event.</p> <p>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.</p>
Description	No	Short description of the resource.
Inherits From	No	This event inherits from the event you select here. Leave blank if you do not want to use inheritance. For convenience, you can open the selected event resource by clicking the underlined label.
Time to Live	Yes	<p>Specify a numerical value and a time unit.</p> <p>Time units can be specified from the drop-down list available to the right of the Time to Live field. Choose one of: milliseconds, seconds, minutes, hours, and days. The default time unit is seconds.</p> <p>The numerical value is interpreted as follows:</p> <ul style="list-style-type: none"> • One or higher (>0): the event expires after the specified number of units elapse. • 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. <p>See Declaration and Expiry Action (Advanced Tab) .</p>
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 <code>Event.sendEvent()</code> function. For convenience, you can open the selected destination resource by clicking the underlined label.
Retry On Exception	No	<p>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:</p> <ul style="list-style-type: none"> • When this check box is selected, TIBCO BusinessEvents attempts to reprocess the event instance that failed and retries indefinitely with a delay of five seconds between retries. • When the check box is not selected TIBCO BusinessEvents does not attempt to reprocess the event instance that failed.

Properties (Standard Tab)

The Properties section has the following fields. Event properties generally map to incoming or outgoing message properties.

Field	Global Var?	Description
Name	No	<p>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 .</p> <p>Note:</p> <p>In addition to standard naming restrictions, do not begin an event property name with <code>_ns_</code> or <code>_nm_</code>. These have a special use. Also note that the property name cannot begin with the character <code>_</code>.</p> <p>For events used in JMS channels</p> <p>Names beginning with <code>_jms</code> or <code>jms</code> (case insensitive) are used only for JMS header properties. You can, however, use properties beginning <code>jms_</code> (case insensitive) for event properties.</p> <p>JMS Header Field Names shows the list of JMS header properties. Consult the JMS specification for more details.</p>
Type	No	<p>One of: <code>String</code>, <code>Integer</code>, <code>Long</code>, <code>Double</code>, <code>Boolean</code>, <code>DateTime</code></p> <p>Note:</p> <p>For properties of type <code>Double</code>, all NaN (Not a Number) values are converted to <code>0.00</code>.</p>
Domain	No	<p>Click the search button and select the domain model you want to use for this property. See Domain Models for details on adding domain models.</p>

Declaration and Expiry Action (Advanced Tab)

If the [Time to Live](#) field is zero or higher, define the action or actions to take when an event expires in the Expiry Action section.

If an event is explicitly consumed in a rule, TIBCO BusinessEvents does not execute the expiry action.

The editor in the Expiry Action section is the same as the Rule function editor. See [Rule Function Resource Reference](#) for details.

See *TIBCO BusinessEvents Architect'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:

Simple Event Payload Element Parameters (Sheet of)

Content/Parameter	Description
Complex Element	An element that contains other elements. This is like a structure in a programming language. The complex element can contain zero or more elements of other types, including other complex elements.
Name	The name of the element.
Cardinality	<p>Values for Cardinality:</p> <ul style="list-style-type: none"> • 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 <i>TIBCO ActiveMatrix BusinessWorks Palette Reference</i> for a complete list.
XML Element Reference	A reference to an element in a stored XML schema. See TIBCO Designer documentation for more information about XML schemas.
Cardinality	See Cardinality under Complex Element.
Schema	Stored XML schema that contains the element or type you want to reference.
Element	The element within the stored XML schema that you want to reference.
Attribute of Type	An attribute with a specified data type. You can specify a scalar data type (string, integer, and so on), you can reference an XML type, or specify the TIBCO ActiveEnterprise Any data type.
Name	The name of the element.
Cardinality	See Cardinality under Complex Element.
Type	The generic data type. For example, decimal or date/time.
Type	The specific data type. For example, float or month. Refer to the <i>TIBCO ActiveMatrix BusinessWorks Palette Reference</i> for a complete list.

Content/Parameter	Description
Sequence	A sequence of elements. Each item in the sequence is a structure of the sub-elements of this element.
Cardinality	See Cardinality under Complex Element.
Choice	A choice of elements. The data type of this element can be one of the sub-elements defined.
Cardinality	See Cardinality under Complex Element.
All	The data type of this element can be all 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.
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: <ul style="list-style-type: none"> • 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.

Attribute	Type	Returns
@extId	string	<p>The event's unique external ID. Optional.</p> <p>The value of the <code>extId</code> is set at creation time, for example, using the event's ontology function, or the <code>Event.createEvent()</code> function. The value cannot be changed after that.</p> <p>Note:</p> <p>The <code>extId</code> value (if set) must be unique across all objects in the cluster.</p> <p>Tip:</p> <p>You can use the property <code>Agent.AgentClassName.checkDuplicates</code> to check for duplicate <code>extIds</code> across the cluster.</p>
@ttl	long	<p>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.</p>
@payload	string	<p>The payload as a string value. See Payload (Advanced Tab) for more on specifying the payload in a SimpleEvent resource.</p>

Working With Time Events

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

Scheduled Time Events

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



Time events configured to repeat at intervals are not supported in multiple-agent (multi-engine) configurations. Rule-based time events, however, are supported.

You can configure a time event to repeat at a configurable time interval. For example, if you configure a time event to repeat every thirty seconds, then every thirty seconds TIBCO BusinessEvents creates a new time event of that type.

You can configure a repeating time event to create a specified number of events at each interval. The time interval begins during engine startup. See Engine Startup and Shutdown Sequence in *TIBCO BusinessEvents Administration* for specific details.

Rule Based Time Events

A rule based TimeEvent resource has only a name and description.

You can then use it in a rule to schedule a simple event to be asserted, using its ontology function, `ScheduleTimeEventName()` in a rule. 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 `ScheduleTimeEventName()` 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.

Adding a Time Event

See [TimeEvent Resource Reference](#) for information on how to complete the values.

Procedure

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



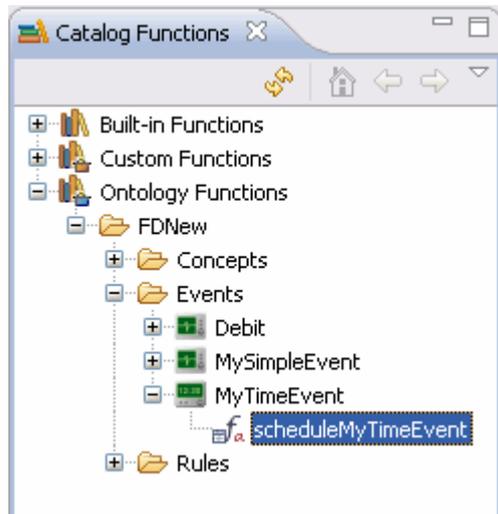
You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename** .

3. Click **Finish**. You see the Time Event editor.
4. In the **Type** field, select the type of time event you want to create and configure accordingly:
 - **ruleBased**: Select Rule Based. Use the event in a rule as explained in [Configuring a Rule Based Time Event in a Rule or Rule Function](#).
 - **repeat**: The event occurs at specified intervals. Select Repeat and configure the time event following guidelines in [TimeEvent Resource Reference](#).
5. Click **Apply** and save the resource.

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

Procedure

1. Open the rule editor for the rule or rule function where you want to use the rule-based time event.
2. Display the Catalog Functions view if it is not available: Select **Window > Show View > Other > TIBCO BusinessEvents > Catalog Functions** .
3. In the Catalog Functions view, expand Ontology Functions and drill down to the rule-based time event. Expand the rule-based time event and select its ontology function (`scheduleTimeEvent`).



4. Drag the time event's ontology function into the Actions area of the rule (or Body area of the rule function) and configure the parameters, following guidelines in [Rule Based TimeEvent Function Reference](#).
5. Save the resource.

TimeEvent Resource Reference



TimeEvent resources are used as timers. You can configure rule-based timers, which are scheduled in a rule, and repeating timers, which are scheduled in the TimeEvent resource. Use time events to trigger rules.



If you are working with a project imported from a release earlier than 5.0.0, you may see metadata properties. However, do not use them. Instead use the settings and properties in the Domain Objects section of the CDD file as needed.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Name	No	<p>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 .</p> <p>For events used in JMS channels</p> <p>Names beginning with <code>_jms</code> or <code>jms</code> (case insensitive) are used only for JMS header properties. You can, however, use properties beginning <code>jms_</code> (case insensitive) for event properties.</p>
Description	No	Short description of the resource.

Field	Global Var?	Description
Type	No	<ul style="list-style-type: none"> ruleBased: The time event is scheduled by a rule. repeat: Time events are created periodically. Complete the rest of the settings to define the period and number of events per interval.
Interval	No	<p>Used with repeating time events only. Specify the interval between time events using a numerical value and a time unit.</p> <p>TIBCO BusinessEvents creates the first time event immediately after the engine starts and then creates the next time event based on the interval.</p> <p>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).</p> <p>Time units available are milliseconds, seconds, minutes, hours, and days.</p>
Events per Interval	No	<p>Used with repeating time events only. Enter the number of events to create in each Repeat Every interval.</p> <p>Default is 1.</p>

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	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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	<p>The time scheduled for asserting an instance of this event into the Rete network.</p> <ul style="list-style-type: none"> 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 <code>ScheduleTimeEventName()</code> function's <code>delay</code> parameter. See Rule Based TimeEvent Function Reference.

Attribute	Type	Returns
@ttl	long	<ul style="list-style-type: none"> For repeating time events: always 0 (zero). For rule-based time events: specified using the ontology function's <code>ttl</code> parameter. See Rule Based TimeEvent Function Reference.

Rule-Based TimeEvent Function Reference

Signature

```
Event ScheduleTimeEventName(long delay, String closure, long ttl)
```

Domain

action

Description

For each rule-based time event you create, an ontology function is also created to enable you to schedule the assertion of an instance of the event in a rule action. The function name follows this format: `ScheduleTimeEventName`.

Parameters

Name	Type	Description
<code>delay</code>	long	The event is created (and asserted) the specified number of milliseconds after the rule action executes.
<code>closure</code>	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.
<code>ttl</code>	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 .

Returns

Type	Description
Event	An instance of the event type specified in the function name (<code>ScheduleTimeEventName</code>).

Using Scheduler Functions (Requires Cache OM)

You schedule simple events using scheduler functions. Events scheduled using these functions are sent to their default destination.

You might use the event scheduler functions instead of a time event in the following cases:

- If you need to send any event on a schedule, you must use the scheduler. If you schedule a memory-only simple event the schedule is persisted. If you schedule a memory only time event (using the time event's function, not the scheduler functions) then the schedule is not persisted.
- If you need to create many of schedules at one time, performance is better using the event scheduler functions.
- Because schedule management and event sending can be handled by a cache agent, fewer resources in the inference agents are required compared to time events. Additionally, with backing store enabled the schedules are only loaded into memory a batch at a time, reducing the total memory requirement compared to time events.

Creating a Scheduler

Procedure

1. Create a scheduler using the function `Cluster.createScheduler()`.

You can do this, for example, in a startup rule function, or other rule function or rule:

```
void createScheduler(String schedulerName, long pollInterval, long refreshAhead)
```

schedulerName A unique ID for the scheduler you are creating.

pollInterval TIBCO BusinessEvents checks the scheduler cache every *pollInterval* milliseconds, for scheduler work items whose *scheduledTime* falls in the current interval.

refreshAhead Time in milliseconds (into the future) used to pre-load the scheduled events from the backing store.



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

For example:

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

2. Schedule the Event to be Sent

Result

You use the function `Cluster.scheduleEvent()` to schedule an event to be sent at a certain time, using a scheduler you created earlier. You can do this in a rule or rule function as needed:

```
void scheduleEvent(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)
```

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](#)).

For more information, refer to the *TIBCO BusinessEvents Architect's Guide*, [Events](#)

Procedure

1. To add an advisory event to a rule, open the rule editor for the rule where you want to use the advisory event.
2. Perform one of the following actions:
 - In the form-based editor, click the plus icon in the Declarations panel and select Advisory Event from the list of resources.
 - In the source editor, add a line like the following in the `declare` block:


```
AdvisoryEvent a;
```

In both cases you can change the alias as desired (from `a` to something else).
3. Save the resource.

Result

The `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](#).

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.
@ruleUri	String	The URI of the current rule when the <code>AdvisoryEvent</code> was asserted.
@ruleScope	Object[]	The scope object of the current rule when the <code>AdvisoryEvent</code> was asserted.
@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

Name	Type	Description
@type	String	Type of advisory within the category. See Attributes Used for Each Type of Advisory Event for details.
@message	String	Message content depends on the type of advisory event. See Attributes Used for Each Type of Advisory Event for details.

Attributes Used for Each Type of Advisory Event

Description	@category	@type	@message
Exceptions arising from user code	Exception	The Java class name of the exception	The stack trace and a message (if the exception includes a message)
TIBCO ActiveMatrix BusinessWorks Activity Failure	Engine	INVOKE BW PROCESS Indicates a failure or cancellation or timeout of the ActiveMatrix BusinessWorks process.	The error message from the failed ActiveMatrix BusinessWorks process, or the timeout message.
Engine is activated	Engine	engine.primary.activated Indicates the engine has been activated.	Engine <i>EngineName</i> activated.
Hot deployment	Deployment	deployment.hotdeploy.success deployment.hotdeploy.fail	Hot deployed project <i>ProjectPath</i> Failed to hot deploy project <i>ProjectPath</i>

Event Property

Event properties can be used in the rule language grammar by following certain rules.

This is the syntax for accessing an event property:

```
eventName.propertyName
```

For example:

```
String x = eventA.customer;
```

where *eventName* is the identifier of the concept instance and *propertyName* is the name of the event property that you want to access.

Concepts

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

Adding a Concept

Procedure

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. Type the values in the **Concept Name** and **Description** fields.



You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename**.

3. Click **Finish**.
4. In the Concepts Editor:
 - Configure the remaining fields as explained in [Wizard and Configuration Tab](#).
 - As needed, add and configure properties as explained in [Properties](#).
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 *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.

Creating a Containment Relationship

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

Procedure

1. In the container concept resource's Property section, add and configure a property of type **ContainedConcept**.
2. Configure the property as follows:

Name: Provide a helpful name. For example, "wheels" could indicate that this concept is contained by a "car" concept (in an appropriate project).

Multiple: Select the check box if this property is an array.

Policy: Changes Only or All Values.

History: The number of historical values to keep.

Creating a Reference Relationship

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

Procedure

1. In the **Property** tab of one of the concept resources, create a property of type `ConceptReference`.
2. Configure the property as follows:

Name: Provide a helpful name such as `is a lineitem of` to express the relationship.

Multiple: Select the check box if this property is an array.

Policy: Changes Only or All Values.

History: The number of historical values to keep.

See *TIBCO BusinessEvents Architect's Guide* for more information on this topic.

Enabling the Concept Reference Serialization in Concept to JSON

Reference concepts are not serialized by default. Set a property in the parent concept metadata to expand them during serialization.

Add the metadata property `tibco.be.schema.ref.expand` for all the concept with the `ConceptReference` property to `true`. This enables the serializer to convert the referenced concept to JSON. This property works for In-memory as well as Grid storage type.

If there are multiple level concept reference, then add this property to all the concept with the `ConceptReference` property.

Procedure

1. In TIBCO BusinessEvents Studio, open the concept for editing.
2. In the concept editor, click **Enable Metadata Configuration** () on top-right corner to enable the metadata properties for editing.
The **Metadata** section is now enabled.
3. Expand the **Metadata** section and click **Add** to add a new property.
A new property with the system-generated-name is added.
4. Edit the property name to `tibco.be.schema.ref.expand` and set the value to `true`.
In runtime, the serializer read this option and convert the referenced concept to JSON.
5. Save the project and rebuild the EAR.

Editing or Deleting Concept Relationships

You cannot edit or delete the relationship using the concept view graphical user interface. Instead, work with the concept properties directly in the Concept Editor.

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 might be able to see metadata properties. However, do not use them. Instead use the settings and properties in the Domain Objects section of the CDD file as needed.

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. This field appears on the New Concept Wizard. The name then appears in the title of the concept editor.
Description	No	Short description of the resource. This field appears on both the New Concept Wizard and in the concept editor.
Inherits From	No	If you want this concept to inherit all the properties of another concept, browse to and select that concept. Concepts that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore the following restrictions apply: <ul style="list-style-type: none"> • 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.
State Models	No	State models that are owned by this concept. Models can be added and removed as needed. Requires TIBCO BusinessEvents Data Modeling.
Auto Start State Model	No	If selected, when a concept is asserted at runtime, its main state machine (if any) is started. If not selected, then state machines are started using this function in rules: <code>Instance.startStateMachine()</code> The function has a parameter specifying whether child concepts' state machines are also started. For contained concepts, the value of the Auto Start State Machine is not relevant and is not selected. Requires TIBCO BusinessEvents Data Modeling.

Properties

The Properties tab has the following fields:

Field	Global Var?	Description
Name	No	<p>The property name.</p> <p> The property name cannot begin with the character <code>_</code>.</p>
Type	No	<p>Any of the following types: <code>String</code>, <code>Integer</code>, <code>Long</code>, <code>Double</code>, <code>Boolean</code>, <code>DateTime</code>, <code>ContainedConcept</code>, <code>ConceptReference</code></p> <p>When you create a property of type <code>ContainedConcept</code>, 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.</p> <p> A concept cannot contain itself. For instance, for a concept A we cannot add a contained property of type A.</p> <p>When you create a property of type <code>ConceptReference</code> you are creating a property that references another concept.</p> <p>See <i>TIBCO BusinessEvents Architect's Guide</i> for more details.</p> <p>Note: For properties of type <code>Double</code>, when a backing store is used, all NaN (Not a Number) values are converted to <code>0.00</code>.</p>
Multiple	No	<p>Select the Multiple check box if this property is an array.</p> <p>Consider, for example, an <code>Order</code> concept: In most cases, an <code>Order</code> concept would allow only one value for the <code>customer</code> property but multiple values for the <code>line_item</code> property. Selecting the Multiple check box creates a property array.</p>
Policy	No	<p>TIBCO BusinessEvents can record historical values using either of these policies:</p> <p>Changes Only TIBCO BusinessEvents records the value of the property every time it changes to a new value.</p> <p>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.</p>

Field	Global Var?	Description
History	No	<p>Determines if historical values are stored, and if so how many. The default maximum value is 32767.</p> <p>Zero (0): TIBCO BusinessEvents does not store historical values for the concept. It stores the value without a time and date stamp</p> <p>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 <i>TIBCO BusinessEvents Architect's Guide</i> for more details.</p> <p>Note: Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.</p>

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	<p>The concept instance's unique external ID. Optional.</p> <p>The value of the <code>extId</code> is set at creation time (for example, using the <code>Instance.createInstance()</code> function) and cannot be changed after that.</p> <p>Note: The <code>extId</code> value (if set) must be unique across all objects in the cluster.</p> <p>Tip: You can use the property <code>Agent.AgentClassName.checkDuplicates</code> to check for duplicate <code>extIds</code> across the cluster.</p>

Entity	Attributes	Type	Returns
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.)
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: <code>acc.Balance@isSet</code> . It indicates if a property is assigned or not. If a property is assigned, then <code>@isSet</code> returns <code>true</code> .

Adding a Scorecard

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

Procedure

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. Type in the values for the Scorecard Name field and Description fields.



You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename** .

3. Click **Finish**.
4. In the Scorecard Editor, add and configure properties as explained in [Concept Resource Reference](#).
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](#) for details about scorecard properties

Accessing a Concept PropertyAtom

Access to a concept *PropertyAtom* is defined with the syntax.

Syntax for accessing a concept property :

```
instanceName.propertyName
```

where *instanceName* is the identifier of the concept instance, and *propertyName* is the name of the concept property that you want to access.

For example to get the current value of the cost *PropertyAtom*:

```
int x = instanceA.cost;
```

For example, to set a value with the current system timestamp:

```
instanceA.cost = value;
```



If the history size is 0, TIBCO BusinessEvents does not record a timestamp.



Rule dependency and rule condition re-evaluation is performed only on concept properties.

This re-evaluation is not performed on an attribute such as @parent.

Get and Set PropertyAtom Value With User-Specified Time

PropertyAtom can be set for the rule language grammar.

You can get and set *PropertyAtom* values as follows:

- Specify a time and get the *PropertyAtom* value stored in the history at that time using one of the standard functions:

```
type Instance.PropertyAtom.gettype(PropertyAtom propertyName, \
                                     long time)
```

where *type* is the type of the *PropertyAtom*, *propertyName* is the name of the *PropertyAtom*, and *time* is the time from which you want to retrieve the value.

- Set a value in the *PropertyAtom* History using one of the standard functions:

```
Instance.PropertyAtom.settype(PropertyAtom propertyName, \
                                type value, long time)
```

where *type* is the type of the *PropertyAtom* and the type of the new value, *propertyName* is the name of the *PropertyAtom*, *value* is the value to store in the ring buffer, and *time* is the timestamp for the new entry.

TIBCO BusinessEvents manages these requests as follows:

- If the ring buffer has vacancies, TIBCO BusinessEvents inserts the new entry into the correct place based on its timestamp, shifts the older values out one place, and returns True.
- If the ring buffer is full, and the new value has a more recent timestamp than the oldest value, TIBCO BusinessEvents inserts the new value into the correct place, shifts older values if necessary, drops the oldest value, and returns True.

- If the ring buffer is full, and the new value has a timestamp that is older than the oldest value in the ring buffer, TIBCO BusinessEvents does not insert the new value into the ring buffer, and it returns False.

Concept Property Arrays

To work with concept property arrays you need to follow certain rules.

Accessing a Concept Property Array

This is the syntax for accessing a concept property array:

```
instanceName.propertyName
```

where *instanceName* is the identifier of the concept instance, and *propertyName* is the name of the concept property that you want to access.

Accessing a Value in a Concept Property Array

To access a value in a property array, identify the position in the array of the value as shown:

```
instanceName.propertyName[indexPosition]
```

For example:

```
String x = instanceA.lineItem[0];
```

This gets the current value of the first property atom in the array, *lineItem*, and assigns it to the local variable, *x*.



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);
```

Array identifier numbers are positional. When history is not tracked and you delete multiple items in an array (using a for loop), delete higher position numbers before lower position numbers to ensure the correct entries are deleted. For example, if you want to delete items in positions 1, 2, and 4, delete them in this order: 4, 2, and then 1.

The array entry that held a deleted concept is removed, reducing the array size by one, and reducing by one the index of every entry in the array at a higher index than the deleted one. If you delete entries with lower position numbers first, the remaining lower numbers are then occupied by different items.

For example, if you delete the item in position 1, then the item that was in position 2 is now in position 1.

Checking if Multiple Property of a Concept is Empty

If you have a concept (for example, `CustomerConcept`) which contains multiple property (for example, `address`) and you did not pass any value to the address property when creating the concept then the following check always returns `false`:

```
customerConcept.address == null
```

The reason for this is that the multiple check on a concept property (simple property, `ContainedConcept`, or `ConceptReference`) makes the property an array. Even if you do not add any property, `ContainedConcept`, or `ConceptReference` to the multiple property, the property is an array with the length zero but not `null`.

Thus, to resolve this, for a multiple property (simple property, `ContainedConcept`, or `ConceptReference`), check the length of the property (array). The length zero indicates there is no simple property, `ContainedConcept`, or `ConceptReference`.

Rules and Functions

If you are using the source editor, adapt these instructions that focus on the form-based editor and mention the equivalent settings in the source editor.

Rule Form Editor

Rule: ApplyDebit

Configuration

Description:

Priority: 1 (Highest)

Rank:

Forward Chain:

Declaration

+ Add - Remove ↑ Up ↓ Down

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

Conditions

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

Actions

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

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 account <"
+account@extId + ">");
}
    Event.consumeEvent(debit);
}
}

```

Adding a Rule

Procedure

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** .

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 select or deselect the **Initially show 'Form'** tab check box 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](#) for details.
6. To disable forward rule chaining (the default behavior) uncheck the Forward Chain check box. (In the source editor set `forwardChain = false`.)
7. In the Declarations section (equivalent to the `declare` statements in the source editor), drag an ontology entity into the section, or perform the following actions.
 - a) Click **Add** to add 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](#) for more information on working in the rule editor.

9. In the Actions section (equivalent to the then statements in the source editor), write action statements (in the TIBCO BusinessEvents rules language).

10. Save the resource.

Rule Editor Reference



The rule editor and rule function editor are similar. This section focuses on the form-based rule editor. You can adapt the information in the Area and Property section to apply to the different blocks of code in the source editor.

Property	Description
Configuration Section	
Name (Wizard only)	The name to appear as the label for the resource. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements .
Description (Editor and Wizard)	Short description of the resource.
Priority (Editor and Wizard)	Specify a value between 1 and 10, where 1 is the highest priority and 10 is the lowest priority. Rules with a higher priority execute before rules with a lower priority. See also Rank. Only set priority or rank where there is a reason to control the order of rules. As a general practice, it is more efficient to let the engine determine the order of rule firing.
Rank	Specifies a rule function that controls the order of execution of rules with the same priority. Validity must include Condition (that is, do not select Action Only as the validity.) See Using Priority and Rank to Control Order of Rule Execution for details. Default is 0.0.

Property	Description
Forward Chain	Determines if the rule is used in forward chaining. If the check box is not selected, changes made by this rule won't trigger other rules. By default the check box is selected
Declaration Section	
Term	A concept or event type in the project that you will use in your rule. Types you add to the declaration define the scope of the rule. For example, a concept definition such as /Concepts/Accounts/CheckAccount. It is not necessary to add scorecards in the declaration in order to use them in the rule.
Alias	A name used to refer to the scope element in the body of the rule. You can change the alias. Like resource names, aliases must be valid identifiers See Identifier Naming Requirements .
Conditions Section	
	Each line in the Conditions area is a single expression that evaluates to <code>true</code> or <code>false</code> . 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 <i>TIBCO BusinessEvents Architect's Guide</i> .)
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.

Adding a Rule Function

See [Rule Function Resource Reference](#) for details on completing values.

Procedure

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), select the **Virtual** check box.



You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor** > **Rename**. See [Element Refactoring Operations](#) for more details.

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 select or deselect the desired check box: **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. Select 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 perform the following actions.
 - 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, select the **isArray** check box. (You can specify a variable array in the source editor in the usual way, for example, `int[] myArr`.)
 - d) If the type you select is an ontology type, in the lower half of the dialog, select a resource from the filtered ontology tree.

e) Click **OK**.

Your selection appears in the list. TIBCO BusinessEvents assigns an alias to it. You can edit the alias.

Add more entities as needed.

9. Add more arguments as needed, and use the up and down arrows to order the arguments as needed.

10. In the Body section (Body statements in the source editor), use the TIBCO BusinessEvents rule language to implement the function. (Virtual rule functions have only a signature, and no implementation at design time.)

See [Using Variables and Functions in the Rule Editor](#) for more information on working in the rule editor.

11. Save the resource.

Rule Function Resource Reference



Rule Function resources enable you to write rule functions that you can use in rules, as startup and shutdown actions, and as preprocessors.

Virtual rule functions are decorated with a V.

Property	Description
Configuration Section	
Name (Wizard only)	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements .
Description (Editor and Wizard)	Short description of the resource.
Return Type (Editor and Wizard)	If the rule function returns a value, specify the Return Type, otherwise leave set to void.
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 <i>TIBCO BusinessEvents Decision Manager User's Guide</i> .
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	<p>Specifies where the rule function can be used. Possible values are as follows:</p> <p>Action Indicates that this rule function can be used only in the Action block of a rule.</p> <p>Action and Condition Indicates that this rule function can be used in the Action and Condition blocks of a rule.</p> <p>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).</p> <p>Note: Only Action rule functions can be used as startup rule functions or shutdown rule functions.</p>
Scope Section	
Term	<p>The type of the argument. Arguments and return type can be any of the following, including arrays of these datatypes:</p> <ul style="list-style-type: none"> • Primitive, that is any of: <code>String</code>, <code>int</code>, <code>long</code>, <code>double</code>, <code>boolean</code>, <code>DateTime</code>, <code>Object</code> • Concept • Event • Specific type of Concept • Specific type of Event <p>The Object data type is used to pass parameters between standard and user-defined functions and external Java sources.</p>
Alias	Each argument requires a type and an alias. Names must be valid identifiers.
Body Section	
List of statements that will be executed when the rule function executes.	

Using Variables and Functions in the Rule Editor

This section provides some tips on working in the rule editor. The rule editor is used for TIBCO BusinessEvents rules, rule functions, and state machine rules.

Using Catalog Functions in the Rule Editor

To use catalog functions in an editor choose one of the following methods.

Procedure

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

- Type the function name, starting with the folder path to the function.
Do not enter the catalog name (which appears in the list of catalog functions in the Catalog Functions view, and looks like a top-level folder). Type a period at the end of the folder name. A popup window shows all folders and functions within the folder whose name you typed. For example type `Database.` to see a list of all functions in the Database folder.
 - Open the Catalog Functions view. To open the view click as follows:
Window > Show View > Other > TIBCO TIBCO BusinessEvents > Catalog Functions.
Drill down on categories within the catalog to expand to lists of functions and drag the desired function to the rule editor.
2. Provide parameter values as indicated by the tooltip. You can hover the cursor over a function to display a tooltip showing the function's arguments. You can also see the tooltip contents in the online reference, TIBCO BusinessEvents Functions Reference.

Result

See *TIBCO BusinessEvents Architect's Guide* for descriptions of the various function catalogs, and an explanation of the decorations that appear on many function names. For information about configuring mapper functions see [Using the Function Argument Mapper](#).

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 [Global Variables](#) for more details about global variables.

Using the Function Argument Mapping Wizard

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

See [Mapping and Transforming Data](#) for a reference to using the Function Argument Mapping wizard.



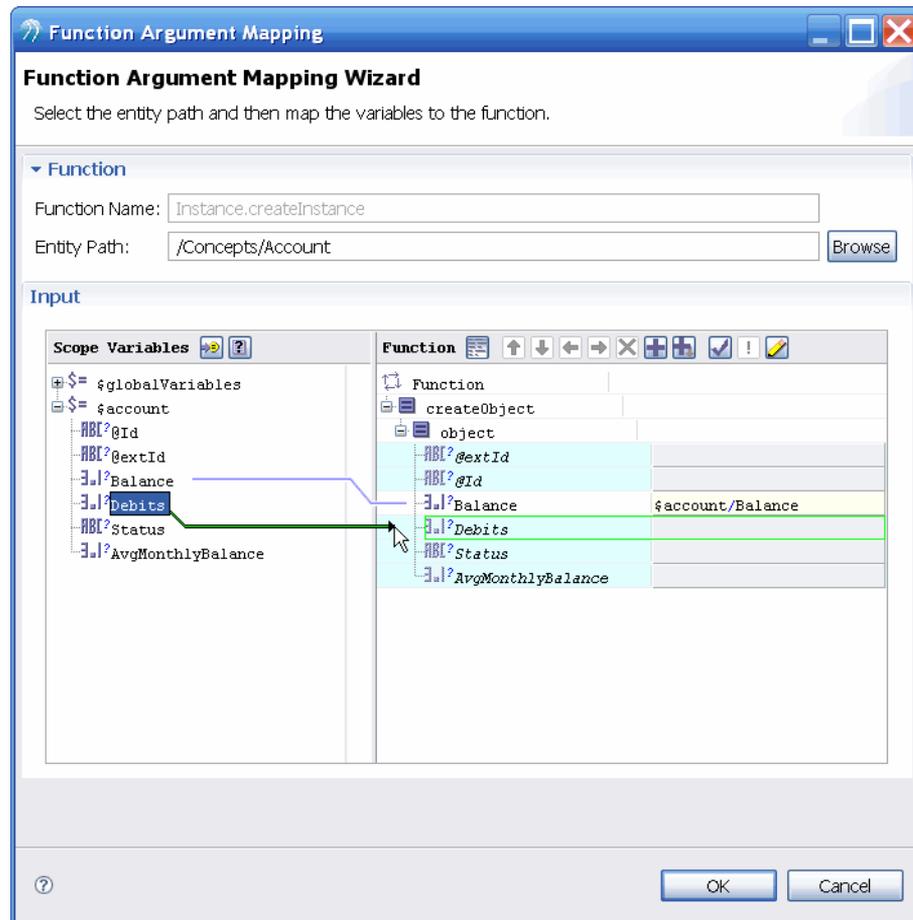
In the TIBCO BusinessEvents language, array indexes start from zero (0). However, in XSLT and XPath languages, they start from one (1). It's important to remember this difference when using the rule language in the rule editor, and when working in the XSLT mapper and the XPath builder.

Procedure

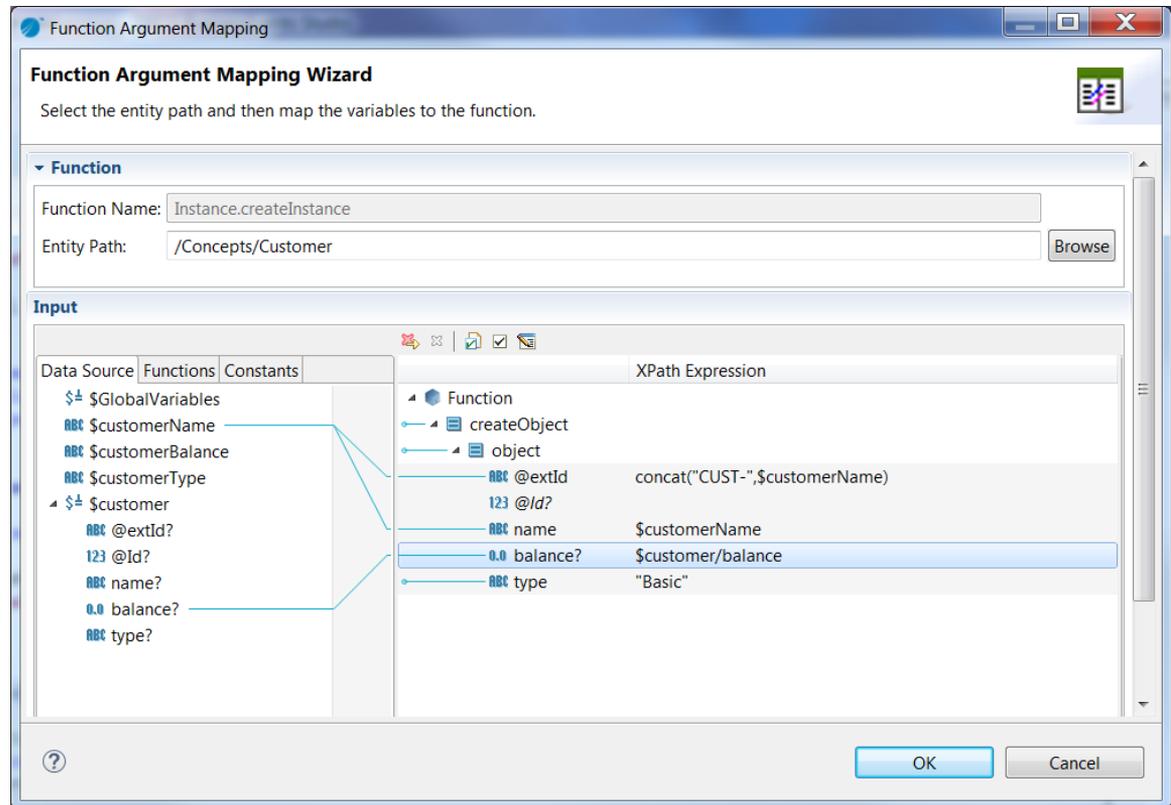
1. Open the Function Argument Mapping wizard in either of the following ways:
 - In the rule editor, type the category of function you want to use and select a mapper function from the list of functions in that category. The function argument area contains the text `"xslt://"`. Ctrl+click the text `"xslt://"` to open the Function Argument Mapping wizard.
 - Type the name of a mapper function category into the **Actions** or **Conditions** areas, then type an open parenthesis (`"(`) TIBCO BusinessEvents displays `"xslt://"`. Ctrl+click the text `"xslt://"` to open the Function Argument Mapping wizard.
 - Type the entire string to specify the function category path and name, followed by (`"xslt://"`). Then control-click the text `"xslt://"` to open the Function Argument Mapping wizard.

2. In the Function Argument Mapping wizard, under the Function section, take appropriate action. For example, if you are using `Instance.createInstance()`, click the **Browse** button to the right of the **Entity Path** field and select a concept type so you can create an instance of it.
3. Under the **Input** section, on the left side, TIBCO BusinessEvents displays a list of variables associated with the project, and on the right, a list of properties for the selected resource type.
4. Drill down and expand the lists on both sides to expose the variables and properties.
5. Drag variables from the left to the appropriate property on the right.

The Function Argument Mapping Wizard in XPath 1.0 Project



The Function Argument Mapping Wizard in XPath 2.0 Project



6. (XPath 1.0) If you want to define the arguments using more complex logic, type the code or click the **XPath Formula Builder**  icon to use the XPath Formula Builder. See [XPath Formula Builder](#).
7. (XPath 2.0) If you want to define the arguments using more complex logic, type the code in the **XPath Expression** column or click the **Show/Hide Edit Tab**  icon to display the **Edit Statement** tab. On the left side, click the **Functions** tab to see the available mapper functions and then drag and drop the functions to the **XPath Expression** textbox. See [XPath Formula Builder](#).
8. Click **OK**.

Using Priority and Rank to Control Order of Rule Execution

For each RTC, the rule agenda is sorted by priority and then within priority by rank, for those rules that use the same ranking. Use of priority and rank is optional. You can also use priority without using rank.

In the rule's **Rank** field (or rule attribute, in the source view), you specify a rule function that returns a double. The larger the return value, the higher the ranking. You can specify the same rule function in different rules to perform ranking across tuples of those rules. Here are the requirements:

- The rule function must have a Validity setting that includes Condition (that is, do not set it to Action Only).
- You can assign the same rule function to different rules as long as the following is true:
 - The scope of the rule function includes only parameters found in all the rules that use the same function. Rank rule function scope must match the rule declaration exactly and in the same order (As with rules, primitives are not allowed.) For example, if the rule has EventA and ConceptB as the scope, the rule function must also have EventA and ConceptB (in that order) as the only two parameters.
 - The parameters must be used in the same order as they appear in the rule declaration.

- The rule function must return a double value. (The default value for the **Rank** field is 0.0.)

Examples

For example, suppose two rules declare `CustName` and `SupportLevel`. You assign the same rule function to both rules. The function returns 3.0 for Gold support level, 2.0 for Silver, and 1.0 for Bronze. As a result, among 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	Priority	Rank
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 depending on the effect of rule actions.)

Rule	Priority	Rank
Rule 2 (Tuple A, B)	Priority: 1	Rank: 1.0
Rule 2 (Tuple A, C)	Priority: 1	Rank: -1.0
Rule 2 (Tuple A, D)	Priority: 1	Rank: -2.0
Rule 3 (Tuple A)	Priority: 4	Rank: 0.0
Rule 1 (Tuple X)	Priority: 5	Rank: 10.0

Reverse Order Example

If you want the rule with the smallest return value to be ranked highest, multiply by negative one (-1.0) to reverse the size of the values. For example suppose you want rules that were asserted earlier to 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.

Using the Quick Fix Feature

Procedure

1. Type the code in the rule editor as if the entity, property, or rule function you want to use already exists. See [Quick Fix Feature Options](#) for some examples.

You see a lightbulb icon in the left margin on any line where there are unknown references that you can configure using Quick Fix.

2. To use the available Quick Fix options, click the lightbulb icon or press Ctrl+1. The options are explained in [Quick Fix Feature Options](#):

Result

Quick Fix Feature Options

Unknown Reference	Quick Fix Editor
Unqualified name. Example: <code>SomeName s;</code>	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: <code>SomeName()</code>	A link to the rule function creation wizard appears. Click it to open the wizard and define the rule function.
Qualified name. Examples: <code>Rules.SomeName();</code> <code>Concepts.SomeName s;</code>	A Quick Fix option appears so you can automatically create the entity or rule function in the given folder. The folder must already exist in the project.
Qualifier is a concept or event and reference is unknown. Example: <code>Concepts.Person person;</code> <code>person.someProperty = "123"</code>	A Quick Fix option appears with the name: Create a property definition in the entity ' <i>entityName</i> '. Click the option to open the New Property Definition dialog, and configure the details of the property. In the example, the <code>someProperty</code> property does not already exist as a property of <code>Concepts.Person</code> .

The following table explains the arguments.

Argument	Notes
Object <i>txns</i>	<p>You can turn this argument into an <code>Object[]</code> within the rule function, for example:</p> <pre>Object[] array = txns;</pre> <p>The resulting array can be used to obtain useful data:</p> <ul style="list-style-type: none"> • <code>array[0]</code> is a <code>Concept[]</code> containing all the <code>Concept</code> objects that: <ul style="list-style-type: none"> – have been created in the current RTC, – and have not been deleted. • <code>array[1]</code> is a <code>Concept[]</code> containing all the <code>Concept</code> objects that: <ul style="list-style-type: none"> – existed before the current RTC, – have been modified in the current RTC, – and have not been deleted. • <code>array[2]</code> is a <code>Long[]</code> containing the id of all the <code>Concept</code> objects that: <ul style="list-style-type: none"> – existed before the current RTC, – and have been deleted in the current RTC. • <code>array[3]</code> is a <code>SimpleEvent[]</code> containing all the <code>SimpleEvent</code> objects that: <ul style="list-style-type: none"> – are going to be added to the cache. • <code>array[4]</code> is a <code>Long[]</code> containing the id of all the <code>Event</code> objects that: <ul style="list-style-type: none"> – existed before the current RTC, or arrived and started the RTC, – and have been deleted in the current RTC.
int <i>errorType</i>	<p>Value can be one of:</p> <ul style="list-style-type: none"> -1 for errors happening during database operations. -2 for errors happening when sending an event.
int <i>errCode</i>	<p>The error code, which is dependent on <code>errorType</code>:</p> <ul style="list-style-type: none"> • For database related errors: <ul style="list-style-type: none"> – The error code as returned by <code>SQLException</code>, 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 <i>errMsg</i>	<p>The associated exception message.</p>
long <i>retryCount</i>	<p>The number of times this transaction has been retried before the present call to the callback.</p>

Tips for Working in the Rule Editor

Here are some tips for working with the Rule Editor

Switching between Form and Source Editors

You can freely switch between form and source editors for rules and rule functions. In each case the editors stay synchronized with the latest changes.

You cannot switch from the source editor to the form editor if there are any syntax or resolution errors in your code.

The Priority setting

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.

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 do this...	Do this...
To Use Content Assist to Complete Values	<p>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.</p> <p>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.</p>
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	<p>To jump to the location where an item is defined, you can use two methods.</p> <ul style="list-style-type: none"> • 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. <p>For example, you would jump from an alias to the declaration, and from an entity or entity property to the entity's editor.</p>

Event Preprocessors

Event preprocessors are rule functions with one argument of type simple event.

Event preprocessors are not used for time or advisory events, and they are multithreaded.

An event preprocessor is assigned to a destination and acts on all events arriving at that destination. Event preprocessors perform tasks after an incoming message arrives at the destination and is transformed into a simple event, but before it is asserted into the Rete network (if it is — events can be consumed in the event preprocessor).

- If you are using Cache Only mode, take care when designing rules that execute as a result of a time event. For example, a rule that has a join condition using a time event and a concept would not execute if the concept is not loaded in the Rete network and so should not be used in an event preprocessor if the concept is configured as Cache Only mode.
- Events consumed in a preprocessor are acknowledged immediately (and not after the post RTC phase).

You can aggregate events, edit events, and perform other kinds of event enrichment in a preprocessor. You can also use preprocessors as explained below.

You must set locks in the preprocessor when concurrency features are used to protect concept instances during RTC. Locking ensures that updates to concept instances during an RTC do not conflict with



another set of updates to the same concept instances in another RTC. Locks are released at the end of the RTC.

If you are using the Cache Only mode for any entities, you must also load the relevant entities from the cache using an event preprocessor.

You can also use preprocessors to improve performance by avoiding unnecessary RTCs in the inference engine. For example you can consume events that are not needed. Another way to use the preprocessor for efficient processing is to transfer an event's contents to a new concept that is not processed by the agent's set of locally active rules. Such a concept is automatically asserted, does not trigger rules, and is saved into the cache (depending on OM configuration) where it is available for processing by any agent as needed.

Transaction Error Handler Rule Function

You can create a transaction error handler callback rule function that enables you to identify which post RTC transactions failed or which events were not sent out during the post RTC phase. The transaction error handler rule function is invoked each time a database transaction exception occurs, and each time a send event exception occurs.

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).

Arguments are explained in [Transaction Error Handler Rule Function Reference](#).

Transaction Error Handler Rule Function Reference

This table contains reference for the Transaction Error Handler rule function.

Argument	Notes
Object <i>txns</i>	<p>You can turn this argument into an <code>Object[]</code> within the rule function, for example:</p> <pre>Object[] array = txns;</pre> <p>The resulting array can be used to obtain useful data:</p> <ul style="list-style-type: none"> • <code>array[0]</code> is a <code>Concept[]</code> containing all the <code>Concept</code> objects that: <ul style="list-style-type: none"> – have been created in the current RTC, – and have not been deleted. • <code>array[1]</code> is a <code>Concept[]</code> containing all the <code>Concept</code> objects that: <ul style="list-style-type: none"> – existed before the current RTC, – have been modified in the current RTC, – and have not been deleted. • <code>array[2]</code> is a <code>Long[]</code> containing the id of all the <code>Concept</code> objects that: <ul style="list-style-type: none"> – existed before the current RTC, – and have been deleted in the current RTC. • <code>array[3]</code> is a <code>SimpleEvent[]</code> containing all the <code>SimpleEvent</code> objects that: <ul style="list-style-type: none"> – are going to be added to the cache. • <code>array[4]</code> is a <code>Long[]</code> containing the id of all the <code>Event</code> objects that: <ul style="list-style-type: none"> – existed before the current RTC, or arrived and started the RTC, – and have been deleted in the current RTC.
int <i>errorType</i>	<p>Value can be one of:</p> <ul style="list-style-type: none"> -1 for errors happening during database operations. -2 for errors happening when sending an event.

Argument	Notes
<code>int <i>errCode</i></code>	<p>The error code, which is dependent on <code>errorType</code>:</p> <ul style="list-style-type: none"> • For database related errors: <ul style="list-style-type: none"> – The error code as returned by <code>SQLException</code>, 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.
<code>String <i>errMsg</i></code>	The associated exception message.
<code>long <i>retryCount</i></code>	<p>The number of times this transaction has been retried before the present call to the callback.</p> <p>Handling Post-RTC action errors:</p> <p>When Post-RTC actions fail, such as acknowledging or sending a message, action handlers try only 10 times with 500 milliseconds sleep. The following parameter scan be configured:</p> <ul style="list-style-type: none"> • <code>be.engine.txn.action.retrycount</code>, default is 10. • <code>be.engine.txn.action.sleepime</code>, default is 500 milliseconds. <p>Handling Post-RTC database transaction errors:</p> <p>In case of database exception, the numer of tries and waits between each tries are handled by the following configuration parameters:</p> <ul style="list-style-type: none"> • <code>be.engine.txn.database.retrycount</code>, default is maximum integer. • <code>be.engine.txn.database.sleepime</code>, default is 5000 milliseconds. <p>After the number of tries are exhausted, the transaction is aborted.</p> <p>You can also perform additional actions by registering an error handler fule function as before:</p> <p>See <code>be.txn.error</code> <code>function=projectPathToErrorHandlerRuleFunction.</code></p> <div style="border-left: 1px solid black; padding-left: 10px; margin-top: 10px;">  <p>When the number of tries are exhausted and the transactions are aborted, it is likely that the cache and database will be in an inconsistent state. Practice care when setting a limit to the number of retries. Logs should be also monintored for database related exceptions and immediate action should taken as soon as possible. It is also advised to disable parallel operations if a limit is set to the number of retries. In this case, multiple transactions are committed to the database together and a single database failure can affect multiple of them.</p> </div>

Rule Template and Rule Template View

A rule template is a specialized type of rule, and a rule template view puts a user-friendly interface around the rule template for use in Web Studio.

Rule templates and rule template views enable non-technical Web Studio users to define executable rules (called *business rules*) within limits defined in the rule template.

Adding a Rule Template View

First, add a rule template. One rule template can be associated with multiple views. .

Procedure

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



You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename** . See [Element Refactoring Operations](#) for more details.

4. Click **Finish**. You see the Rule Template View editor.
5. Add or edit the description.
6. As needed, browse and select a different rule template
7. As needed, click the Rule Template link to edit the selected rule template, for example to add new bindings.

Bindings defined in the rule template appear in the Bindings section of the **Presentation** section.

8. In the **Presentation** tab, use plain text or HTML to define an input form. Drag and drop the bindings into the Presentation section, or type them.

The syntax for a binding is as follows (the closing tag is required):

```
<binding id="bindingName"></binding>
```

You can click the **Preview** tab to see how the view will appear in WebStudio.

9. Optional: Click the **HTML** tab and click **Browse** to select an HTML file, which can be used for the input form.

Use the binding ID in the HTML file for creating the text box or dropdown for the binding. The syntax for the binding is same as specified earlier step. You can apply styling and javascript to these HTML view bindings.



Currently, when you click on the Preview tab, the preview of the HTML file is not displayed.

10. Save the resource.

The Rule Template Editor

Comparing similar sections of the Rule Template editor and the Rule editor highlights the unique features of rule templates.

Configuration Section

The Configuration section is the same as the Configuration section in a rule. Priority, ranking, and forward chaining features work the same at runtime whether a rule is created using the Rule Editor or the Rule Template Editor.

Declarations and Variables Section

The Declarations and Variables section (`declare` in the source view) is similar to the Declaration section of a rule. However, in rule templates you can declare primitive types (as you can in rule functions). You can also define initial expressions for primitive types, for example, `int j = 50;`

Pre-conditions Section

The Pre-conditions section (`when` in the source view) is similar to the Conditions section of a rule. However, WebStudio users can add additional conditions when defining individual business rules. The Pre-conditions section specifies conditions that must be met in *all* instances of a rule template before the rule's actions execute.

This feature enables complex calculations to be done before the definition of the condition. The Pre-conditions section could also contain conditions that might appear in any rule, such as the following:

```
con.prop1 > con2.prop1;
```

Action Context Section

The Action Context section (`actionContext` in the source view) has a similar purpose to the Actions section of a rule.

The Action Context section (`actionContext` in the source view) has a similar purpose to the Actions section of a rule. The Action Context section, however, defines *all possible actions* that can be taken by a business rule (after all conditions are met). Only the action context statements that the WebStudio user selects and defines as commands in the business rules are actually taken (depending on rule evaluation at runtime).

Defining the superset of all possible actions simplifies and restricts the configuration of business rules in WebStudio.

The actions are not completely defined in the rule template. Completing the definition of an action is a WebStudio user task. If bindings are used (and a view) then in WebStudio, the business rule writer has to enter only the binding values to complete the definition.

Action context statements are of three types: create, modify, and call, plus arbitrary actions, as explained next.

Create Statements

Create statements enable business rule developers to create new concepts or events. Create statement uses fully qualified name for the concepts and events instead of simple name. For example:

```
create Concepts.Bulk bulk;
```

In the rule template, that is as far as you go. It's up to the business rule developer to assign the specific constructor values for the entity type.

Modify Statements

Modify statements enable business rule developers to modify concepts or events. For example:

```
modify bulk;
```

Only entities in the scope of the rule can be used. As with create statements, it is up to the business rule developer to determine which properties to change and assign values to those properties.

Call Statements

Call statements enable business rule developers to call rule functions. Call statement uses fully qualified name for the concepts and events instead of simple name. Call statement uses a variable to store the return value. Call statement uses this return variable even if the return type is void. For example:

```
call RuleFunctions.matchFound matchfound;
```

The business rule developer assigns the parameters for the rule function.

Arbitrary Actions

Rule template developers can add arbitrary actions as well as create, modify, and call statements. For example:

```
create Concepts.Bulk bulk;
if (mybinding > 10) {
    MyRFs.myRF(otherbinding);
    mycon.prop1 = thirdbinding;
}
```

Bindings Tab

The form editor for rule templates provides an additional tab, the Bindings tab, with two sections that are not found in the Rule editor: Bindings and Views (bindings and views in the source view).

Bindings

A binding is used like a local variable whose value is assigned by WebStudio users when defining business rules.

When defining a rule template, you can use bindings in condition checks and in action statements. For example, suppose you define a binding called `dollarAmount`. You could then define a condition as follows:

```
output.UNIT_PRICE > dollarAmount;
```

At runtime, the `UNIT_PRICE` is checked against the specific dollar amount defined in the business rule.

Similarly, you could use the binding in an action statement as follows:

```
RuleFunctions.matchFound(match, null, dollarAmount);
```

When executed, the engine calls the `matchFound` rule function and passes the value of `dollarAmount` as defined in the business rule.

Bindings are primitive types. You can optionally provide an initial value, and the value can be defined using a domain model. You can specify a domain model by clicking in the Domain Model cell, and then clicking the **Browse** icon. If a domain model is specified for a binding, the view defined for a rule template displays the input field for the binding as a drop down box. In the following example, the String binding `day` is initially assigned to the value of `Monday`, and it is tied to the domain model `RuleFunctions.WeekDays`.

```
String day = "Monday" (RuleFunctions.WeekDays);
```

Views

In the Views section of the Bindings tab, you select a view that you have already defined. The selected view is used in WebStudio to present the rule template to the users and enable them to define the business rules (that is, executable rules).

If you do not associate a view with a rule template, a builder-based interface is used.

Tips for Working in the Rule Editor

Here are some tips for working with the Rule Editor

Switching between Form and Source Editors

You can freely switch between form and source editors for rules and rule functions. In each case the editors stay synchronized with the latest changes.

You cannot switch from the source editor to the form editor if there are any syntax or resolution errors in your code.

The Priority setting

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.

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 do this...	Do this...
To Use Content Assist to Complete Values	<p>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.</p> <p>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.</p>
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	<p>To jump to the location where an item is defined, you can use two methods.</p> <ul style="list-style-type: none"> • 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. <p>For example, you would jump from an alias to the declaration, and from an entity or entity property to the entity's editor.</p>

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 list. Otherwise, an input widget of the appropriate type is displayed, for example, a check-box for a boolean field.

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
	<p>Configuration Section</p> <p>Same as the Rule Editor Configuration section. See Rule Editor Reference for details.</p>
	<p>Declarations and Variables Section</p>
<p>Type</p>	<p>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.</p> <p>In addition you can specify variables using primitive types.</p> <p>For example (in the source view):</p> <pre>Concepts.Accounts.CheckAccount checkacc; int i = 123;</pre> <p>It is not necessary to add scorecards in the declaration in order to use them in the rule or rule template.</p>
<p>Alias</p>	<p>A name used to refer to the scope element or variable in the Pre-conditions (<i>when</i>) and Action Context (<i>actionContext</i>) sections of the rule. Aliases must be valid identifiers See Identifier Naming Requirements .</p>
<p>Expression</p>	<p>If you specify a primitive you can also specify an expression.</p>
	<p>Pre-conditions Section</p>
	<p>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 <code>true</code> or <code>false</code> such as are found in regular rules. Each line is joined to the others with an implicit and an operator.</p> <p>For the OR operator, use a double pipe (<code> </code>) on the same line.</p> <p>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 <i>TIBCO BusinessEvents Architect's Guide</i>.)</p>
	<p>Action Context Section</p>
	<p>Defines the list of all possible actions that can be executed when a rule instance based on this template is fired. Only the subset of these action context statements that are defined as commands in the business rules are actually considered.</p> <p>When you add an action context statement, first select one of the action types, then select an entity (create or modify action types) or rule function (call action type).</p>

Functions

TIBCO BusinessEvents provides various built-in function to perform the standard tasks; however you can also create your custom function to perform the task as per your requirement.

Catalog Functions

The functions registry includes various catalogs of functions provided with the product, and each catalog organizes its functions into categories. You can use functions in rule conditions and actions and in rule function bodies.

Some functions work together. For example, the Standard catalog function `String.append` requires that you first use the function `String.createBuffer`, to create a buffer to which strings can be appended.

To View the Catalog functions

All catalogs appear in the Catalog Functions view. To open the view navigate to **Window > Show View > Other > TIBCO BusinessEvents** and select Catalog Functions. The catalog view appears on the right, by default.

To View Documentation for Functions

Documentation for functions is provided in tooltips you can access while using the Catalog Functions view. Hover the mouse over the function name to see the tooltips.

You can also access this documentation using the online functions reference, available in the HTML version of the documentation, but not in PDF. To access the HTML version of documentation, visit <https://docs.tibco.com/products/tibco-businessevents> and open the **Web Help**.

Expand the contents panel to **Online References > Online References**, and on the right select the *TIBCO BusinessEvents Functions Reference* link.

Built-in Functions

For all the built-in functions, this section lists the main categories in each function catalog (but not sub-categories).

See *TIBCO BusinessEvents Functions Reference* for full details.

Standard Functions

The standard functions catalog contains the most used functions.

The standard function catalog includes the following categories:

Channel

These functions return information about destinations, and can resume and suspend a destination.

Cluster

Cluster functions help with multi-engine functionality.

Dashboard Function

Dashboard functions process an incoming request to generate an appropriate response. Starts streaming updates, Stops streaming updates.

DataGrid

These functions are for use with Cache object management. See [Cache Related Functions](#).

Collections

Collections functions allow you to deal with various collections structures. The functions names are:

- Comparator
- Iterator
- List
- Map
- Set
- SortedMap
- SortedSet

Date

Date functions allow you to compare two DateTime values using only the date portion of the value.

DateTime

DateTime functions allow you to perform the date/time related tasks and more: add units of time to DateTime, compare, retrieve, and format dates and times.

Engine

Engine functions allow you to retrieve information about the engine, for example, available memory or the number of rules fired. The functions names are:

- Locale
- Profiler
- Rtc
- Variable

Event

Event functions allow you to assert, create, and send simple events and perform other event-related tasks, for example, return the default destination URI of a simple event. The functions belonging to this group are

- Exit functions

Exception

Exception functions enable you to create exceptions.

File

File functions provide various useful functions used when working with files.

Hawk function

Functions to interact with Hawk micro-agents methods.

HTTP

HTTP functions are used with the HTTP channel. The Functions that belong to this group are named

- Servlet functions

Instance

Instance functions allow you to create and delete concept instances and perform other instance-related tasks, for example, return an instance given an internal ID. The functions belonging to this group are named:

- PropertyArray
- PropertyAtom
- Reflection
- StateMachine

Log

Log functions allow you to write statements in logs.

Math

Math functions allow you to perform advanced mathematical operations.

Metric Function

Creates a new Metric instance based on the data provided in the XSLT Mapper and adds it to the working memory. Adding the instance to the working memory will cause any rule conditions that depend on the concept to be evaluated.

Number

Number functions allow you to perform type conversions from and to numbers and return the maximum and minimum values for a numeric type.

SOAP

SOAP functions enable you to work with SOAP messages sent through an HTTP channel.

String

String functions allow you to perform comparisons, searches, conversions, and other operations with strings.

System

System functions allow you to send messages to a debug log, retrieve global variables, retrieve system properties, and write data to a file. The functions that belong to this group are named:

- ID functions
- IO functions, which allow the writing and closing of specific files.

Temporal

Temporal functions allow you to examine and perform calculations on values stored in a property's history. For information about using temporal functions. The functions that belong to this group are named:

- Calculus
- History
- Numeric
- Statistic

Util

Util functions category has one sub-category for working with HashMaps.

VRF

VRF functions (that is, Virtual Rule Function functions) allow you to work with decision tables. See TIBCO BusinessEvents Decision Manager User's Guide for details.

XPath

XPath functions allow you to evaluate XPath expressions.

ActiveSpaces Functions

These functions allow you to interact with ActiveSpaces metaspaces. They are used with the ActiveSpaces channels to perform operations on the ActiveSpaces metaspace and spaces connected by the channel.

Process Orchestration Functions

These functions allow you to interact with process jobs. See TIBCO BusinessEvents Process Orchestration documentation for details.

CEP Load Balancer Functions

Load balancer functions are used to configure and work with the load balancer. There are two types of load balancer functions:

- Receiver functions, which allow you to create or discard a receiver, and retrieve the receiver's membership details.
- Router functions, which allow you to create or discard a router, and send an event to a remote receiver.

CEP Pattern Functions

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

Hawk Functions

Functions to interact with Hawk micro-agents methods.

DBMS Functions

Database functions are provided for working with database concepts. See TIBCO BusinessEvents Data Modeling for more on database concepts.

Security Functions

These functions are used internally by the TIBCO BusinessEvents Decision Manager add on, for authentication.

Studio Functions

These functions allow you to use the Studio Util functions to build classes and EAR files.

Custom Functions

Using TIBCO BusinessEvents you can define custom functions and expose them as catalog functions or mapper functions. By exporting the project containing the custom Java functions to a project library, the custom Java functions can be used in other projects.

Adding Custom Java Functions

Using TIBCO BusinessEvents you can define custom functions and expose them as catalog functions. By exporting the project containing the custom Java catalog functions to a project library, the custom Java functions can be used in other projects.

Procedure

1. Define a simple Java class under the JavaSrc folder. For example, `MyFunction.java`.
2. To expose Java methods as functions, declare them as public static variables:

```
public static int sum(int a, int b)
{
    int x = a+b;
    return x;
}
```

3. The annotation `@BE Package` is a class level annotation: add the class-level annotation `@BEPackage` before any classes are declared.

```
@BEPackage(catalog = "arithmeticOperations", category = "arithmetic.operations",
synopsis = "Performs few arithemtic operations")
```

4. The annotation `@BEFunction` defines the synopsis, signature, parameters, return types, descriptions, examples and other relevant information about the function. The information provided in the annotation is displayed in the tooltip for the catalog function.

```
@BEFunction(name = "SummationTest",
description="Adds two given numbers",
signature="int sum(int a, int b)",
freturn =
@com.tibco.be.model.functions.FunctionParamDescriptor(name = "", type = "int",
desc = "Returns the summation of a and b."),
params = {
@com.tibco.be.model.functions.FunctionParamDescriptor(name = "a",
type = "int", desc = "First arg"),
@com.tibco.be.model.functions.FunctionParamDescriptor(name = "b",
type = "int", desc = "Arg two")
},
fndomain = {ACTION, CONDITION, QUERY, BUI},
version="1.0",see="",cautions="none", example=" ")
```

5. Save the file.
Once you add and save the annotations, the custom Java functions are available in the Catalog Functions view.

Adding Libraries

TIBCO BusinessEvents core libraries are automatically added to the Java Build Path.

If the project library contains Java source files, then those Java source files must be added to the Java build path source section. This ensures that the Java classes present in the project library are accessible from the Java source file of the current project.

Importing Old Projects

When older projects are imported to a latest environment, the project nature is updated to include the Java nature. The `.project` file contains the following entries:

```
<natures>
  <nature>com.tibco.cep.studio.core.studioNature</nature>
  <nature>org.eclipse.jdt.core.javanature</nature>
</natures>
```

Compiling Project and Building an EAR File

There is no change in the process to compile and build an EAR file.

Click **Project > Build Enterprise Archive** from the menu to build an EAR file.

Debugging

While debugging the projects, you can use Java breakpoints to step into the custom functions code. Java debug breakpoints and rulefunction breakpoints work seamlessly, allowing you to step into the code after hitting a breakpoint and inspect the values of the variables as they change.

Annotations Reference

Reference for @BEPackage

Field	Description
enabled	This annotation allows the user to specify if an associated annotation is enabled or disabled for use.
catalog	Name of the catalog where the custom function will reside.
category	Category for the custom functions defined in the class.
synopsis	Brief description for the group of custom functions defined in the specified directory.

Reference for @BEFunction

Field	Description
enabled	This annotation allows the user to specify if an associated annotation is enabled or disabled for use.
deprecated	This annotation element identifies if its associated method is deprecated or not.
name	Name of the custom function.
synopsis	Brief description of the custom function.
signature	Signature for the custom function.
params	List of parameters for the custom function.
freturn	Return type of the custom function.

Field	Description
version	The TIBCO BusinessEvents version from which a given catalog function became public API.
see	Refers to the other custom functions or java doc URLs.
mapper	Used for functions that involve invoking a mapper UI on Ctrl-Click .
description	Detailed description of the custom function.
async	A boolean value indicating if the use of this function will cause the rules to be evaluated again.
reevaluate	A boolean value value indicating if the use of this function will cause the rules to be evaluated again.
cautions	List any cautions that may be applicable when using the custom function.
domain	The specified domain, from which Java methods are accesible. Examples of domain are ACTION, CONDITION, QUERY, BUI.
fndomain	Describes an enumerated function execution domain.
example	Example of the custom function.

Considerations when Defining the Functions

Java methods defining the functions can have primitive data types, or Concept, Event and Object. They can return any primitive data types or Concept, Event and Object.

If the custom functions have primitive data types only, it will also be visible and uable in the mapper. In that case, the category attributes of the @BEPackage annotation is used as the folder for your custom functions.



The interface has some limitations as it does not allow static methods. Do not use annotations when using interfaces. Create an interface as a normal java interface without any annotations and implement it in a class without any annotations. To see these interface methods under the custom catalog function, create a static overloading method with annotations.

For example:

```
public interface InterfaceProjectLibTest
{
    public abstract String[] getAllDestinations();
    public abstract String getClusterName();
}

public class interfImpl implements InterfaceProjectLibTest
{
    static int count=0;
    @Override
    public String[] getAllDestinations()
    {
        Object[] ars= null;
        RuleSession rs = RuleSessionManager.getCurrentRuleSession();
        String[]
        activdest=(String[])rs.invokeCatalog("Channel.getAllDestinations", ars);
        count = activdest.length;
        return activdest;
    }
}
```

```

    }

    @Override
    public String getClusterName() {
        // TODO Auto-generated method stub
        Object[] ars= null;
        RuleSession rs = RuleSessionManager.getCurrentRuleSession();
        String clustername;
        clustername=(rs.invokeCatalog("Cluster.getClusterName",
ars)).toString();
        return clustername;
    }

    @BEFunction(name = "getNoOfDests", signature= "getNoOfDests",
description="Returns Number of destinations of the current rulesession")
    //@BEFunction
    public static int getNoOfDests()
    {
        interfImpl im = new interfImpl();
        im.getAllDestinations();
        + System.out.println("count: " + count);
        return count;
    }

    @BEFunction(name = "ReturnAllDestinations", signature= "getAllDest",
description="Returns all destinations of the current rulesession")
    public static String[] getAllDest()
    {
        interfImpl im = new interfImpl();
        String[] activedests = im.getAllDestinations();

        return activedests;
    }

    @BEFunction(name = "ReturnClusterName", signature= "getClusName()",
description="Returns the clustername")
    public static String getClusName()
    {
        interfImpl im = new interfImpl();
        String clustername=im.getClusterName();
        System.out.println("ClusterName is: " + clustername);
        return clustername;
    }
}

```

Ontology Functions

Ontology functions are generated by TIBCO BusinessEvents based on the concepts, events, and rules in your project.

There are three types of ontology functions:

Constructors

They allow you to create a simple event or concept instance.

Time events

They allow you to create and schedule a time event. See [Working With Time Events](#).

Rule functions

They allow you to invoke a rule function.

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

Various ontology functions appear depending on the project and the add-on products installed:

- The ontology functions catalog lists all the entity types in a project. Each type has a function for creating an instance of that type.
- Each rule function also has an ontology function enabling it to be invoked in rules.
- 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 customers who wish to continue using them. To make them visible in the Catalog Functions view, perform the following actions.

Procedure

1. Open the following file for editing:

```
BE_HOME/studio/eclipse/configuration/studio.tra
```

2. Add the appropriate property, using the specified name. For example, the following formats are typically used:

```
TIBCO.BE.function.catalog.function-catalog-name=true
TIBCO.CEP.modules.function.catalog.function-catalog-name=true
```

3. Save the file and restart TIBCO BusinessEvents Studio.

Function Tooltips and Decorations

When you float the cursor over a function in the registry, TIBCO BusinessEvents displays the description and syntax in a tool tip next to the cursor. Functions that can be used with various product features are decorated with small symbols that indicate useful information about use of the function.

Tool Tips

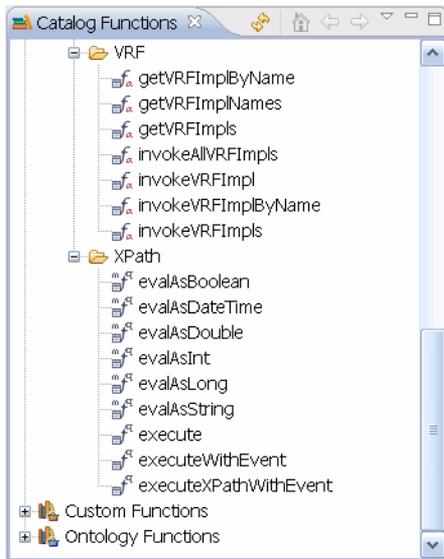
Tooltips also form the online reference to the function catalogs. See *TIBCO BusinessEvents Functions Reference*.



You can create your own tool tips for custom functions. You can also 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). the second ones have decorations that cluster around the "f" next to the function name in the function catalog. A function can have zero, one, or more decorations. The following figure shows all the available decorations. They are described in the sections after the figure:



Action-Only Functions

Some of these functions have side effects, for example they can change values. Other functions are limited to actions for other reasons. These action-only functions are identified by a small *a* at the bottom right of the *f*. For example: *f_a* `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: *m_f* `evalAsString`.

Functions That Can Be Used in TIBCO BusinessEvents Decision Manager

Functions that can be used in TIBCO BusinessEvents Decision Manager are marked with a small table icon, *tf_a* `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, *q_f* `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

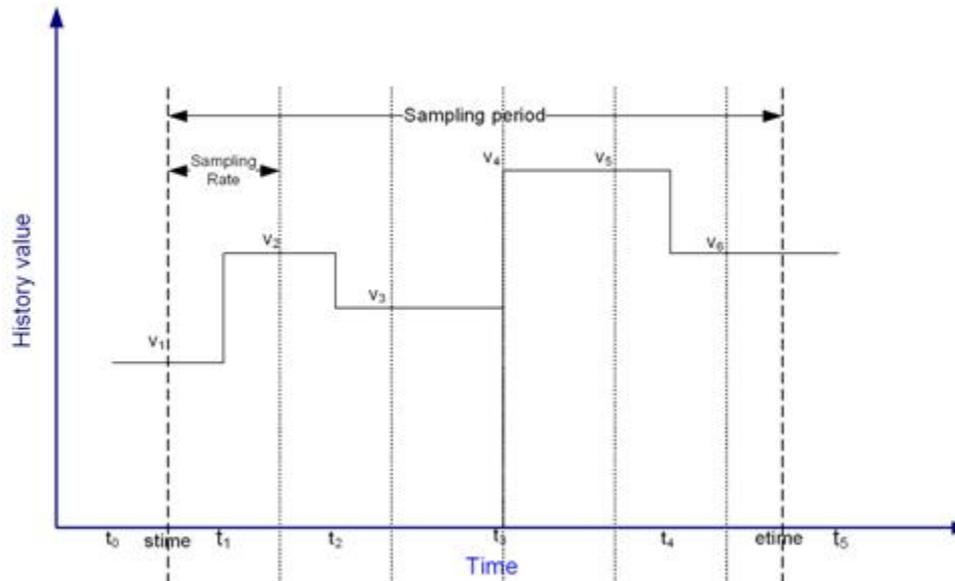
The set of Standard functions that come with TIBCO BusinessEvents includes functions that allow you to perform calculations on numeric values sampled over time.

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.

Temporal Functions Parameters



All temporal functions include these parameters, illustrated in [Temporal Functions Parameters](#):

property

The property for which you want to sample values.

stime

The time from which you want to begin sampling values (the start time) measured in milliseconds since 00:00:00 UTC on January 1, 1970.

etime

The time at which you want to stop sampling values (the end time) measured in milliseconds since 00:00:00 UTC on January 1, 1970.

sample_rate

The number of milliseconds between samples.

bound_by_stime

A flag indicating whether the start-time is flexible:

- `True` indicates that if the start time you provide is earlier than the timestamp for the oldest available value, you want to perform the calculation starting with the oldest available value.
- `False` indicates that if the start time you provide is earlier than the timestamp for the oldest available value, you want to abort the calculation.

Virtual Rule Functions and VRF Catalog Functions

The VRF category of functions (within the standard functions) are used only with TIBCO BusinessEvents Decision Manager.

The VRF category of functions provide flexibility when you are working with virtual rule function implementations. Virtual rule functions are implemented by decision tables.



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()
```



Defining the execution order of multiple decision tables for one VRF: When a VRF has multiple implementations (decision tables), the order in which the decision tables execute can be defined using each decision table's Priority setting, which is set in the decision table Properties tab.

Adding a Virtual Rule Function

To add a virtual function, perform the following steps.

Procedure

1. Right-click the folder where you want to store the virtual rule function and select **New > Rule Function**. You see the New Rule Function Wizard.
 - a) In the **Rule Function Name** field, type a name for the rule function.
 - b) In the **Description** field, type a description. (In the source editor the description appears in the * @description line of the comments at the top of the editor).
 - c) Select the **Virtual** check box.



You cannot change a new resource name after you click Finish. (You can change the description, however.)

2. Click **Finish**.

Click the tabs at the bottom of the editor to switch between the form-based editor and the source editor as you work, depending on your preference. These instructions use the form editor and mention the equivalent settings in the source editor.

3. In the Form editor Configuration section, add or edit a description as desired. (In the source editor the description appears in the * @description line of the comments at the top of the editor).
4. If you did not do so in the Wizard, select the Virtual 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 perform the following actions:
 - 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.

Result

For example, here is the source view for a simple virtual rule function:

```
/**
 * @description Action to take when account is suspended
 */
virtual void rulefunction Rules.FollowUp {
  attribute {
    validity = ACTION;
  }
  scope {
    Concepts.Account account;
  }
  body {
  }
}
```

VRF Function Arguments

The VRF functions use various subsets of the following common arguments:

Common Arguments for VRF Functions

Name	Type	Notes
<code>vrfURI</code>	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 <code>CreditCardApplication</code> example, the virtual rule function <code>Person_VirtualRuleFunction()</code> has the following URI: <code>/Virtual_RF/Person_VirtualRuleFunction</code>
<code>vrfImpl</code>	Object	An object representing a virtual rule function implementation. This argument is required when invoking specific virtual rule function implementations.
<code>implName</code>	String	The name of a decision table (also known as a virtual rule function implementation). For example, in the <code>CreditCardApplication</code> example, the virtual rule function <code>BankUser_VirtualRuleFunction</code> has an implementation (decision table) called <code>bankUser</code> . The <code>implName</code> argument is used to retrieve a corresponding implementation object, or to execute an implementation.

Name	Type	Notes
<code>args</code>	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 <code>processApplication</code> implementation in the <code>CreditCardApplication</code> example project requires concepts of type <code>Application</code> , <code>BankUser</code> , and <code>CreditCardApplication</code> to be passed as arguments. In order to invoke the <code>processApplication</code> implementation, an instance of each concept type must be passed in the <code>args</code> array.
<code>returnValues</code>	Object array	<i>Not used in this release.</i> This argument is used only for the <code>invokeVRFImpls</code> function. When invoking multiple implementations, the return value of each implementation is stored in this array. The array will contain a null entry for each implementation that does not return a value.

Cache Related Functions

Various standard functions in the Standard catalog `Cluster.DataGrid` category enable you to work with objects in the cache.

Cache load functions load items into the Rete network so they are available.

Tool tips in the user interface (and reproduced in the *TIBCO BusinessEvents Functions Reference*) explain the details of how to use the functions. This section explains certain patterns of use.

These core functions are used with both the Oracle and TIBCO cache providers.

The previous versions of these core functions (which began with `C_`) are deprecated. Migrate projects created in earlier releases to use the current names.

Query functions are not available for the 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.



When querying by extid, you must use the functions `Cluster.DataGrid.CacheLoadConceptByExtIdByUri` and `Cluster.DataGrid.CacheLoadEventByExtIdByUri`.

Locking Functions

```
Lock()
UnLock()
```

In the event preprocessor, use the `Lock()` function to prevent other threads or engines from operating on the same entity. The lock is automatically released at the end of the RTC.

Use `UnLock()` only in a preprocessor and only to handle cases where you need to release the lock immediately instead of at the end of the RTC, for example because some information is missing that would be required to go forward.

See *Using Locks to Ensure Data Integrity Within and Across Agents in TIBCO BusinessEvents Architect'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.

Specialty Functions Not for General Use

The following functions are used only in certain applications:

```
CacheLoadConceptIndexedByExtId()
CacheLoadEntity()
CacheReevaluate()
EnableCacheUpdate()
```

Indexing for More Efficient Cache Queries

When you use 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.

Create Indexes Using a Coherence Function

This method applies to the Coherence cache provider only. It is not the preferred method.

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_DatatypeAtomGetter` 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);
```

Creating 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.

Procedure

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, select the **Present in Index** check box for the property you want to index.
4. Save the CDD file.

Enabling Explicit Tuple Format for DataGrid Fields

You can set a property in the CDD so that they are instead stored as tuples. That is, TIBCO BusinessEvents attempts to create explicit space structures for each entity type.

In Shared Nothing mode, explicit tuple format is assumed internally. You should not set this property in Shared Nothing mode.

You can still use query language queries (as explained in [Query the Cache Using BQL Queries](#)) if entities and their properties are stored as blobs. However, performance is improved if they are stored as tuples.

Only certain datatypes can be stored as tuples:

- Primitive datatypes

`int, long, boolean, string, datetime, contained, reference`

Are stored as the equivalent TIBCO BusinessEvents DataGrid types:

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

- Concepts
 - Concept property arrays are stored as blob columns
- Events
 - Payload is a byte array or blob
- State Models

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

Prerequisites

Procedure

- To enable explicit tuple format, at the cluster level, set the following field to true: "**Store Properties As Individual Fields**".

Query the Cache Using BQL Queries

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



- 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 *TIBCO BusinessEvents Architect's Guide*.

Procedure

1. Add a rule function in the inference agent that contains the following function:

```
Query.Util.startDynamicQuerySession().
```

Configure this as a startup rule function (in the CDD file). For more on dynamic query sessions see *TIBCO BusinessEvents Event Stream Processing Query Developer's Guide*.

2. Add another rule function. Configure it as an event preprocessor (in the CDD file).
3. In the event preprocessor function, create a query string that selects entities from the cache. You can use any valid BQL query.

4. Use the query string in the following function:

```
Query.Util.executeInDynamicQuerySession(queryString, null, true)
```

The function returns a list of entities from the cache.

5. Use the list returned as needed. For example, iterate over the list and load the entities, as shown in the following example.



You can also do cache lookups using @id and @extId attributes.

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

Oracle Coherence Cache Query functions work only with a Coherence cluster.

For information on using Oracle Coherence with TIBCO BusinessEvents, see *TIBCO BusinessEvents Configuration Guide*.

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.

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.

Structure of a Function Catalog

A function catalog is an XML file that conforms to the schema file `function_catalog.xsd`.

This allows TIBCO BusinessEvents to integrate your custom functions with the function registry in TIBCO BusinessEvents Studio. The function catalog must be in the XML format shown below and described in [Elements in the Function Catalog](#) to map properly to the schema.



- 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

The Function Catalog consists of various elements and sub-elements described in this table.

Function Catalog Elements

Element Name	Sub-Elements	Description
<catalog name="name">		The root element. Attribute: name="name" where <i>name</i> 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.

Element Name	Sub-Elements	Description
	<name>	The name of the function.
	<class>	The java class that implements the function.
	<desc>	Optional. A description of the function.
	<async>	Set to <code>true</code> if the function executes asynchronously. Set to <code>false</code> if the function executes synchronously.
	<helpurl>	Not used in this release.
	<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 <code>true</code> to alert TIBCO BusinessEvents that this function has side effects and is not valid in conditions. Otherwise, set to <code>false</code> . Valid values: <code>true</code> , <code>false</code> .
	<isValidInBui>	If this function can be used in decision tables, set this element to <code>true</code> . Otherwise, set to <code>false</code> . Valid values: <code>true</code> , <code>false</code> .
	<isValidInQuery>	If this function can be used in queries, set this element to <code>true</code> . Otherwise, set to <code>false</code> . Valid values: <code>true</code> , <code>false</code> .
	<reevaluate>	Relevant only when a function is used in a condition. Valid values: <code>true</code> , <code>false</code> .

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`:
 - 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`:
 - 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.

Rule Language Grammar

There are some basic concepts that you need to know about working with the rule language grammar.

Whitespace

Whitespace is used to separate tokens (identifiers, keywords, literals, separators, and operators) just as it is used in any written language to separate words. Whitespace is also used to format code.

These are whitespace characters, excluding line terminators:

- the ASCII SP character, also known as *space*
- the ASCII HT character, also known as *horizontal tab*
- the ASCII FF character, also known as *form feed*

Line terminators include these characters:

- the ASCII LF character, also known as *newline*
- the ASCII CR character, also known as *return*
- the ASCII CR character followed by the ASCII LF character

Comments

Comments are used to comment a rule.

Comment rules are as follows:

```
/* text */    The text from /* to */ is ignored
// text      The text from // to the end of the line is ignored
```

Separators

Separators are used to separate statements, expressions, or arguments in a function.

The following tokens are used for separators:

Tokens	Description
;	Statement separator for conditions and actions.
(Expression Grouping begin, or function argument list begin.
)	Expression Grouping end or function argument list end.
,	Argument list separator.

Identifier Naming Requirements

An identifier (or *name*, to use the user interface label) is an unlimited-length sequence of letters and digits, the first of which must be a letter. Letters include uppercase and lowercase ASCII Latin letters A-Z, a-z, and the underscore (_).

Here are some guidelines for naming of identifiers:

- Do not use the dollar sign (\$).
- Identifiers are case sensitive.

- Identifiers cannot have spaces (except shared resource identifiers).
- Identifiers may not be the same as any literal, keyword, or other reserved word. See [Keywords and Other Reserved Words](#) and [Literals](#).

Letters and digits may be drawn from the entire Unicode character set, which supports most writing scripts in use in the world today, including the large sets for Chinese, Japanese, and Korean. This allows programmers to use identifiers in their programs that are written in their native languages.

Digits include the ASCII digits 0-9, while two identifiers are the same only if they have the same Unicode character for each letter or digit. Note that some letters look the same even though they are different Unicode characters. For example, a representation of the letter A using `\u0041` is not the same as a representation of the letter A using `\u0391`.

Two example identifiers: `new_order E72526 creditCheck`



Here is a more succinct way for programmers to understand the requirements:

```
<Identifier> := [ <ID_START> { <ID_PART> }* ]
<ID_START> := except '$', any character for which
java.lang.Character.isJavaIdentifierStart() returns true
<ID_PART> := except '$', any character for which
java.lang.Character.isJavaIdentifierPart() returns true
```

Local Variables

Local variables of certain types are used in rule actions and rule functions.

The following types can be used:

- Primitives
- Concepts
- Events

Primitive Arrays

Primitive arrays with a fixed length can also be used when working with a rule language grammar .

Here are examples of array declaration, initialization, and array creation expressions:

- Array declaration and initialization:

```
int i; // int
int[] ii = {1,2,i}; // array of int
```

- Array creation with initialization expression:

```
ii = int[] {1,2,3};
```

- Array creation without initialization expression:

```
int[] arr = int[5] {};
arr = int[5]{};
```

- Getting the length of the array:

```
int arrLength = arr@length;
```

Literals

Certain literals are used for the rule language grammar.

The TIBCO BusinessEvents rule language supports these literals:

int

One or more digits without a decimal. May be positive or negative.

Examples:

```
4 45 -321 787878
```

long

An integer literal suffixed with the letter L. The suffixed L can be either upper or lower case, but keep in mind that the lower case L (l) can be difficult to distinguish from the number one (1).

Examples:

```
01 0777L 0x10000000L 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

Escape sequences are used to represent characters when working with the rule language grammar.

You can represent characters in literals using these escape sequences:

Escape Sequences

Character	Escape Sequence
backspace	\b
horizontal tab	\t
linefeed	\n

Character	Escape Sequence
form feed	\f
carriage return	\r
double quote	\"
single quote	\'
backslash	\\

Operators

Certain operators are defined for use with the rule language grammar.

Operators in this list are also used in the Java language and work the same as the Java operators:

Operators in the TIBCO BusinessEvents Rule Language

Operator	Notes
++ --	increment, decrement
+ -	unary plus, unary minus
* \ %	multiplication, division, remainder
+ -	addition, subtraction
> < >= <=	greater than, less than, greater than or equal to, less than or equal to
=	assignment
== !=	equality, inequality (Does deep string comparison, unlike Java.)
&& !	boolean AND, OR, NOT
-- += *= /= %=	combined operation and assignment. As an example, <code>expr += 2;</code> is the same as <code>expr = expr + 2;</code> += works on strings as well as numbers. For example, you have a variable <code>String s = "abc";</code> and then you use <code>s+= "def";</code> 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: <code>boolean b = customer instanceof USCustomer;</code>
.	property access

Operator	Notes
@	attribute access

Keywords and Other Reserved Words

Certain keywords and reserved words cannot be used in the rule language grammar.

Do not use the words listed in this section as identifiers, resource names, or folder names. Case sensitivity depends on context (as noted in documentation). The list also includes keywords and other reserved words for all add-on products.

\$lastmod
abs
abstract
accept
ACTION
AdvisoryEvent
after
alias
all
and
any
as
asc
assert
attribute
avg
backwardChain
between
body
boolean
break
by
byte
case
catch
char
class
Concept
CONDITION
const
ContainedConcept
continue
count

date
DateTime
day
days
dead
declare
default
define
delete
desc
distinct
do
double
during
emit
else
entity
enum
Event
except
exists
extends
extId
false
final
finally
first
float
for
forwardChain
from
goto
group
having
hour
hours
id
if
implements

import
in
instanceof
int
interface
intersect
last
is_defined
is_undefined
key
last
latest
like
limit
lock
long
maintain
max
millisecond
milliseconds
min
minute
minutes
mod
moveto
native
new
not
null
object
offset
one
or
order
order
package
pattern
pending_count
policy

priority
private
process
protected
public
purge
QUERY
rank
repeat
requeue
return
rule
rulefunction
scope
second
seconds
select
short
SimpleEvent
sliding
starts
static
strictfp
String
such
sum
super
switch
synchronized
that
then
this
throw
throws
time
times
TimeEvent
timestamp
to

transient
 true
 try
 tumbling
 undefined
 union
 unique
 using
 validity
 virtual
 void
 volatile
 when
 where
 while
 with
 within

Reserved Global Variables

Certain global variables are reserved in TIBCO BusinessEvents. Do not use these keywords for creating global variables.

Reserved Global Variable	Default Value
Domain	domain
Deployment	<Name of project>
MessageEncoding	ISO8859-1

Keywords Reserved By Oracle

Certain keywords are reserved by Oracle and not necessarily by TIBCO BusinessEvents. Do not use these words when creating backing store scripts, else an error might occur.

ACCESS
 ADD
 ALL
 ALTER
 AND
 ANY
 ARRAYLEN
 AS
 ASC

AUDIT
BETWEEN
BY
CHAR
CHECK
CLUSTER
COLUMN
COMMENT
COMPRESS
CONNECT
CREATE
CURRENT
DATE
DECIMAL
DEFAULT
DELETE
DESC
DISTINCT
DROP
ELSE
EXCLUSIVE
EXISTS
FILE
FLOAT
FOR
FROM
GRANT
GROUP
HAVING
IDENTIFIED
IMMEDIATE
IN
INCREMENT
INDEX
INITIAL
INSERT
INTEGER
INTERSECT
INTO

IS
LEVEL
LIKE LOCK
LONG
MAXEXTENTS
MINUS
MODE
MODIFY
NOAUDIT
NOCOMPRESS
NOT
NOTFOUND
NOWAIT
NULL NUMBER
OF
OFFLINE
ON
ONLINE
OPTION
OR
ORDER
PCTFREE
PRIOR
PRIVILEGES
PUBLIC
RAW
RENAME
RESOURCE
REVOKE ROW
ROWID
ROWLABEL
ROWNUM
ROWS
START
SELECT
SESSION
SET
SHARE
SIZE

SMALLINT
SQLBUF
SUCCESSFUL
SYNONYM
SYSDATE
TABLE
THEN
TO
TRIGGER
UID
UNION UNIQUE
UPDATE
USER
VALIDATE
VALUES
VARCHAR
VARCHAR2
VIEW
WHenever
WHERE
WITH

Attributes

TIBCO BusinessEvents provides attributes that you can use in rules to access information of various kinds. Use the @ operator to access attributes.

Attributes

Entity	Attributes	Type	Returns
SimpleEvent	@id	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.)
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.

Entity	Attributes	Type	Returns
	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.

Exception Handling

The TIBCO BusinessEvents Rule Language includes `try`, `catch`, and `finally` blocks and has an error type attribute `@errorType`.

The `try`, `catch`, and `finally` blocks behave like their same-name Java counterparts.



You can also use the special `AdvisoryEvent` event type to be notified of exceptions that originate in user code but that are not caught with a `catch` block. To use the `AdvisoryEvent`, click the plus sign used to add a resource to the declaration. `AdvisoryEvent` is always available in the list of resource.

Syntax

Exception handling syntax defines which combinations can be used.

These combinations are allowed:

- `try/catch`
- `try/finally`
- `try/catch/finally`

try

```
try {try_statements}
```

catch

```
catch (Exception identifier ) {catch_statements }
```

finally

```
finally {finally_statements}
```

Examples

Exception handling syntax for the rule language grammar is presented through some examples.

try/finally Example

```
String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} finally {
    //If readStatus() throws an exception,
    //MyScorecard.status will be set to "default status"
    //but the exception won't be caught here.
    //Otherwise MyScorecard.status will be set to the
    //return value of readStatus()
    MyScorecard.status = localStatus;
}
```

try/catch/finally Example

```
String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} catch(Exception exp) {
    System.debugOut("readStatus() threw an exception with message"
        + exp@message);
} finally {
    //If readStatus throws an exception,
    //MyScorecard.status will be set to "default status"
    //Otherwise MyScorecard.status will be set to the
    //return value of readStatus()
}
```

```

    MyScorecard.status = localStatus;
}

```

try/catch Example

```

String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} catch(Exception exp) {
    System.debugOut("readStatus() threw an exception with message "
        + exp@message);
}

//If readStatus throws an exception,
//MyScorecard.status will be set to "default status"
//Otherwise MyScorecard.status will be set to the
//return value of readStatus()
MyScorecard.status = localStatus;

```

try/catch Example with Checking errorType and re-throw

```

String localStatus = "default status";
try {
    //readStatus might throw an exception
    localStatus = readStatus();
} catch(Exception exp) {
    if (exp@errorType == "java.lang.NullPointerException")
        throw exp;
    System.debugOut("readStatus() threw an exception with message " + exp@message);
}

//If readStatus throws an exception other than NullPpointerException,
//MyScorecard.status will be set to "default status"
//If readStatus throws NPE, MyScorecard.status will be set to "default status" and
NPE will be re-thrown.
//Otherwise MyScorecard.status will be set to the return value of readStatus()
MyScorecard.status = localStatus;

```

Flow Control

The TIBCO BusinessEvents rule Language includes commands to perform conditional branching and iteration loops.

if else

The `if/else` command allows you to perform different tasks based on conditions.

Syntax:

```
if(condition){  
    code_block;  
}  
else{  
    code_block;  
}
```

for

The `for` command allows you to create a loop, executing a code block until the condition you specify is false.

Syntax:

```
for(initialization; continue condition; incrementor){  
    code_block;  
    [break;]  
    [continue;]  
}
```

Where:

`break` allows you to break out of the loop.

`continue` allows you to stop 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;]  
}
```

Where:

`break` allows you to break out of the loop.

`continue` allows you to stop executing the code block but continue the loop.

Rule Language Datatypes

Rule language datatypes are defined with conversion tables, with information about operators and types, and with information about how TIBCO BusinessEvents handles inconsistency problems with datatypes.

Concept Properties to XML Datatype Conversions

Concept Properties to XML Datatype Conversions

Property Type	Int	Long	Float	Double	Boolean	String	DateTime	ComplexType	@ref
int	L	L	L	L		L			
long	N	L	N	N		L			
double	N	N	N	L		L			
String	L	L	L	L	L	L	L		
boolean					L	L			
Datetime						L	L		
ContainedConcept								D	
ConceptReference									ID

N - Numeric conversion, loss of information 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

Rule language grammar defines operators compatibility with types.

Compatibility is described as follows:

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, /, c, m, p	=, math, eq, cmp, /, c, m, p			=, math, eq, cmp	
	lon	+	=, math, eq, cmp	=, math, eq, cmp, /, c, m, p	=, math, eq, cmp, /, c, m, p			=, math, eq, cmp	
	dou	+	=, math, eq, cmp	=, math, eq, cmp, /, c, m, p	=, math, eq, cmp, /, c, m, p			=, math, eq, cmp	
	boo	+				=, eq		=, eq	
	ent	+					=, eq, inst	=, eq, inst	

Right Side of Operator									
	str	int	lon	double	boo	ent	obj	dat	
obj	=, +, eq, cmp, inst	=, math, eq, cmp	=, m at h, eq , c m p	=, ma th, eq, cm p	=, eq	=, eq, inst	=, eq, inst	=, eq, inst	
dat	+						=, eq, inst	=, eq, cmp, inst	

Abbreviation	Meaning and Notes
boo	Boolean.
cmp	Comparison operators: <, >, <=, >=
dat	Date/Time
dou	Double
ent	Entity. Type includes Concept, Event and Scorecard. Both operands must either be of the same type or have a subtype-supertype relationship
eq	Equality operators: ==, !=
inst	instanceof
int	Integer
lon	Long
math	Numerical operators: unary +, unary -, =, -, *, /, %
obj	Object
str	String

Correcting Inconsistencies of Type

TIBCO BusinessEvents attempts to correct inconsistencies of type whenever possible by converting expressions to the appropriate type.

It converts expression types in the following cases:

- An expression uses the plus sign (+) with a string operand.

- An arithmetic expression includes numbers of differing types.
- The value of an expression is assigned to a variable of a different type.
- The value of an expression is passed to a function that declares a different type.

There are some inconsistencies of type that TIBCO BusinessEvents cannot correct. For example, all expressions within conditions must be of type boolean. If an expression within a condition evaluates to anything other than boolean, it would be illogical for TIBCO BusinessEvents to convert the expression to boolean. In cases like this, TIBCO BusinessEvents returns an error at compile time.

String Operands

When an expression uses the plus sign (+) with a string operand, TIBCO BusinessEvents treats the expression as a request for concatenation rather than addition. It converts the second operand to a string and concatenates the two strings.

For example:

```
"area code: " + 650
```

becomes

```
"area code: 650"
```

Arithmetic Expressions

The following information applies to these operators:

```
* / % + - < <= > = == !=
```

When an expression uses one of the above arithmetic operators with two numbers of different numeric types, TIBCO BusinessEvents promotes one of the two operands to the numeric type of the other. It makes these promotions as follows:

- If either operand is a double, TIBCO BusinessEvents promotes the other to a double.
- Otherwise, if either operand is a long, it promotes the other to a long.

Assignment Conversion

If the value of an expression is assigned to a variable, TIBCO BusinessEvents converts the expression's type to that of the variable. This might include, for example, converting a double to an int, or converting a generic model type to a more specific model type.

Function Argument Conversion

Conversions of function arguments are handled in the same way as assignment conversions.

Mapping and Transforming Data

Variables in the scope of a rule or rule function can be mapped to arguments of a function used in that rule or rule function.

See also [XPath Formula Builder](#) for related information, as well as [Working with Rules and Functions](#).

The Function Argument Mapper allows you to supply the data that a function expects as input. For instructions on accessing the Function Argument Mapping wizard in the rule editor see [Using the Function Argument Mapping Wizard](#).

XPath 1.0 and XPath 2.0

TIBCO BusinessEvents supports XPath 1.0 and XPath 2.0. The default is XPath 2.0 for the new projects. TIBCO BusinessEvents uses the Standard Widget Toolkit (SWT) mapper to support XPath 2.0 and XSLT 2.0.

The following BE functions SWT mapper, and the underlying implementation of these functions uses the related XML runtime:

- `Event.createEvent`
- `Instance.createInstance`
- All `XPath.evalAs` functions, for instance `XPath.evalAsString`

Using the SWT mapper for XPath 2.0, you can perform the mapping in the same way you can do for XPath 1.0; however, the mapper uses the XPath 2.0 expressions. In contrast to XPath 1.0, the type casting from one data type to another data type is not done implicitly in XPath 2.0. In XPath 2.0, you must perform the explicit typecasting using the constructor function. BusinessEvents also provides the option in the Function Argument Mapping wizard to autofix the typecasting error. You can fix the common issues using the Mapper Function Migration wizard, see [Migrating Mapper Functions From XPath 1.0 to XPath 2.0](#) for more details on how to common mapping issues. You can also use the `studio-tools` utility to fix those common issues, see [Migrating Mapper Functions to XPath 2.0 Using Command Line](#) for more details.

In XPath 1.0, the mapper has two different windows the Function Argument Mapping and XPath Function Builder; however, in XPath 2.0 both window are now combined in the same Function Argument Mapping wizard. Thus, if your project uses XPath 1.0 then you will see the Function Argument Mapping wizard and XPath Formula Builder in different windows. However, if your project uses XPath 2.0 then you will see the Edit tab inside the Function Argument Mapping wizard (see [Function Argument Mapping Wizard for XPath 2.0](#)).

The mapper for XPath 1.0 had some TIBCO provided functions that were not part of the XPath 1.0 specifications, for instance `format-dateTime`. These functions are currently unavailable with the SWT mapper for XPath 2.0, and the old XPath expression may give error, in such case you have to manually fix those error.

All the following sections provides information on mapping data with respect to XPath 1.0 user interface; however, you can perform all those functions in XPath 2.0. For reference on the user interface for XPath 2.0, see [Function Argument Mapping Wizard for XPath 2.0](#).

Function Argument Mapping Wizard for XPath 1.0

Using the Function Argument Mapping wizard for XPath 1.0 you can map the source data to the input arguments of the mapper function. The wizard consists of two sections: Function and Input.

Function Section

The function section, in the upper part of the dialog, shows the view-only name of the function you are working with and the editable entity path to the item whose properties and attributes you want to map to the function arguments.

Input Section

In the input section, there are two panels:

- Scope Variables Panel

The scope variables panel shows the list of properties and attributes available to the function, as well as global variables defined in the project.

- Function Panel

The function panel uses an Extensible Stylesheet Language Transformation (XSLT) template that specifies how scope variables should be transformed to provide the expected input. Normally, you do not need detailed knowledge of XSLT to specify a function's expected output. However, if you are familiar with XSLT and you wish to see the actual code, you can right-click on any item in the Function panel and choose Copy from the popup menu. Then open a blank text document and choose Paste. The XSLT is displayed in your text document.

You can also use your own XSLT templates to perform transformations instead of using the techniques described in the following sections. You can paste XSLT into your function input fields, or into the XPath Formula field in XPath Formula Builder. You cannot, however, paste XSLT directly into the function argument in the rule editor.

The Input tab consists of several toolbar buttons, popup menus, and icons to help in performing mapping of data for XPath 1.0.

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 38](#) describes the buttons and right-click menu items available in the panels of the **Input** tab.

Input tab toolbar buttons

Button	Right-Click Menu	Description
Scope Variables Panel		
		Coercions. 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.

Button	Right-Click Menu	Description
	Copy	Copies the selected element. The element can be later pasted.
Function Panel		
		Shows or hides the mapping formulas for the input elements.
		Move Up. Moves the selected element up in the Function tree.
		Move Down. Moves the selected element down in the Function tree.
	Move Out	Move Out. Promotes the selected element to the next highest level in the Function tree.
	Move In	Moves the currently selected element into a new statement. This displays the Move Into New Statement dialog that allows you to choose the statement you wish to move the element into. See XSLT Statements for more information about XPath statements.
	Delete	Deletes the mapping for the selected element. If no mapping is defined, the element itself is deleted (along with all child elements). Note Elements are predefined. Do not delete elements. Deletion of an element causes mapper validation errors due to the mismatch of the right panel's content with its schema.
		Insert. Click Insert to pop-up a New XSLT Statement dialog where you can define an XSLT statement. The statement is inserted in the function input schema on the same level of the hierarchy as the currently selected element. You can add one XSLT statement at a time with this button. The right-click menu item Statement provides a shortcut for multi-line statements, such as Choice or If. See the description of the Statement menu item below for more information. Note Elements are predefined. Do not add new elements. Doing so causes a mismatch of the right panel's contents with its schema. See XSLT Statements for more information about XSLT statements.
		Add Child. Adds a statement for a child element to the currently selected element.

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 Incorrect Mappings for more information.
		Edit Statement. Allows you to modify an XSLT statement for the element. See XSLT Statements 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 XPath Formula Builder 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 for a description of the sub-items of this menu.
	Undo <i>operation</i>	Rolls back the last operation performed. The name of the last operation is shown.
	Redo <i>operation</i>	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.
	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 39](#) 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.

Icons for schema items

Icon	Description
	Complex element that is a container for other datatypes. This is also called a <i>branch</i> in the schema tree.
	Simple string or character value.
	Simple integer value.
	Simple decimal (floating point) number.
	Simple boolean value.
	Simple Date or Time. This can be any of the following datatypes: <ul style="list-style-type: none"> • Time • Date • Date & Time • Duration • Day • Month • Year • Month & Year • Day & Month
	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 40](#) describes the additional qualifiers that appear next to the name of schema items.

Additional icons for hints

Qualifier	Scope Variables or Hint	Statement
	No qualifier indicates the element is required.	N/A
	A question mark indicates an optional Item.	An implicit "if" statement is set for this statement. This occurs when you map an optional element from the Scope Variables to an optional element in the function input schema or if you specify Surround element with if test on the Content tab of the Edit Statement dialog.
*	An asterisk indicates the item repeats zero or more times.	N/A
+	A plus sign indicates the item repeats one 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.

Function Argument Mapping Wizard for XPath 2.0

Using the Function Argument Mapping wizard you can map the variable of the source to the argument of a mapper function.

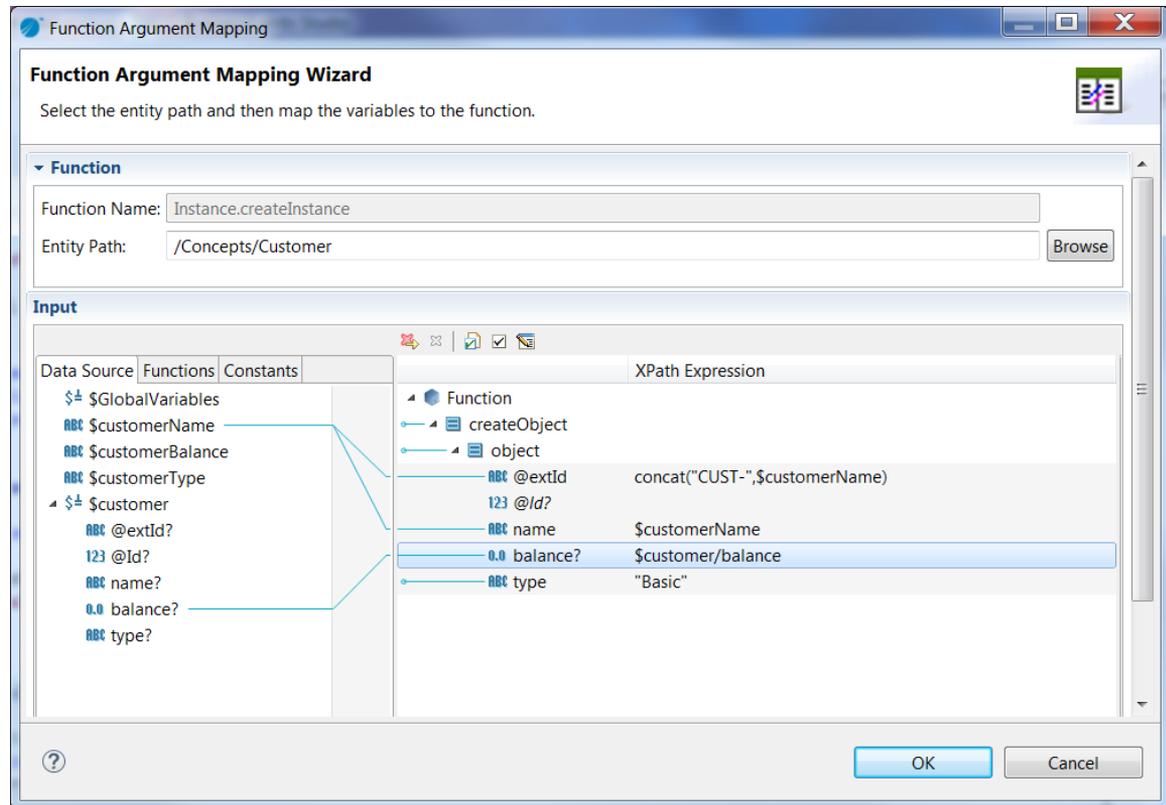
Functions Section

The **Functions** section is similar to the Function Argument Mapping wizard for XPath 1.0. It displays the view-only name of the function you are working with and the editable entity path to the item, whose properties and attributes you want to map to the function arguments.

Input Section

The **Input** section in the Function Argument Mapping wizard for XPath 2.0 is a combination of the Input section of the Function Argument Mapping wizard for XPath 1.0 and XPath Function Builder. For XPath 2.0, there is no separate XPath Function Builder but it has been merged in the Function Argument Mapping wizard. However, the Function Argument Mapping wizard still performs the same functions and in the same way with only the user interface changes.

The Function Argument Mapping Wizard in XPath 2.0 Project



In the Input section, on the left side, there are three tabs: Data Source, Functions, Constants.

The **Data Source** tab is similar to the Scope Variables panel in the Function Argument Mapping wizard for XPath 1.0. It lists the properties and attributes available to the function, as well as global variables defined in the project.

The **Functions** and the **Constants** tab are the same tabs which are available in the XPath Formula Builder for XPath 1.0. However, the Functions tab now contains the XPath 2.0 mapper functions. Some of these new functions were provided by TIBCO in XPath 1.0, which are now part of the XPath 2.0 function library. See [The XPath Formula Builder Reference](#) for more details on these tabs.

On the right side, similar to the Function panel in the Function Argument Mapping wizard for XPath 1.0 the XSLT template is present which specify how the Data Source would be transformed to provide the expected input. There is a toolbar which consists of the tools show the edit tab and auto-fix the errors.

Function Argument Mapping Wizard for XPath 2.0 Toolbar

Icon	Tool	Description
	Remove Mapping	Removes the selected mapping from the selected attribute.

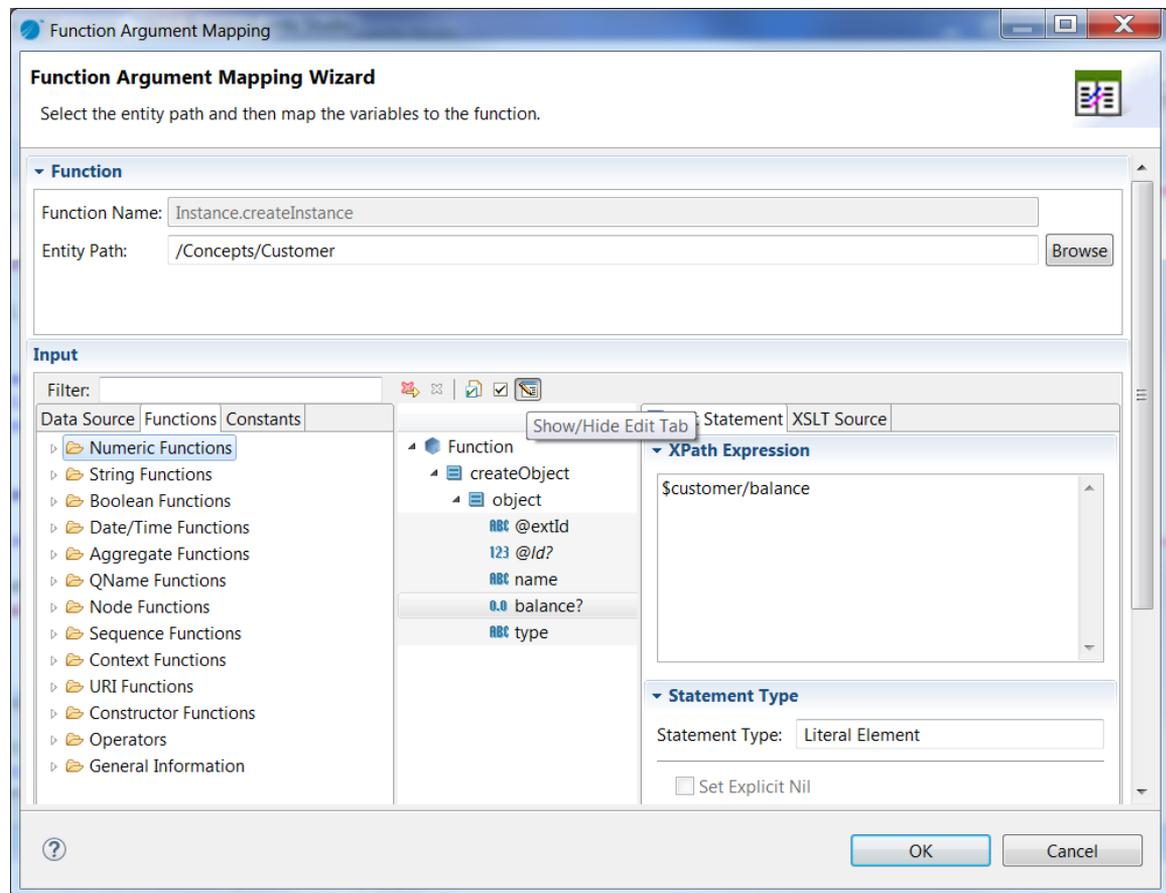
Icon	Tool	Description
	Show Check and Repair	<p>Displays mapper check and repair window similar to XPath 1.0. It verifies the XSLT template you have created in the Function panel against the expected input. A list of errors and warnings appear and you can choose which items you wish to fix.</p> <p>TIBCO BusinessEvents attempts to fix simple problems such as adding missing items that are expected and typecasting errors.</p>
	Fix Typecasting Error	<p>Fixes the typecasting error in XPath 2.0 where explicit casting is not done. For BusinessEvents, if the XPath expression elements cannot be implicitly cast, an error will be displayed. For instance, if the XPath expression is \$price > 1000, and \$price is defined as a String, then an explicit cast must be done (the XPath 1.0 mapper does this casting implicitly).</p>
	Show/Hide Edit Tab	<p>Toggles the XPath expression editor for building complex XPath statement for the selected input argument. See XPath Formula Builder for XPath 2.0.</p> <p>This builder functions in the same way in which XPath Function Builder was working in XPath 1.0. See XPath Formula Builder for more details.</p>

XPath Formula Builder For XPath 2.0

XPath 2.0 mapper do not have a separate window for XPath Formula Builder. The expression editor is now part of the Function Argument Mapping wizard and can be shown by the Show/Hide Edit Tab button.

Click the **Show/Hide Edit Tab** icon to display or hide the **XPath Statement** and **XSLT Source** tabs where you can build the complex XPath expression. You can use the data scope variables, mapper functions and constants from the respective tab on the left. In XPath 2.0, the behavior of the formula builder remains the same as XPath 1.0. See [XPath Formula Builder](#) for more details on its functioning in XPath 1.0.

XPath Formula Builder for XPath 2.0



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**  icon (XPath 1.0) or **Show/Hide Edit Tab**  icon (XPath 2.0).

Most options in the Edit Statement dialog are straightforward. However, there are some complex scenarios that require multiple steps.

You may also wish to refer to [XSLT Statements](#) for a reference of XSLT statements when deciding which XSLT statement can be used to achieve the result you desire.

You can specify XPath formulas to transform an element if you need to perform more complex processing. The XPath Formula Builder allows you to easily create XPath formulas. For more advanced use of XPath, see [XPath Formula Builder](#). There are also a variety of third-party books and resources about XSLT and XPath.

The datatypes of the function's arguments display as hints. Once a mapping or formula is specified, a hint becomes an XSLT statement. See [Statements Hints and Errors](#) for more information about hints and statements.

Statements Hints and Errors

When you display the Function tab, the existing statements are examined, and any input elements that do not have a statement are displayed as hints. Hints are reminders that you can specify a statement for the input element, but they are not stored as part of the XSLT template for the function's input. Hints are displayed in italics with a light blue background. Once you specify a mapping or a formula for a hint, the input element becomes a statement. You can also drag the hint to the left past the dividing line between the panels and the hint becomes a blank statement.

Once you specify a statement in the Function panel and click OK, it becomes part of the XSLT template used to create the input data. Statements are only deleted if you manually delete them using the delete button, or if you use the **Mapper Check and Repair** () button to automatically fix errors. Therefore, if the input schema for the function changes, your statements may no longer be valid. See [Tester Preferences](#) for more information about using the **Mapper Check and Repair** button to fix statements in the Function panel.

Any statement or hint that has an error is displayed in red. A hint is only displayed in red if it is a required input element. All required input elements must have statements specified. The **Mapper Check and Repair** button can help you automatically fix some errors. See [Incorrect Mappings](#) for more information about fixing errors.

Buttons Menus and Icons

The Input tab contains several toolbar buttons, popup menus, and icons. This section describes the various graphical elements of the Input tab.

Toolbar and Right-Click Menu on the Input Section

The Scope Variables panel and the Function panel have several buttons for performing various functions. There is also a popup menu when you right-click on elements in each panel. [Table 38](#) describes the buttons and right-click menu items available in the panels of the **Input** tab.

Input tab toolbar buttons

Button	Right-Click Menu	Description
Scope Variables Panel		
		Coercions. 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.

Button	Right-Click Menu	Description
	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.
		Move Up. Moves the selected element up in the Function tree.
		Move Down. Moves the selected element down in the Function tree.
	Move Out	Move Out. Promotes the selected element to the next highest level in the Function tree.
	Move In	Moves the currently selected element into a new statement. This displays the Move Into New Statement dialog that allows you to choose the statement you wish to move the element into. See XSLT Statements for more information about XPath statements.
	Delete	Deletes the mapping for the selected element. If no mapping is defined, the element itself is deleted (along with all child elements). Note Elements are predefined. Do not delete elements. Deletion of an element causes mapper validation errors due to the mismatch of the right panel's content with its schema.
		Insert. Click Insert to pop-up a New XSLT Statement dialog where you can define an XSLT statement. The statement is inserted in the function input schema on the same level of the hierarchy as the currently selected element. You can add one XSLT statement at a time with this button. The right-click menu item Statement provides a shortcut for multi-line statements, such as Choice or If. See the description of the Statement menu item below for more information. Note Elements are predefined. Do not add new elements. Doing so causes a mismatch of the right panel's contents with its schema. See XSLT Statements for more information about XSLT statements.
		Add Child. Adds a statement for a child element to the currently selected element.

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 Incorrect Mappings for more information.
		Edit Statement. Allows you to modify an XSLT statement for the element. See XSLT Statements 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 XPath Formula Builder 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 for a description of the sub-items of this menu.
	Undo <i>operation</i>	Rolls back the last operation performed. The name of the last operation is shown.
	Redo <i>operation</i>	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.
	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 39](#) 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.

Icons for schema items

Icon	Description
	Complex element that is a container for other datatypes. This is also called a <i>branch</i> in the schema tree.
	Simple string or character value.
	Simple integer value.
	Simple decimal (floating point) number.
	Simple boolean value.
	Simple Date or Time. This can be any of the following datatypes: <ul style="list-style-type: none"> • Time • Date • Date & Time • Duration • Day • Month • Year • Month & Year • Day & Month
	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 40](#) describes the additional qualifiers that appear next to the name of schema items.

Additional icons for hints

Qualifier	Scope Variables or Hint	Statement
	No qualifier indicates the element is required.	N/A
	A question mark indicates an optional Item.	An implicit "if" statement is set for this statement. This occurs when you map an optional element from the Scope Variables to an optional element in the function input schema or if you specify Surround element with if test on the Content tab of the Edit Statement dialog.
*	An asterisk indicates the item repeats zero or more times.	N/A
+	A plus sign indicates the item repeats one 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.



Element	Value
Order	
OrderId	
RequiredDate	
ShipName	
CustomerName	
Street	
City	
State	
Country	"USA"
PostalCode	94304
OrderDetails	

Constants can also be used in functions and search predicates. To learn more about complex XPath expressions that use functions and search predicates, see [XPath Formula Builder](#).

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 41](#) describes the validation behavior. Datatype validation listed with the prefix `xsd:` is defined in the namespace <http://www.w3.org/2001/XMLSchema>. See *XML Schema Part2: Datatypes* specification at <http://www.w3.org/TR/2004/PER-xmlschema-2-20040318/> for more information on the proper representation of these datatypes. Datatype validation listed with the prefix `xdt:` is defined in the namespace <http://www.w3.org/2003/11/xpath-datatypes>. See *Xquery 1.0 and Xpath 2.0 Functions and Operators* specification at <http://www.w3.org/TR/2003/WD-xpath-functions-20031112/> for more information on the proper representation of these datatypes.

Datatype validation

Data Type	Validation
boolean	xsd:boolean
double	xsd:double
string	xsd:string
dateTime	xsd:dateTime
long	xsd:long
int	xsd:int

Incorrect Mappings

Any incorrect statements are displayed in red in the Function panel.

Errors can occur for a number of reasons. For example,

- a required element has no statement, and therefore must be specified
- the function's input schema has changed and existing statements may no longer be valid
- the XPath formula for an element may contain an error

You should correct any errors before attempting to test or deploy your process definition.

To help find potential problems in your mappings, click the Mapper Check and Repair button. This button displays a dialog with all potential problems in the specified mappings. You can select the Fix checkbox for potential errors, and TIBCO BusinessEvents 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 check box in the Fix column for these items. For example, if an element expects a string and you supply a complex type, the corrective action to fix the problem is not clear, and therefore TIBCO BusinessEvents cannot automatically fix the problem. You must repair these items manually.

XPath 2.0

For XPath 2.0, in addition to the earlier specified errors, you can also see the errors when there is no explicit typecasting of different data types, such as, mapping a boolean value (`$processapplication/start`) to a input of data type double. In such cases, you can fix the typecasting error in following ways:

- Run the Mapper Function Migration wizard to fix the common mapper issues. See [Migrating Mapper Functions From XPath 1.0 to XPath 2.0](#) for more details.
- Click the **Fix Typecasting Error** icon to autofix all typecasting errors.
- Open Check and Repair window and select the Fix checkbox for the typecasting error to fix the selected error.
- Perform explicit typecasting using the constructor functions, such as, for earlier example use `xsd:double($processapplication/start)` to explicitly typecast the boolean value to double.

Repair Incorrect Mappings

If you want to return to the original expected Function input and remove all of the currently specified mappings, perform the following:

Procedure

1. Delete the root element of the function's input by selecting it and clicking the **Delete** button.

Click the **Mapper Check and Repair** button.

2. In the Mapper Check and Repair dialog Fix column, select the check boxes for all items.
3. Click **OK**.

Result

Alternatively, you can simply select the root input element and press the delete key on your keyboard as a shortcut for the procedure above.

After deleting all mappings and schema items and then repairing the input schema, the function's input reverts to the state before you open the Argument Mapping Wizard for the first time for this function.

Migrating Mapper Functions from XPath 1.0 to XPath 2.0

After the project is migrated from Xpath 1.0 to XPath 2.0, to fix some common mapping errors, you can use the Mapper Function Migration wizard.

This wizard fixes the typecast errors, if the data type can be converted using the `xsd: constructor` functions. It also fixes the issues which the gives the error: `XSLT is out of sync with schema component properties`.



This wizard cannot fix all mapper errors, but it fixes the common issues when switching from XPath 1.0 to XPath 2.0.



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

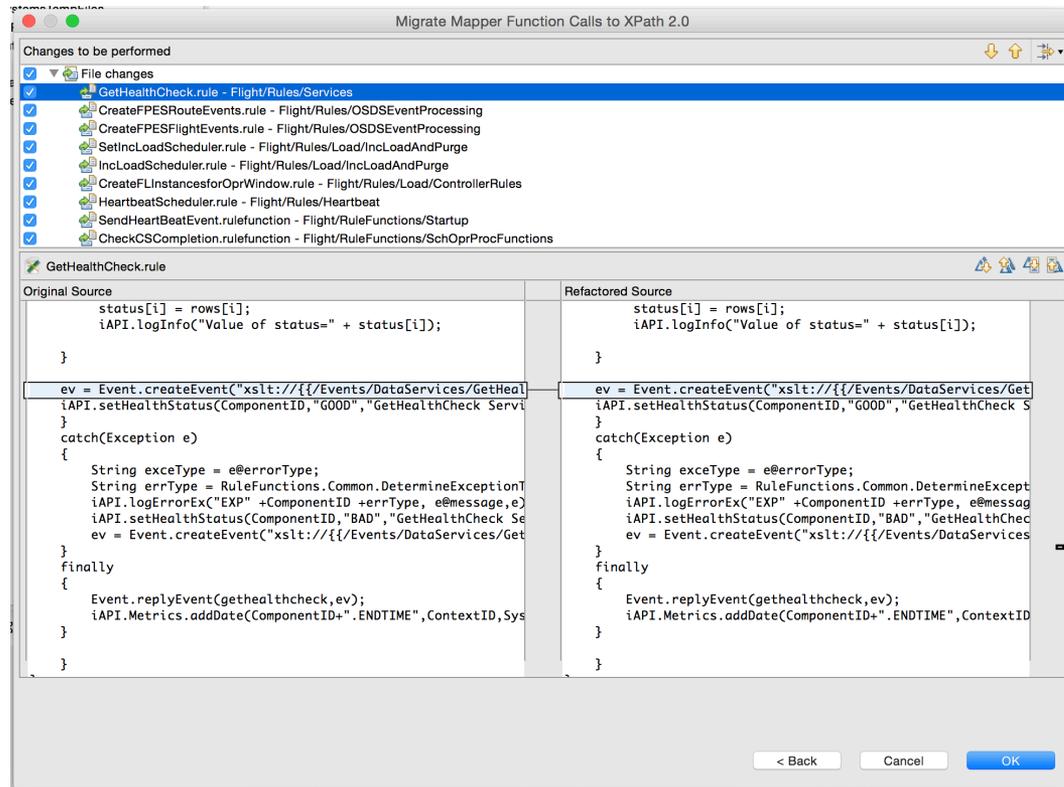
Procedure

1. In BusinessEvents Studio, right-click the project and select **Refactor > Migrate Mapper Function Calls**.

The Migrate Mapper Function Calls to XPath 2.0 window is displayed with information on the action that is performed on the project.

2. Click **Preview** to review the changes that is done to the project. The preview window is displayed with preview of all the changes.

Mapper Function Migration Wizard Preview Window



3. Select or clear the check box for each file after reviewing the changes. Click **OK** to apply the changes.

What to do next

After the wizard completes, you can revert the changes immediately using the **Edit > Undo File Changes** menu. However, after any additional changes to the resources in the workbench, you cannot use the **Undo** command.

Shortcuts

The Move In, Insert, Add Child, and Edit Statement buttons on the toolbar are ways to manually manipulate XSLT statements in the Function panel. These buttons, however, only add or modify one statement at a time. Also, there are some situations where you wish to convert a hint into a statement without performing any mapping. This section describes shortcuts for manipulating XSLT statements.

Statement Menu Options

When you select an element in the function input schema and right-click, a menu appears. The Statement menu item contains several sub-items that are useful shortcuts for creating XSLT statements.

- Surround with Choice: a shortcut for adding a choice statement and its associated conditions or otherwise statements around the currently selected element.
- Surround with If: a shortcut for adding an if statement and placing the currently selected element as the sub-element of the if.
- Surround with For-Each: a shortcut for moving the current element into a For-Each statement.
- Surround with For-Each-Group: a shortcut for moving the current element into a For-Each-Group statement and adding a Group-By grouping statement.
- Duplicate: a shortcut for creating a duplicate of the currently selected element (including any mappings or XPath formulas for the element). The duplicate is added below the currently selected element.
- Insert Model Group Content: a shortcut for inserting the contents of a selected model group into the mapper tree. The selected element in the function input schema is replaced by the contents of the model group you select.

Dragging to the Left

Dragging an element in the function input schema to the left past the divider between the two areas of the **Input** tab changes a hint into an XSLT statement.

This shortcut is useful in the following situations:

- When you have a complex element with no sub elements and no content.
- When you have a choice element, dragging to the left brings up the Mapping Wizard and you can choose a type for the element.
- When you have an element of type Any, dragging to the left brings up a dialog and you can specify the type for the element.

Cutting and Pasting

The Function panel is an XSLT template for specifying the function's input schema. You can choose any element in the Function panel and select Copy from the right-click menu or press the Control-C keys to copy the XSLT statement for the element. Once the XSLT is copied, you can paste it into a text editing tool to view or modify the code.

You can also paste arbitrary XSLT code into the Function panel using the right-click menu or the Control-V keys. Pasting XSLT code from the copy buffer places the code above the currently selected element in the Function panel.

Automatic Testing (at Runtime)

When you map Scope Variables elements to Function elements, the behavior of the mapping depends upon the types of elements you are mapping. In the simplest case of mapping a required element in the Scope Variables schema to a required Function element, the value of the Scope Variables element is assigned to the required Function element.

However, when elements are optional or nillable, more complex tests are necessary. When you drag the Scope Variables element to the Function element, the necessary tests are automatically placed into the Function XSLT template.

This section describes the result of mapping different types of elements. The types of mappings are described, then an example is given that illustrates these mappings and shows the XSLT code that is generated automatically when these mappings are performed at runtime.

Required to Required

Specifies that the statement should always include the required Function element and its value should be obtained from the required Scope Variables element that the element is mapped to.

Optional to Optional

Specifies that the statement should test if the Scope Variables element is present, and if so, include the optional element in the function's input. If the Scope Variables element is not present, the optional element is omitted from the function's input.

Nillable to Nillable

Specifies that both the Scope Variables and Function elements can be nil. Therefore, the value of the Function element is set to the value of the Scope Variables element. The value of the Function element is set explicitly to nil if that is the value of the Scope Variables element.

Optional to Nillable

Specifies that the statement should test if the optional Scope Variables element exists. If the element exists, the Function element should be created and set to the value of the Scope Variables element. If the Scope Variables element does not exist, the element is omitted from the function input schema.

Nillable to Optional

Specifies that the statement should test if the Scope Variables element has a value specified, and if so, the optional element in the Function panel should be set to the value of the Scope Variables element. Otherwise, if the Scope Variables element is nil, the optional element is omitted from the Function panel.

Optional & Nillable to Optional & Nillable

Specifies that if the optional Scope Variables element exists, then include the optional Function element in the input schema. If the Scope Variables element is nil, set the value of the Function element explicitly to nil. If the Scope Variables element is not nil, set the value of the Function element to the value of the Scope Variables element. If the Scope Variables element is not present, then omit the optional element from the function input schema.

Examples of Mappings

Some mappings require several steps to achieve the desired results. This section describes some complicated mapping scenarios and how to achieve the desired mappings using the tools available.



There are many methods to insert or modify XSLT statements in the function input schema. The examples in this section illustrate the simplest procedures to obtain the desired results. However, you do not have to follow the same procedures outlined in this section to achieve the correct set of statements.

Setting an Element Explicitly to Nil

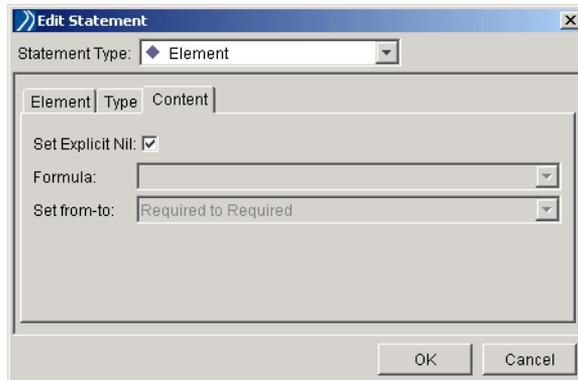
In some situations, you may wish to set an element explicitly to nil. One situation is when you wish to insert a row into a database table and you wish to supply a NULL for one of the columns.

Procedure

1. Select the input element you wish to set to nil.

Click the **Edit Statement**  icon (XPath 1.0) or **Show/Hide Edit Tab**  icon (XPath 2.0) on the **Input** tab toolbar.

- (XPath 1.0) Click the **Content** tab of the Edit Statement window and select the **Set Explicit Nil** check box.



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 .

- (XPath 2.0) Select the **Set Explicit Nil** check box under the **Statement Type** section under the **Edit Statement** tab.

Merging Input from Multiple Sources

You may have multiple items in the Scope Variables that you wish to map to one repeating element in the Function panel.

For example, you may have multiple formats for customer records and you wish to create a single, merged mailing list containing all customers in one format.

Example Schema of Merging Input From Multiple Sources



You wish to combine customers into a single repeating structure to create a mailing list.

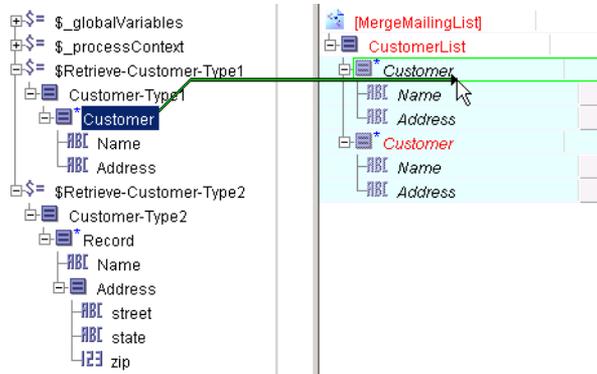
Multiple types of customers are retrieved, each having a different format for the address.

Procedure

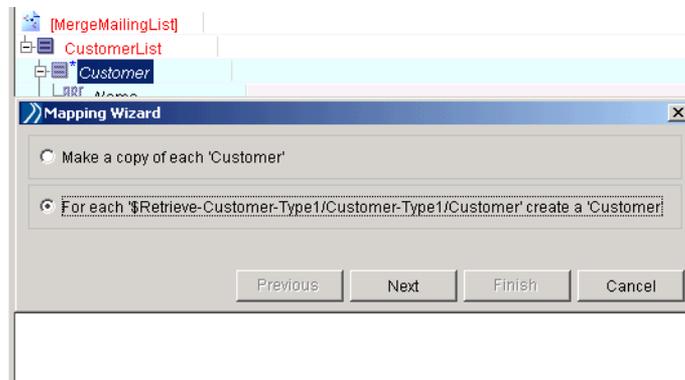
1. Select the repeating element in the Function panel, right-click, and select **Statement > Duplicate** from the menu.

Because you are creating two different formulas for mapping, you need two copies of the repeating element, one for each format. The resulting output contains only one repeating customer element, but the two copies in the Function panel make it simpler to perform two different mappings.

2. Map one of the elements from the Scope Variables to the first copy of the repeating element in the Function. For example, map `$Retrieve-Customer-Type1/Customer` to `MergeMailingList/ CustomerList/ Customer`.

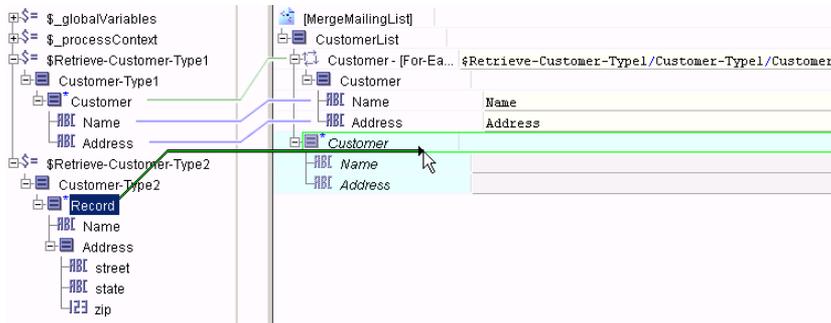


The Mapping Wizard dialog appears and presents choices for what you would like to accomplish. Choose the For Each option and click **Next**.

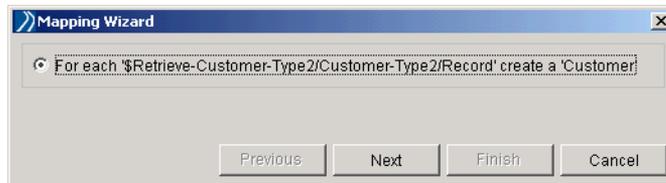


The mapping wizard asks if you wish to automatically map items with the same names. Click **Finish** to accept the default mappings.

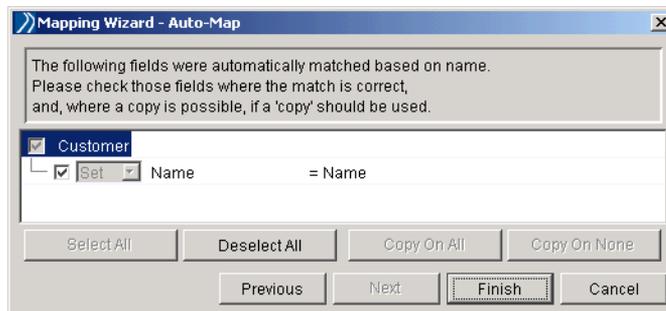
3. Map the other element from the Scope Variables to the second copy of the repeating element in the Function. For example, map `$Retrieve-Customer-Type2/Record` to `MergeMailingList/ CustomerList/ Customer`.



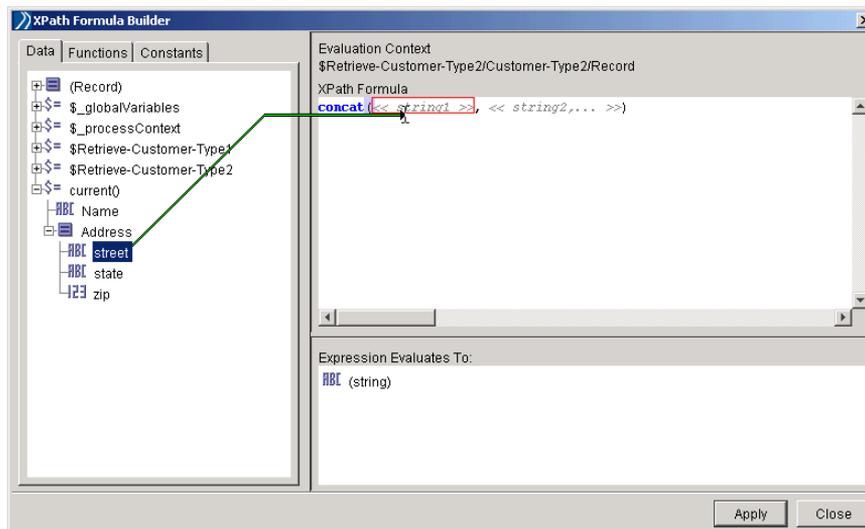
In the Mapping Wizard dialog, choose the For Each option and click **Next**.



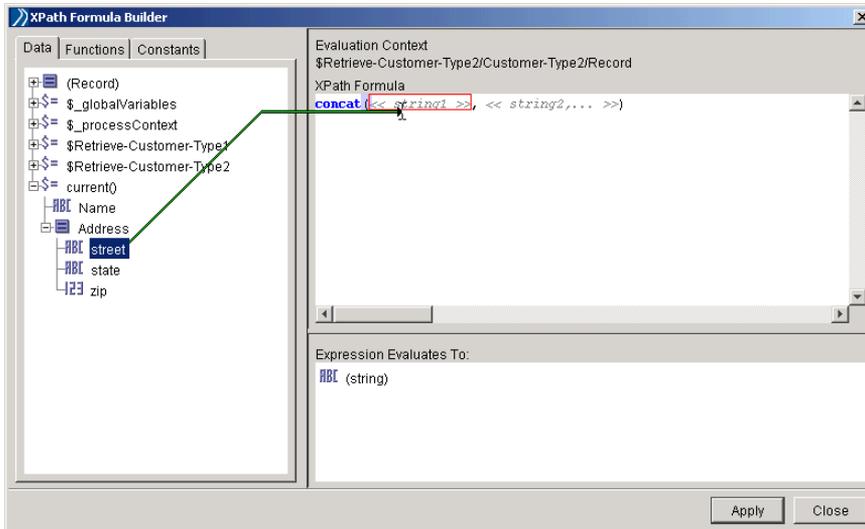
The mapping wizard presents you with an option to automatically map elements with the same name. Click **Finish** to accept the default mappings.



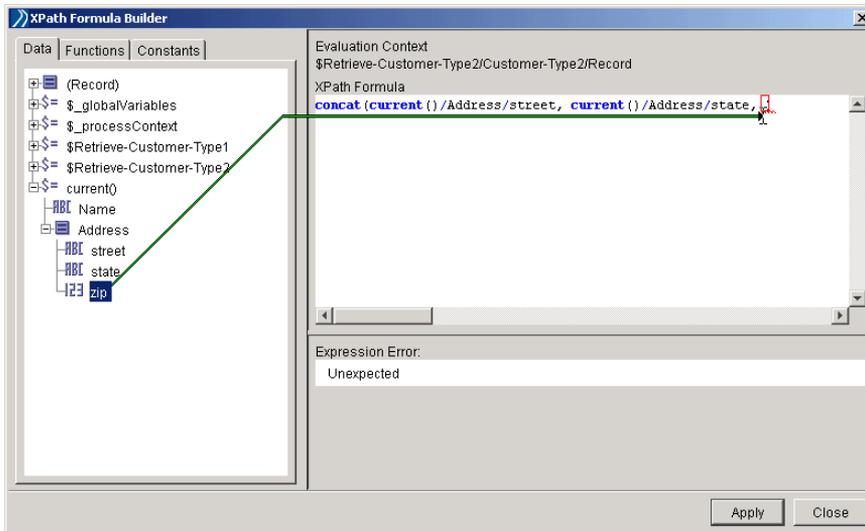
4. Select the Address element and click the XPath Formula Builder icon in the **Input** tab toolbar. In the XPath Formula Builder, drag a `concat()` function into the XPath Formula field. This function is used to concatenate the three elements in the Record element in the Scope Variables area to one Address element in the function's input.



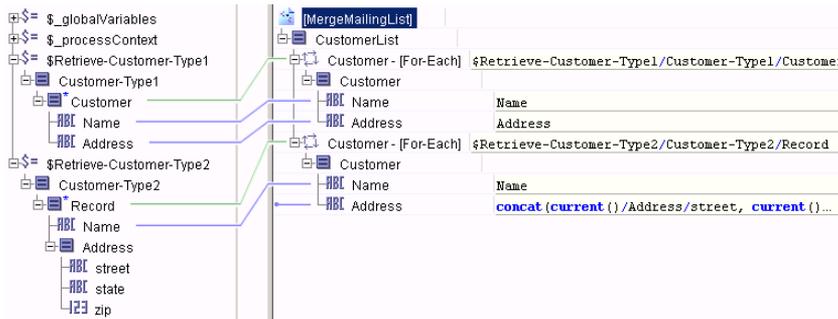
Click the **Data** tab, then drag the `$current()/Address/street` element into the `<< string1 >>` placeholder in the `concat()` function.



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.



5. Click **Apply**, then click **Close** to dismiss the XPath Formula Builder dialog.
6. This results in the following mapping:

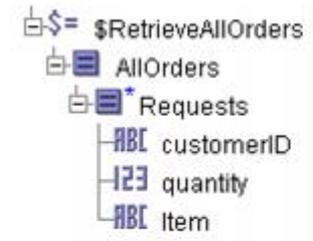


Converting a List Into a Grouped List

You may need to convert a flat list of items into a more structured list.

For example, you may have a list of all orders that have been completed. You might want to organize that list so that you can group the orders placed by each customer.

Example Schema of List Conversion to a Grouped List



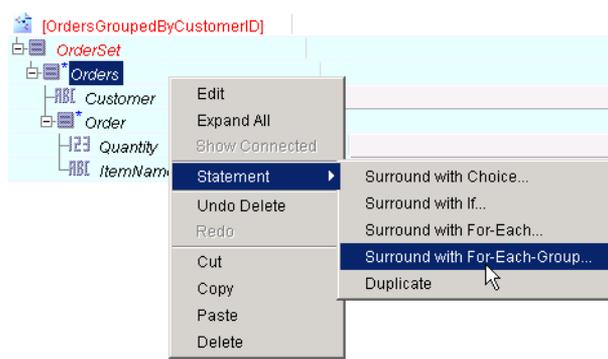
Flat list of orders placed by all customers. Each item in the repeating element `Requests` lists an order. The same customer can have multiple orders in this list.



The resulting schema is a repeating list of `Orders`, each item in the list is a customer ID. Each customer ID will, in turn, contain a list of orders.

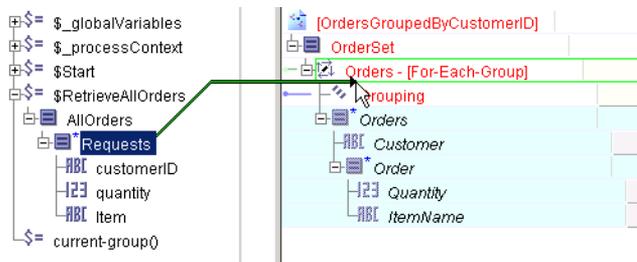
Procedure

1. Choose the repeating element in the function input schema that holds the grouped data. In this example, that element is `Orders`. Right-click on this element and choose **Statement > Surround with For-Each-Group...** from the pop-up menu. This is a shortcut to create a For-Each-Group statement with the `Orders` element as a child element and a Grouping statement to contain the element you wish to group-by.

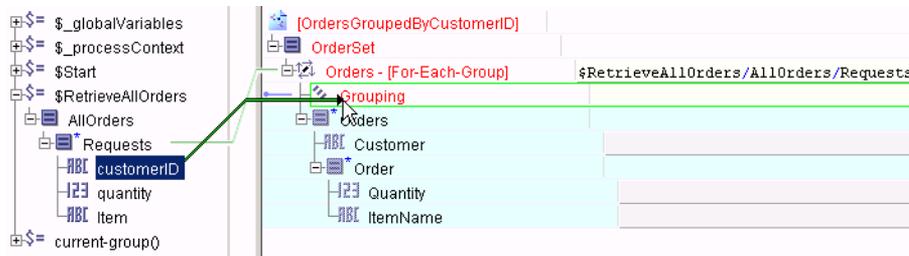


Adding the Grouping statement creates the `=$current-group()` element in the Scope Variables area. The Grouping statement creates the list grouped by the desired element, and the `current-group()` function allows you to access the items in the `Requests` repeating element that correspond to the group that is currently being processed.

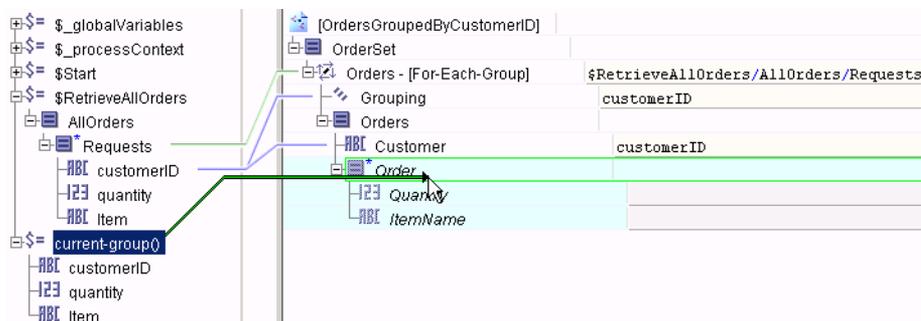
2. Drag the repeating element from the Scope Variables area to the For-Each-Group statement.



3. Drag the element you wish to group by from the Scope Variables area to the Grouping statement in the Function panel. In this example, `customerID` is the grouping element.



4. Map the `current-group()` element in the Scope Variables area to the repeating element `Order` under the `Customer` element in the Function panel.

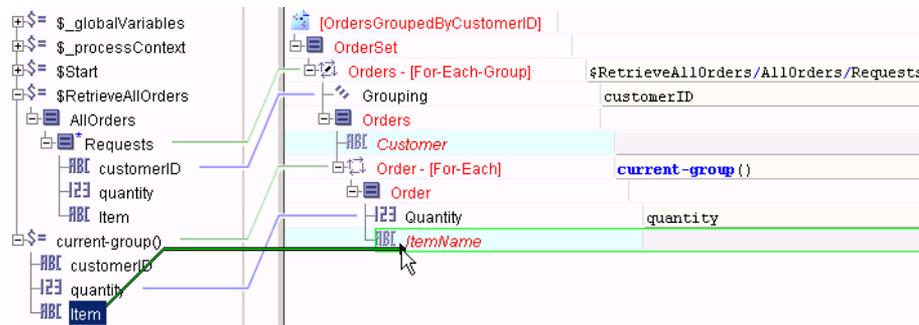


The default choice in the mapping wizard for this mapping is to create a For-Each. Choose this option in the mapping wizard.

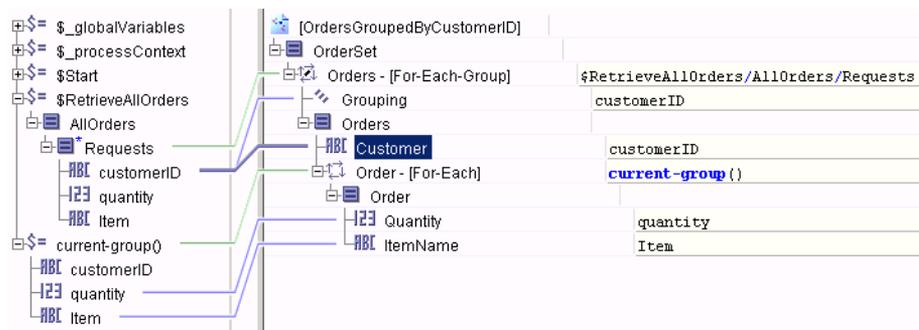


This creates an item in the `Order` list for each item in the current customer ID group that is being processed. The mapping wizard asks if you wish to map items with the same name in the current group and the orders group.

5. Map the remaining element from the `current-group()` element into the desired element in the For-Each group. In this case, `quantity` would map to `Quantity` automatically, and `Item` must be mapped to `ItemName`.



- Map the customerID element in the Requests element into the Customer element in the Function panel.

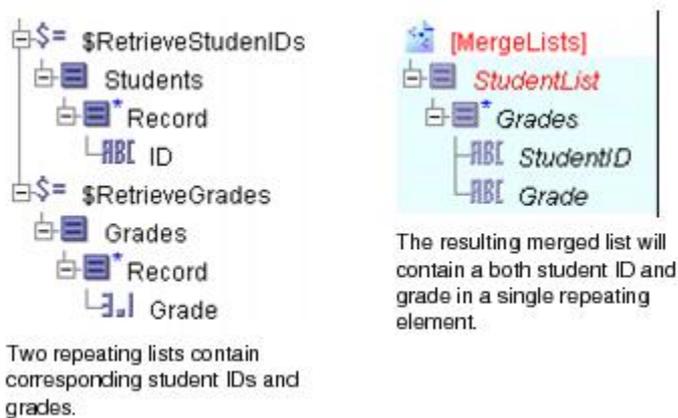


Merging Two Corresponding Lists

You may need to merge two lists that have corresponding items into one repeating element.

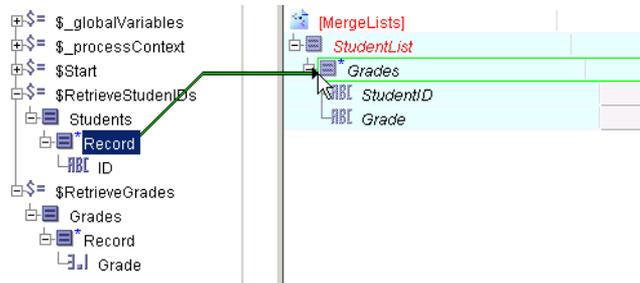
For example, you may have a list of student IDs and a list of grades, each grade corresponds to the student ID in the same position in the student ID list.

Example Schema of Merging of Two List

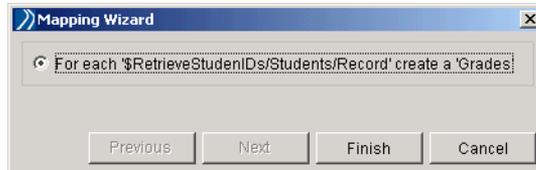


Procedure

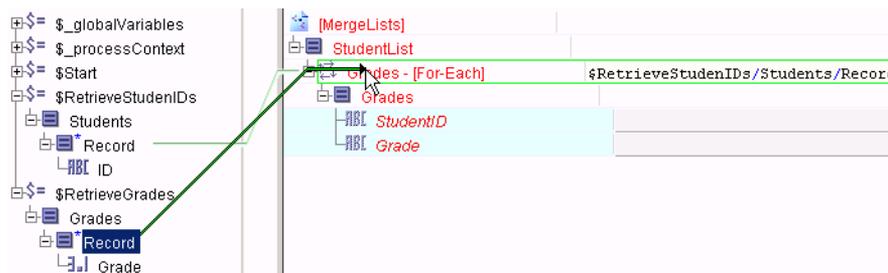
- Map the first repeating element from the Scope Variables area into the Grades repeating element in the Function panel. In this example, the \$RetrieveStudentIDs/Students/Record is the first repeating element.



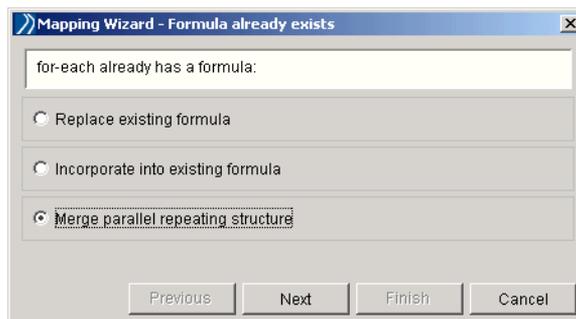
This brings up the mapping wizard with the default choice of creating a For-Each statement. Click **Finish** in the Mapping Wizard dialog to create the For-Each statement.



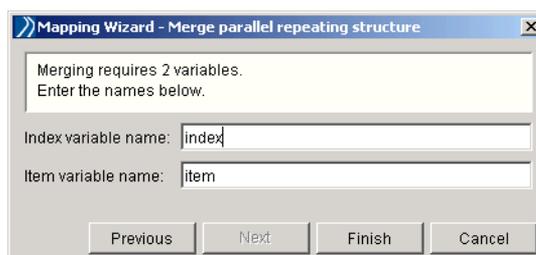
2. Drag the second repeating element into the For-Each statement.



3. The Mapping Wizard dialog appears asking you to choose an option. Choose the Merge parallel repeating structure option and click **Next**.



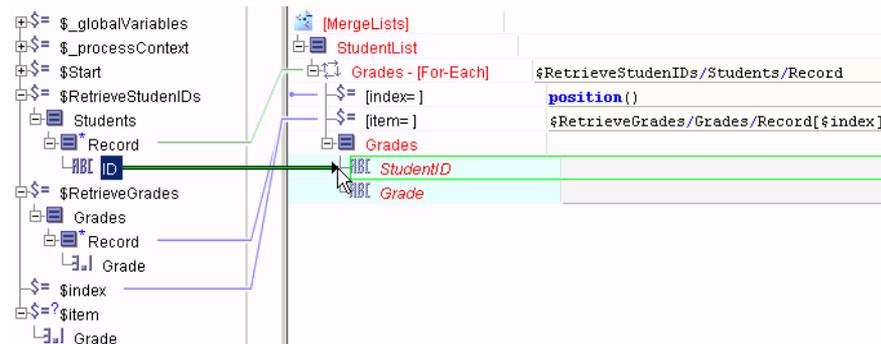
4. Merging two parallel repeating structures requires two variables. The mapping wizard prompts you to name these two variables. One variable is to hold the position number of the current item being processed, and the other variable is to hold the item in the second list that corresponds to the position 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.



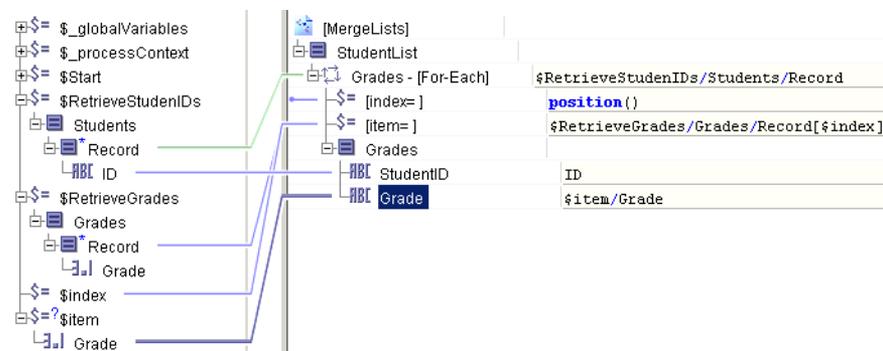
The two variables appear in the Scope Variables area once you have completed this step. The two variables also appear in the Function panel with the correct XPath statement to produce the desired result.

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.

- Map the ID element to the StudentID element in the function arguments.



- Map the `#[item=Grade]` element to the Grade element in the Function panel.

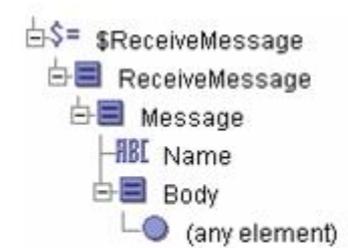


Coercions

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. Using the **Coercions** button in the **Input** tab toolbar you can create and manage your coercions.

The following example illustrates a schema with an element defined as the "any element" 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.

Example Schema of Coersion



The incoming message can have a body of type any element.



The function is expecting either an Order or a CreditLimitIncrease.

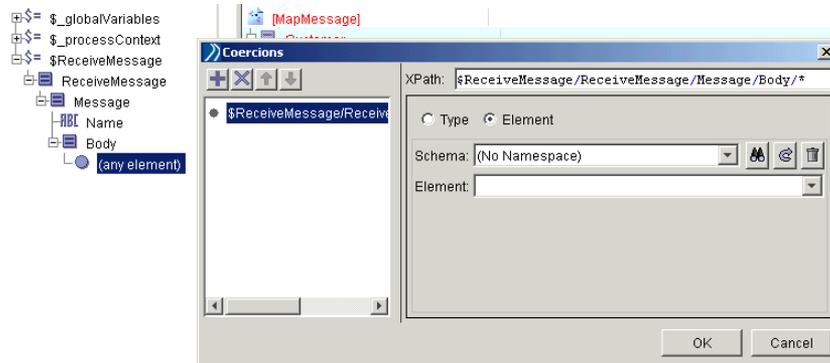
The following procedure describes how to coerce the `Body` element of the incoming message into a specific datatype and map it to a choice element.



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.

Procedure

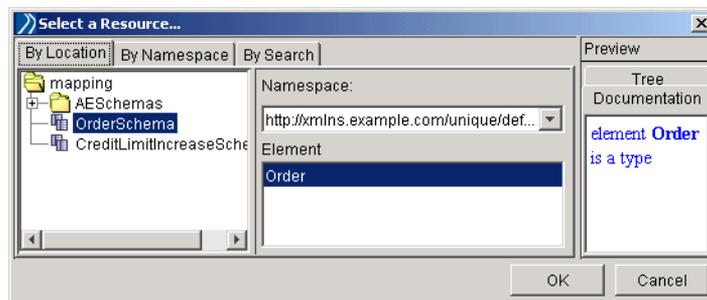
1. Select the element of type any element in the Scope Variables schema. Click the **Coercions** button in the **Input** tab toolbar. In the Coercions dialog, click the **Insert** button (+) to add a coercion for the currently selected element.



The Coercions dialog allows you to manage all of your coercions for a function in one dialog. You can create, modify, or delete coercions for any element in the Scope Variables schema using this dialog, not just the currently selected element. If you are creating a coercion for an element that is not currently selected, use the XPath field to specify the location of the element.

Click the **Element** radio button to specify that you are specifying a schema element.

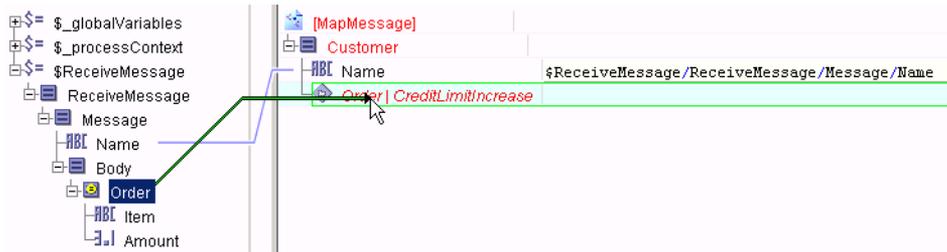
2. Click the Browse Resources button next to the Schema field to browse a list of schemas that can be used. In the Select a Resource... dialog, select the schema that you would like to specify



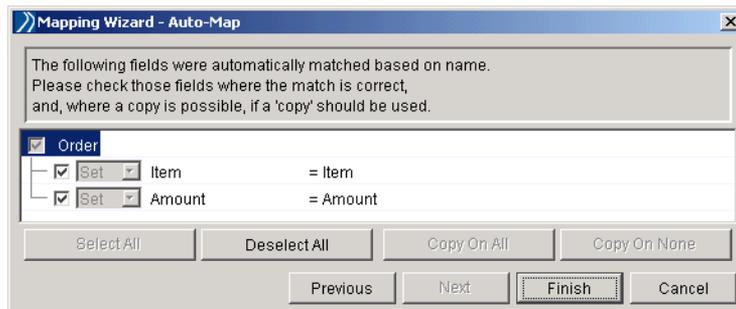
Click OK to coerce the element into the datatype of the selected schema element. The following would be the resulting schema where the element of the datatype any element has been replaced with the Order schema.



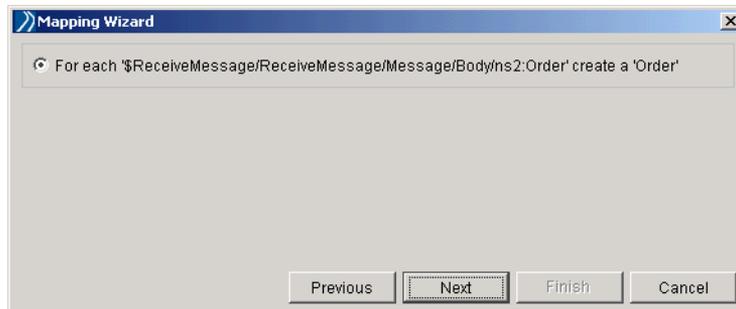
- Map the Name element to the Name element in the Function panel. Then, map the coerced Order element to the choice element in the Function panel.



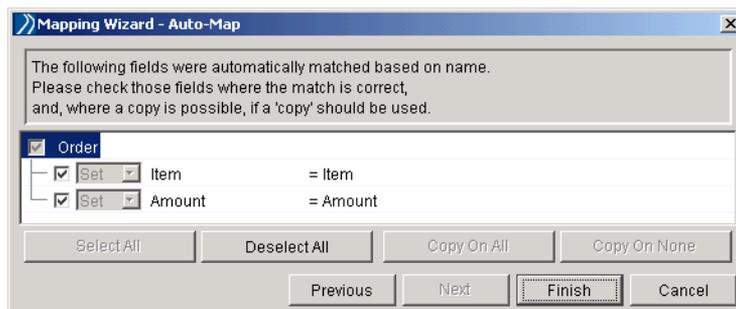
The Mapping Wizard dialog appears and asks if you wish to create an Order or a CreditLimitIncrease element. Select Order and click **Next**.



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.



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 is the completed mapping.



XSLT Statements

Using the XSLT statement you can set up the transformation of mapping.

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**  icon (XPath 1.0) or **Show/Hide Edit Tab**  icon (XPath 2.0), 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 ... >
  </ns0:when>
  <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.

XSLT Equivalent

```
<ns0:copy-of xmlns:ns0="http://www.w3.org/1999/XSL/Transform" select=""/>
```

Element

Creates an element with the specified name.

XSLT Equivalent

```
<elementName>value</elementName>
```

For-Each

Performs the specified statements once for each item in the selected node. This is useful if you wish to process each item of a repeating element once.

XSLT Equivalent

The following iterates over the list of files transferred from a ActiveMatrix BusinessWorks FTP Put activity and outputs an element with the name of each file for each file transferred.

```
<ns:for-each select="$FTP-Put/FTPputOutputFile/FileTransferred">
  <fileName>
    <ns:value-of select="$FTP-Put/FTPputOutputFile/FileTransferred/Name" />
  </fileName>
</ns:for-each>
```

For-Each-Group

Groups the items in a list by a specified element. This statement requires a Grouping statement to specify which element to group-by.

See [Converting a List Into a Grouped List](#) for an example of using the For-Each-Group statement.

XSLT Equivalent

```
<ns0:for-each-group xmlns:ns0="http://www.w3.org/1999/XSL/Transform" select="" />
```

Generate Comment

Places a comment element into the XSLT template. This comment 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"/>
```

XPath Formula Builder For XPath 1.0

TIBCO BusinessEvents uses XPath in the XPath Formula Builder, available in the Function Argument Mapper tool. You can use XPath, for example, when defining payloads for events TIBCO BusinessEvents also uses XPath as the language for defining conditions and transformations.

In XPath 1.0, the XPath Formula Builder opens in a separate window; however, in XPath 2.0, the XPath Formula Builder is part of the Function Argument Mapping wizard. They both function in the same manner with only difference in user interface. See [Function Argument Mapping Wizard for XPath 2.0](#) for more details on XPath Formula Builder for XPath 2.0.

XPath (XML Path Language) is an expression language developed by the World Wide Web Consortium (W3C) for addressing parts of XML documents. XPath also has basic manipulation functions for strings, numbers, and Boolean values.

To use XPath in TIBCO BusinessEvents, you need only be familiar with the basic XPath concepts, but you may wish to learn more about XPath when building complex expressions.

For a complete description of XPath, refer to the XPath specification (which can be obtained from www.w3.org).

TIBCO BusinessEvents uses XPath (XML Path Language) to identify elements whose content may be used, for, example in an event payload. You can also use XPath to perform basic manipulation and comparison of strings, numbers, and Boolean values.

Addressing Schema Elements

All Scope Variables and Function arguments are represented as an XML schema. Regardless of where the data comes from or its format, TIBCO BusinessEvents represents the data as a schema tree. The data can be simple (strings, numbers, Boolean values, and so on), or it can be a complex element. Complex elements are structures that contain other schema elements, either simple elements or other complex elements. Both simple and complex elements can also repeat. That is, they can be lists that store more than one element of the given type.

XPath is used to specify which schema element you would like to refer to. Each schema has its own associated structure, for example, a set of simple values or simple data and other complex data.

To reference a particular data item in a schema, you start with the root node and then use slashes (/) to indicate a path to the desired data element. For example, if you wish to specify the Street attribute in the ShipName complex element that is in the GetOrderInformation node, you would use the following syntax:

```
$GetOrderInformation/ShipName/Street
```

The path starts with a dollar sign to indicate it begins with a root node, then continues with node names using slashes, like a file or directory structure, until the desired location is named.

Evaluation Context

XPath also has a method for referencing relative paths from a particular node. If you have an *evaluation context*, or a particular starting node in a schema tree, you can specify the relative path to other elements in the tree.

For example, if your evaluation context is `$GetOrderInformation/ShipName`, then you can reference the sub-items of `ShipName` without specifying the entire path. If you wish to reference `$GetOrderInformation/RequiredDate`, the relative path would be `../RequiredDate`. The path is relative to the evaluation context — `RequiredDate` is one level higher in the schema tree than the elements of `ShipName`.

Namespaces

Some schema elements must be prefixed with their namespace. The namespace is automatically added to elements that require this when creating mappings or when dragging and dropping data in the XPath formula builder.

Search Predicates

An XPath expression can have a search predicate. The search predicate is used to locate a specific element of a repeating schema item. For example, a `$GetOrderInformation/OrderDetails/OrderItem` item is a repeating element. If you wish to select only the first item in the repeating element, you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[1]
```

The `[1]` specifies the first element of a repeating item.

Sub-items can also be examined and used in a search predicate. For example, to select the element whose `ProductId` is equal to "3A54", you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[ProductId="3A54"]
```

You can also use functions and expressions in the search predicate. For example, if you wish to find all elements after the first, you would specify the following:

```
$GetOrderInformation/OrderDetails/OrderItem[position()>1]
```

See the **Functions** tab of the XPath Formula Builder for a list of available functions available and online documentation.

Testing for Nil

Some elements can be explicitly set to nil. You can test an element to determine if it is set to nil or not. For example, the following XPath expression returns true if the `$Order/Item/OnSale` element is set to nil:

```
$Order/Item/OnSale/@xsi:nil="true"
```

Comments

You can add comments to XPath expressions using the XPath 2.0 syntax for comments. The syntax is:

```
{-- <comment here> --}
```

For example, the following XPath expression contains a comment:

```
$GetOrderInformation/ShipName/Street {-- returns the street --}
```

The XPath Formula Builder

You access the XPath Formula Builder using a button in the Function Argument Mapping Wizard. First select an item in the Function Argument panel (in the Input section). Then click the **XPath Formula**

Builder button ()

The XPath formula builder allows you to drag and drop schema elements and XPath functions to create XPath expressions. The schema elements, when dragged into the XPath Formula field, automatically become valid XPath location paths for the desired item. If a function is dragged into the XPath formula window, there are placeholders for each parameter of the function. You can drag and drop schema elements over the parameter placeholders to replace each placeholder.

[Table 42](#) describes the different areas of the XPath formula builder.

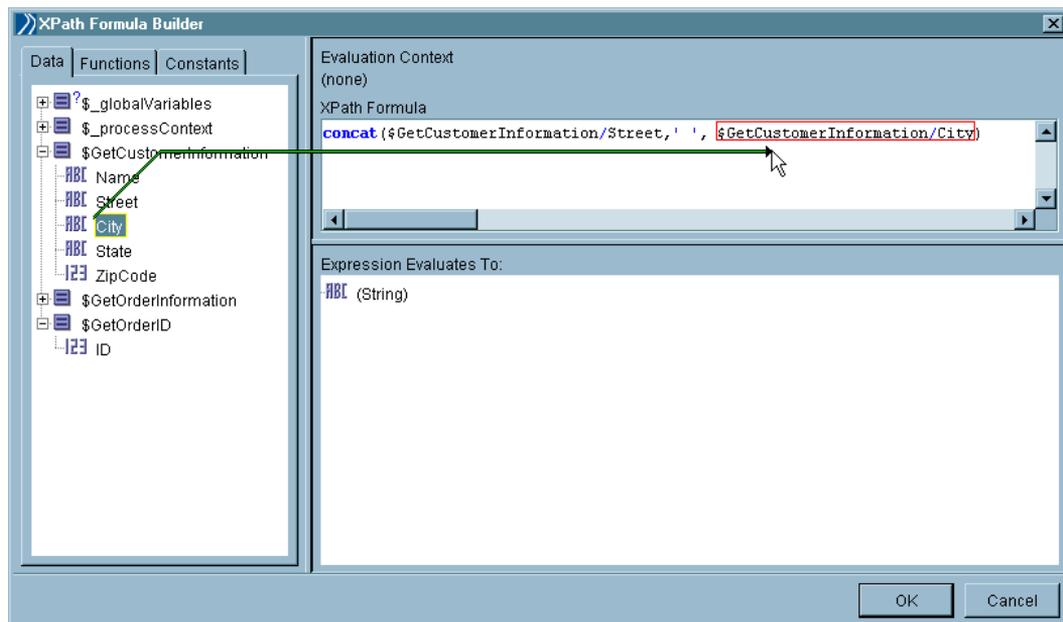
XPath Formula Builder Reference

Element	Description
Data tab	Displays the Scope Variables schema tree. All elements in this tree are available to drag and drop into the XPath Formula field.
Functions tab	<p>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.</p> <p>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.</p> <p>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.</p> <p>For more information about XPath functions, see the description of the function that is displayed when it is selected in the XPath formula builder.</p>
Constants tab	<p>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.</p> <p>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.</p>
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 for more information about the evaluation context.

Element	Description
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 4 illustrates using the XPath formula builder to create a valid function. The function concatenates the data elements `$GetCustomerInformation/Street` and `$GetCustomerInformation/City` and places a space between the two elements.

Creating an XPath formula



String Representations of Datatypes

When data must be represented in the input or output of an activity, the data is represented as a string. This section explains the string representations of various datatypes. TIBCO BusinessEvents follows the XPath 1.0 standard for representing all numeric datatypes. TIBCO BusinessEvents follows the XML Schema canonical format for all other datatypes.

Numeric Datatypes

Numeric datatypes include all types derived from `xs:integer`, `xs:decimal`, `xs:float`, and `xs:double`.

All decimal, float, and double numbers are compressed to an integer when represented, if there are only zeros following the decimal point (for example, "1.000" is represented as 1). Scientific notation is never used to represent a floating point number as a string (for example, "`xs:double('1.234E05')`") is represented as 123400). Data is truncated if the number of digits exceeds the maximum precision for the datatype (for example, "`xs:float('1.23456789')`") is represented as 1.2345679).

Both zero and negative zero are represented as 0. Positive and negative infinity are represented as `Infinity` and `-Infinity`. Not a number is represented as `NaN`.

Boolean

The boolean datatype is used to indicate a true or false state.

`xs:boolean('true')` and `xs:boolean('1')` are represented by `true`. The XPath function `true()` is also represented as `true`.

`xs:boolean('false')` and `xs:boolean('0')` are represented by `false`. The XPath function `false()` is also represented as `false`.

Date Datatypes

TIBCO BusinessEvents Function Argument Wizard (also known as the function argument mapper) implements dates in one of two ways. Either a date is stored as the number of milliseconds since January 1, 1970, or the date is implemented according to the XPath 2.0 or XQuery 1.0 standards as a set of normalized components (`xs:date`, `xs:time`, `xs:dateTime`, and so on) with an optional time zone offset. Activities that are associated with Java (for example, Java Code, Java Method, and so on) use the first implementation. Activities that are associated with XML (for example, Mapper, Parse XML, and so on) use the second implementation. The second implementation supports arbitrary precision of the seconds component.

Conversion between these representations may result in a loss of information either because of the difference in time zone representation or the precision of the seconds.

Date and Time Functions

There are some functions in the XPath formula builder using which you can parse or format strings that represent dates and times. These functions are:

- `format-dateTime(format, dateTime)`
- `format-date(format, date)`
- `format-time(format, time)`
- `parse-dateTime(format, string)`
- `parse-date(format, string)`
- `parse-time(format, string)`

The `format` parameter of these functions is based on the format patterns available for the `java.text.SimpleDateFormat` Java class. In the `format` parameter, unquoted alphabetic characters from A to Z and a to z represent the components of the date or time string. You can include non-pattern alphabetic characters in the string by quoting the text with single quotes. To include a single quote, use `'`. [Table 43](#) describes the alphabetic characters and their associated presentation in a date or time string.

Formatting characters in date or time strings

Letter	Description	Example
G	Era Four or more Gs return the full name of the era.	AD
y	year Two ys return two-digit year.	2003; 03

Letter	Description	Example
M	Month in year Three or more Ms 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
F	Day of week in month	3
E	Day in week Four or more Es return the full name of the weekday.	Friday; Fri
a	AM/PM marker Four or more as 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 44](#) illustrates some example date and time format patterns and the resulting string.

Example date and time format patterns

Date/Time Pattern	Result
"yyy.MM.dd G 'at' HH:mm:ss"	2003.3.11 AD at 09:43:56
"EEE, MMM d, 'yy"	Tue, Mar 11, '03
"hh 'o''clock' a, zzzz"	9 o'clock AM, GMT-8:00
"K:mm a"	0:08 PM
"yyMMdHHmmssZ"	010704120856-700

Cluster Deployment Descriptor

The Cluster Deployment Descriptor (CDD) is an XML file used to configure a project for deployment.

One EAR file and one CDD file define all the settings for all the engines and agents you want to deploy for a single application.

The CDD file is explained in detail in *TIBCO BusinessEvents Configuration Guide*.

TIBCO BusinessEvents and TIBCO Live Datamart Integration

TIBCO Liveview Web provides various visualizations, such as, charts, bars, graphs, and so on, to represent the data in real time. Using TIBCO LiveView Web you can also perform aggregations over the data. The visualizations are updated in real time based on the changes to the underlying entities of the BusinessEvents project (concept, event, or metrics). The charts could be pie chart to show status, bar charts to display trends in change in values, and so on.



The Live Datamart plugin for TIBCO BusinessEvents is available only in the TIBCO BusinessEvents Enterprise edition.

The following entities of a BusinessEvents project are supported:

- Concepts



History is not supported for Concepts.

- Events
- Metrics



Metric URLs and tracking fields are not supported.

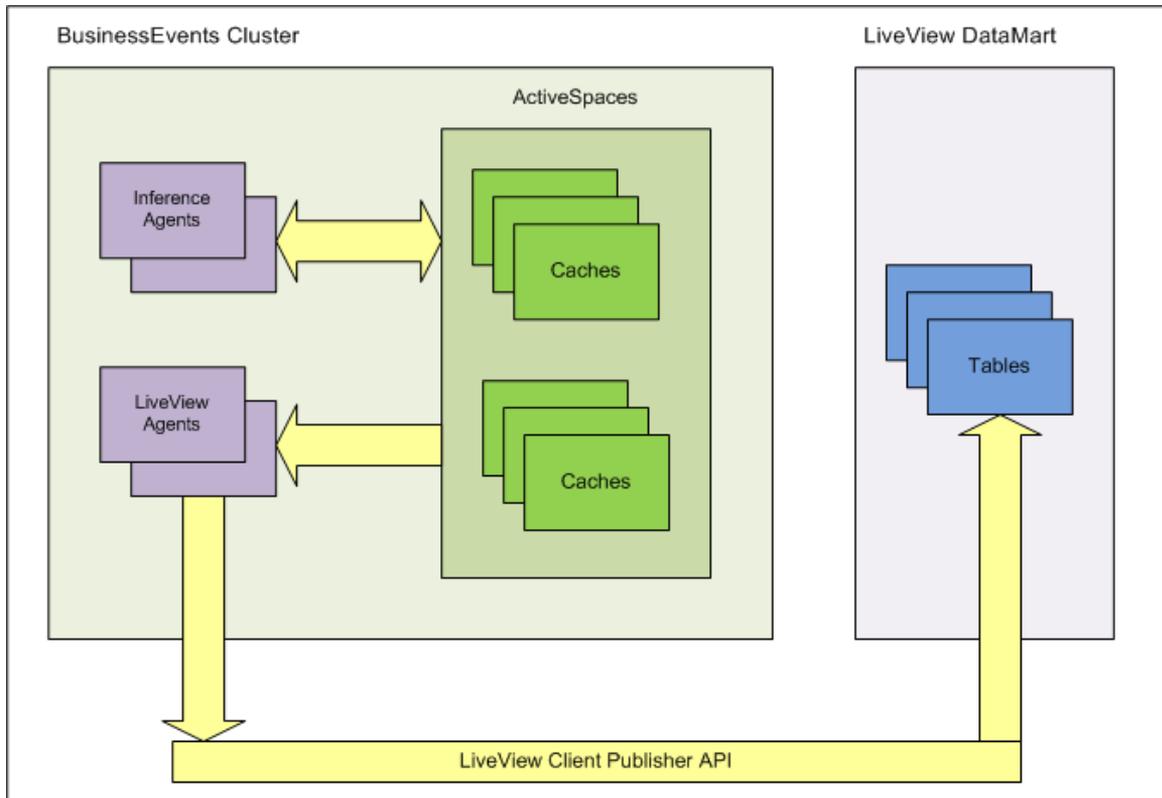
There are three main components of this integration:

- TIBCO StreamBase Server
- TIBCO Liveview Web
- TIBCO BusinessEvents Enterprise Edition

TIBCO BusinessEvents Enterprise Edition ships an example TIBCO BusinessEvents project with Liveview configuration files at the following location: `TIBCO_HOME\examples\liveview\FraudDetectionLiveView`.

High Level Architecture

In the high level, the BusinessEvents and LiveView DataMart uses LiveView client API, where BusinessEvents acts as a publisher, pushing data for the selected entities from BusinessEvents cluster to the StreamBase server. The LiveView Web connects to the underlying StreamBase server to provide support for the charting, quering, alerting, and so on. The LiveView client API provides the mechanism to create, modify, or delete te LiveView data tables.



It is recommended to periodically clean up or purge data from Live Datamart, otherwise it will run out of machine resources.

Displaying BusinessEvents Project Entities in LiveView Web

TIBCO LiveView Web Dashboard is used to showcase various visualizations based on the data sent from TIBCO Business Events.

Based on the project configuration, BusinessEvents generates an Live DataMart project, containing LiveView configuration files (that is, Live DataMart Tables) corresponding to BusinessEvents entities (concepts, events, or metrics). The `id` of the entities is used as primary key for these Live DataMart tables. TIBCO LiveView Web dashboard then uses data from the Live DataMart tables, and based on the visualization setup, displays the project metrics in different charts.

Prerequisites

- Install TIBCO StreamBase. Refer to *TIBCO StreamBase documentation* on how to install TIBCO StreamBase.
- Download TIBCO Liveview Web and setup LiveView Web with StreamBase. Refer to the *TIBCO LiveView Web documentation* on how to setup LiveView Web for a project or for all projects in the Live DataMart server.

Procedure

1. In the BusinessEvents Studio, add the **LiveView** agent class to the BusinessEvents project CDD. You can configure the LiveView agent class in the BusinessEvents project for Live Datamart server connection. Refer *TIBCO BusinessEvents Configuration Guide* for more details.



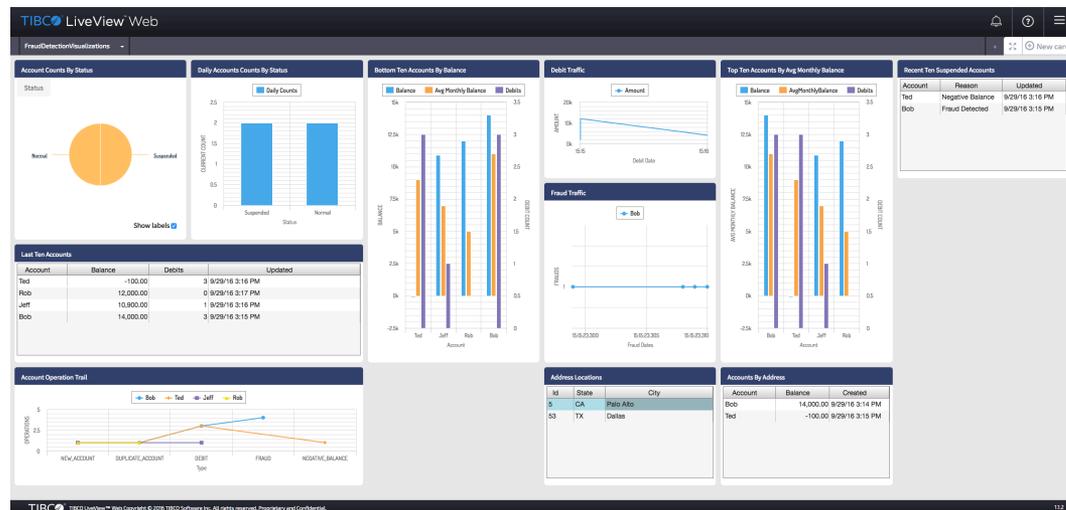
Ensure that in the CDD file, under the **Cluster** tab the **Store Properties As Individual Fields** check box is selected for the **Object Management: [Cache]** configuration. If this check box is not selected then the values for fields with contained and reference concepts are set to null, while others are set as expected.

2. Save the project.
3. If the **Generate LiveView Files** check box is selected for the LiveView agent, the LiveView configuration (.lvconf) files for the project are generated on saving the CDD file. Alternatively, generate the LiveView configuration (.lvconf) files for the project using the studio-tools utility. See [Generating TIBCO Live DataMart Configuration Files](#) and [LiveView Configuration File Reference](#) for more details.
4. Start the TIBCO Live DataMart server with the location of the generated LiveView configuration files for the project.
Refer to the *TIBCO StreamBase documentation* for more details on how to run the project.
5. Start the BusinessEvents engine with the BusinessEvents project configured for Liveview.
6. Open the Liveview Web URL (such as, <http://localhost:10080/lvweb>) to view the Liveview Web UI.
If an ACL is configured, there will be an authenticated login else, you are directly taken to the dashboard.
7. In the dashboard, you can create a page and card, while adding various visualization, aggregation, and alerts over the LiveView tables.

Refer to the *TIBCO LiveView Web documentation* for more details.

The following image shows the sample dashboard created in LiveView Web for the bundled example FraudDetectionLiveView:

LiveView Web Sample Dashboard



Generating Live DataMart Configuration Files Using BusinessEvents Studio Tools Utility

Using the TIBCO BusinessEvents Studio Tools utility you can manually generate LiveView Configuration (.lvconf) files for the BusinessEvents project.

Alternatively, the LiveView Configuration (.lvconf) files for the BusinessEvents project are generated automatically if the **Generate LiveView Files** checkbox is selected for the LiveView agent. Please refer to the *TIBCO BusinessEvents Configuration Guide* for more details on the configuration of the LiveView agent.

A separate LiveView configuration file is generated for each entity that is configured to be pushed to the TIBCO Live DataMart server.

Procedure

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 generateLDMPProject [-h] -o outputDirectory -p studioProjDir -c CDDFilePath -e EARFilePath
```

TIBCO BusinessEvents Studio Tools for Generating LiveView Configuration Files

Option	Description
-core generateLDMPProject	Within the core category of operations, specifies the generateLDMPProject operation to generate LiveView configuration files (.lvconf) for the BusinessEvents project.
-h	Optional. Displays help.
-o	Specifies the output directory to place the generated files. If not provided, the files are generated in the folder as defined in the Output Directory filed in the LiveView agent in the project CDD file.
-p	Specifies the path to the TIBCO BusinessEvents Studio project directory.
-c	Specifies the path to the CDD file of the project.
-e	Specifies the path to the EAR file of the project.

LiveView Configuration File Reference

The LiveView configuration file is generated by TIBCO BusinessEvents representing a BusinessEvents entity (such as, concept, event, or metric). The LiveView configuration file is used by Live DataMart Server to create the corresponding Live DataMart tables.

For each entity that is configured in the LiveView agent, a XML configuration file is generated. Each property of the instance is transformed to a field of Live DataMart table. The configuration file also record the datatype of each property. The id field is used as the primary key and the extid field is used as the secondary key for the table. Additional indexes for fields can be added based on the **Properties Metadata** section in the CDD file (**Cluster > Object Management > Domain Object > Overrides**).

The following sample shows the XML configuration for the an example concept Account:

```
<?xml version="1.0" encoding="UTF-8"?>
<liveview-configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.streambase.com/schemas/lvconf/">
<data-table id="Concepts_Account">
<fields>
<field name="Balance" type="double"/>
<field name="Debits" type="int"/>
<field name="Status" type="string"/>
<field name="AvgMonthlyBalance" type="double"/>
<field name="Created" type="timestamp"/>
<field name="Updated" type="timestamp"/>
<field name="Address" type="long"/>
<field name="FraudAttempt" type="int"/>
```

```
<field name="Reason" type="string"/>
<field name="id" type="long"/>
<field name="extId" type="string"/>
<field name="version__" type="int"/>
<field name="typeId__" type="int"/>
<field name="deleted__" type="bool"/>
</fields>
<primary-key>
<field ref="id"/>
</primary-key>
<indices>
<index>
<field ref="extId"/>
</index>
</indices>
</data-table>
</liveview-configuration>
```

TIBCO ActiveMatrix BusinessWorks 6 Integration

With the use of the TIBCO BusinessEvents API, you can run a TIBCO BusinessEvents engine from within BusinessWorks 6 to call rule functions, create events and concepts and assert events and concepts.

See TIBCO ActiveMatrix BusinessWorks documentation for more details on BusinessWorks 6.



For more information on BusinessEvents API, see online Java API Reference and MissManners and RuleFunctionAPI examples under `BE_HOME/examples/standard` to get familiar with the API.

Procedure

1. Install TIBCO BusinessEvents and TIBCO ActiveMatrix BusinessWorks 6.
2. Import the TIBCO BusinessEvents project RuleFunctionAPI/RFAPI from the standard examples in TIBCO BusinessEvents Studio.
 - Create an EAR for the same project or
 - Create a new TIBCO BusinessEvents project with Rulefunction and create an EAR for it.
3. Create a JAVA class to call the TIBCO BusinessEvents Rulefunction API.
 - a) Create a JAR file from the same JAVA class.
4. Create a TIBCO ActiveMatrixBusinessWorks 6 application with the JAVA INVOKE activity with the TIBCO ActiveMatrix BusinessWorks 6 Studio.
5. Add the JAR file and the required JARs from TIBCO `BE_HOME/lib` as listed below to add to TIBCO ActiveMatrix BusinessWorks 6 application's build path libraries.

The following are the TIBCO BusinessEvents Library that are required:

- `be-functions.jar`
 - `cep-common.jar`
 - `cep-kernel.jar`
 - `cep-base.jar`
 - `commons-lang3-3.2.jar`
 - `cep-ui-rt-common.jar`
 - `org.eclipse.uml2.uml.jar`
 - `org.eclipse.uml2.types.jar`
 - `TIBCOrt.jar`
 - `javassist.jar`
 - `antlr-3.2.jar`
6. Configure the JAVA Invoke Activity by using the `invoke` method from the `CallRuleFunction` class to call TIBCO BusinessEvents RuleFunction API.
 7. Provide the required JAVA activity input parameters, such as REPO URL, TIBCO BusinessEvents engine TRA file, CDD file, PU, Rulefunction name and so on.

These parameters are TIBCO BusinessEvents project artifacts required to call TIBCO BusinessEvents RuleFunction.
 8. Go to the TIBCO ActiveMatrix BusinessWorks 6 Application Module Dependencies and import the packages as listed below required to call the TIBCO BusinessEvents RuleFunction API.

Packages required from TIBCO ActiveMatrix BusinessWorks 6 include:

- org.eclipse.emf.ecore.resource.Resource\$Factory\$Registry
- org.eclipse.emf.ecore.EObject
- org.eclipse.emf.common
- org.eclipse.emf.common.util.URI
- org.eclipse.emf.ecore.resource.impl.ResourceSetImpl
- org.eclipse.emf.ecore.impl.EPackageImpl
- org.eclipse.emf.ecore.xml.type.XMLTypePackage
- org.eclipse.emf.ecore.xmi.impl.EcoreResourceFactoryImpl
- org.eclipse.emf.ecore.util.FeatureMap
- org.eclipse.emf.common.notify.Notification
- org.apache.log4j.Appender
- org.apache.log4j.spi.LoggerRepository
- com.tibco.xml.tns.impl.TargetNamespaceCache
- com.tibco.util.StringUtilities
- com.tibco.xml.schema.SmType
- org.eclipse.emf.ecore.xmi.XMLOptions
- com.tibco.xml.schema.impl.SmNamespaceProviderImpl
- com.tibco.xml.ws.wsdl.WsException
- com.tibco.xml.tns
- com.tibco.xml.tns.impl
- com.tibco.xml.tns.parse
- com.tibco.xml.tns.parse.helpers
- com.tibco.xml.data.primitive
- com.tibco.xml.datamodel
- com.tibco.io.xml
- com.tibco.xml.schema.channel.SchemaModelProvider
- com.tibco.xml.validation.kernel.DefaultSchemaCache
- com.tibco.xml.channel.error.helpers.ErrorThrower
- com.tibco.xml.schema.parse.SmParseSupport
- com.tibco.xml.schema.build.MutableSchema
- com.tibco.xml.datamodel.helpers.XiChild
- com.tibco.xml.schema.helpers.NoNamespace
- com.tibco.xml.schema.flavor.XSDL
- com.tibco.sax.ResolverUtilities
- com.tibco.xml.data.primitive.values.XsBoolean
- com.tibco.xml.datamodel.navigation
- com.tibco.xml.xquery

9. Deploy the application in TIBCO ActiveMatrix BusinessWorks 6 as "BW Application" or export the application to EAR and deploy in the TIBCO ActiveMatrix BusinessWorks domain with TIBCO Enterprise Administrator.

TIBCO Enterprise Runtime for R Integration

You can integrate BusinessEvents with TIBCO Enterprise Runtime for R to delegate statistical operations to TIBCO Enterprise Runtime for R engines.

R Scripts

The R script is a text file containing R language commands. The file is stored with extension `.r`. The R script can be stored either inside a project or outside of a project. You can also refer to the supplied example `FraudDetectionTerr` at `BE_HOME\examples\standard\FraudDetectionTerr` for a working example of BusinessEvents and TIBCO Enterprise Runtime for R integration.

Catalog Functions for TIBCO Enterprise Runtime for R

You can call TIBCO Enterprise Runtime for R engines and use their functionalities from BusinessEvents using catalog functions. Using the various TIBCO Enterprise Runtime for R catalog functions you can perform the following operations:

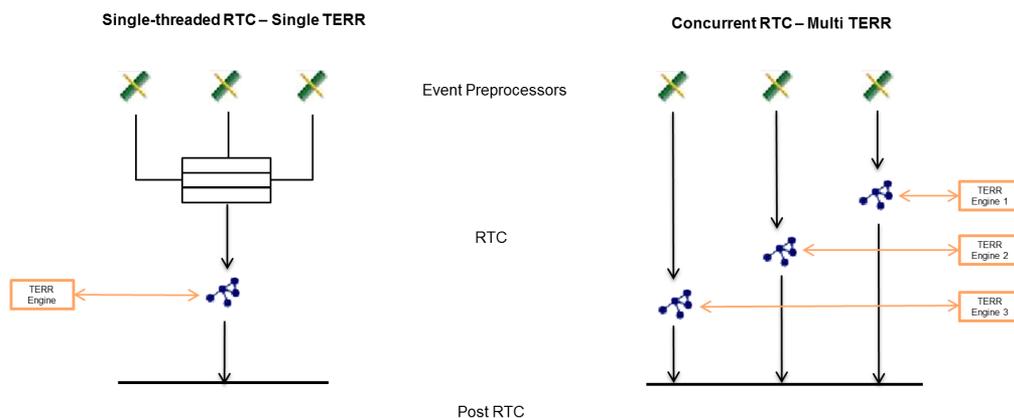
- **Lifecycle operations of one or more TERR engines:** create, start, stop, delete, check if it is running, get TIBCO Enterprise Runtime for R error message, and so on.
- **Data manipulation:**
 - Loading R scripts into TIBCO Enterprise Runtime for R engines.
 - Executing R functions from previously loaded R scripts.
 - Generating complex data structures like Datalist or Dataframe and sending them to TIBCO Enterprise Runtime for R engines for processing.
 - Receiving complex data structures like Datalist and Dataframe from TIBCO Enterprise Runtime for R engines for R and extracting its data.

See **TIBCO BusinessEvents > CEP Analytics > Analytics** in *TIBCO BusinessEvents Functions Reference* documentation for more details.

Run to Completion (RTC) Cycle

TIBCO Enterprise Runtime for R engine is a single threaded engine. Thus to scale better, use concurrent RTC with multiple TIBCO Enterprise Runtime for R engines. Each RTC thread carries its own TIBCO Enterprise Runtime for R engine, see figure [RTC and TIBCO Enterprise Runtime for R Engine](#). For example, for a project configured with Shared Queue (**Thread Count** is 3) and **Concurrent RTC**, each thread name (`Engine.threadName` function) is mapped to a TIBCO Enterprise Runtime for R engine. Thus, three threads are mapped to three TIBCO Enterprise Runtime for R engines.

RTC and TIBCO Enterprise Runtime for R Engine



Datatype Mapping

See [BusinessEvents and TIBCO Enterprise Runtime for R Mapping Reference](#) for mapping details on different datatypes of BusinessEvents and TIBCO Enterprise Runtime for R.

Configuring BusinessEvents for TIBCO Enterprise Runtime for R

Configure the BusinessEvents engine TRA file to set up the TIBCO Enterprise Runtime for R location.

Prerequisites

TIBCO BusinessEvents and TIBCO Enterprise Runtime for R must be installed on the same machine.

Procedure

1. Open the `BE_HOME/bin/be-engine.tra` file for editing.
2. Edit the environment variable `TERR_HOME` with the absolute path where TIBCO Enterprise Runtime for R is installed.

```
tibco.env.TERR_HOME=<absolute_path_where_TIBCO_Enterprise_Runtime_for_R_is_installed>
```



You can also run BusinessEvents with TIBCO Enterprise Runtime for R TIBCO BusinessEvents Studio for debugging purposes. When running the BusinessEvents engine from BusinessEvents Studio, perform the following configurations:

- Add `TERR_HOME` variable under the **Environment** tab for the launch configurations of the project. Assign the absolute path, where TIBCO Enterprise Runtime for R is installed, as the value of `TERR_HOME`.
- Add `terJava.jar` library under the **Classpath** tab for the launch configurations of the project.

See [Adding and Working with Launch \(Debug or Run\) Configurations](#) on page 292 for more details on how to configure launch configurations.

3. Start or restart the BusinessEvents engine.

BusinessEvents and TIBCO Enterprise Runtime for R Mapping Reference

BusinessEvents and TIBCO Enterprise Runtime for R uses different data types, which are mapped to specific types on information exchange.

BusinessEvents to TIBCO Enterprise Runtime for R Mapping

BusinessEvents Type	TERR Type
Int / Int []	Integer / Vector(Integer)
Long / Long[]	Integer / Vector(Integer)
Float / Float[]	Numeric / Vector(Numeric)
Double / Double[]	Numeric / Vector(Numeric)
String / String[]	Character / Vector(Character)
Boolean / Boolean[]	Logical / Vector(Logical)

BusinessEvents Type	TERR Type
DateTime / DateTime[]	Character / Vector(Character)

TIBCO Enterprise Runtime for R to BusinessEvents Mapping

TERR Type	BusinessEvents Type
Integer / Vector(Integer)	Integer / Integer[]
Numeric / Vector(Numeric)	Double / Double[]
Character / Vector(Character)	String / String[]
Logical / Vector(Logical)	Integer / Integer[] <ul style="list-style-type: none"> • TRUE is mapped to '1' • FALSE is mapped to '0' • NA is mapped to '-1'
DataList	Object Use catalog functions to get the data (<code>DataList.getElement</code>).
DataFrame	Object Use catalog functions to get the data (<code>DataFrame.getColumn</code>).
Factor	Not supported. Note: You can modify the data type in TIBCO Enterprise Runtime for R, before sending it to BusinessEvents.

Performance Profiler

You can run TIBCO BusinessEvents Performance Profiler utility to gather statistics about activities that occur during each RTC cycle. This information helps to identify bottlenecks in the project, which can often be addressed by redesigning rules or other aspects of a project.

The Profiler utility collects statistics relating to the run to completion (RTC) rule evaluation cycle in an inference agent. The utility does not collect data about object management. It also does not collect data for other types of agents. It helps you understand, for example, internal execution times and frequencies of rules.

The Profiler records time spent during each RTC on activities such as 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.

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

There are different ways you can turn the 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 *TIBCO BusinessEvents Configuration Guide*.

Using Properties

Set the following properties in the Cluster Deployment Descriptor (CDD) Processing Unit tab, for all processing units (engines) whose RTC performance you want to profile.

Profiler Configuration Properties (Sheet of)

Property	Notes
<code>be.engine.profile.Agent_Class_Name.enable</code>	
	If set to true, enables Profiler for the specified agent class (<i>Agent_Class_Name</i>) when the agent initializes. Default is false.
<code>be.engine.profile.*.enable</code>	
	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.
<code>be.engine.profile.Agent_Class_Name.file</code>	
	The name of output file that the Profiler writes to, for the specified agent (<i>Agent_Class_Name</i>). Default behavior is as follows: If <code>be.engine.profile.*.file</code> is specified and <code>be.engine.profile.Agent_Class_Name.file</code> is not specified, then the file name is the value of <code>be.engine.profile.*.file</code> , with the <i>Agent_Class_Name</i> appended. If the properties <code>be.engine.profile.*.file</code> and <code>be.engine.profile.Agent_Class_Name.files</code> are not specified the name is created as follows: <code>be-profile_Agent_Class_Name.csv</code>
<code>be.engine.profile.*.file</code>	

Property	Notes
	<p>The default (prefix for the) name of the output file that the Profiler writes to. In all cases, the appropriate <i>Agent_Class_Name</i> is appended.</p> <p>Default name is <code>be-profile.csv</code> and it is located under the current working directory, if file name is not specified.</p>
<code>be.engine.profile.Agent_Class_Name.duration</code>	
	<p>Specifies the duration of profile data collection in seconds, for the specified <i>Agent_Class_Name</i>.</p> <p>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.</p> <p>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.</p> <p>Default is -1.</p>
<code>be.engine.profile.*.duration</code>	
	<p>Specifies the duration of profile data collection in seconds, for all agents.</p> <p>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.</p> <p>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.</p> <p>When <code>be.engine.profile.Agent_Class_Name.duration</code> and <code>be.engine.profile.*.duration</code> are both present, the duration specified in <code>be.engine.profile.Agent_Class_Name.duration</code> takes precedence.</p> <p>Default is -1.</p>
<code>be.engine.profile.Agent_Class_Name.level</code>	
	<p>Level of depth that profile data will be collected for the specified agent (<i>Agent_Class_Name</i>):</p> <ul style="list-style-type: none"> • -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). <p>Default is -1.</p>
<code>be.engine.profile.*.level</code>	

Property	Notes
	<p>Level of depth that profile data will be collected for all agents:</p> <ul style="list-style-type: none"> -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). <p>When <code>be.engine.profile.Agent_Class_Name.level</code> and <code>be.engine.profile.*.level</code> are both present, the level specified in <code>be.engine.profile.Agent_Class_Name.level</code> takes precedence.</p> <p>Default is -1.</p>

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 [Profiler Configuration Properties](#).

To turn the Profiler on

In your rule or rule function, use the following function to turn on the Profiler:

```
Engine.Profiler.startCollectingToFile(String fileName, int level, long duration)
```

The above function turns on the TIBCO BusinessEvents Profiler 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 [Profiler Configuration Properties](#):

`String fileName`: See `be.engine.profile.Agent_Class_Name.file`

`int level`: See `be.engine.profile.Agent_Class_Name.level`

`long duration`: See `be.engine.profile.Agent_Class_Name.duration` (Here `Agent_Class_Name` is the current agent in which the rule or rule function that calls this function is 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.

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 [Profiler Configuration Properties](#).

Before you Begin

Ensure that the property `hawk.enabled` is set to `true` in the CDD at the cluster level before the TIBCO BusinessEvents engine starts.

To turn the Profiler on

Use the following method to turn on the Profiler:

```
StartFileBasedProfiler(String session, String fileName, int level, long duration)
```

The above method turns on the TIBCO BusinessEvents Profiler for the specified agent. The Profiler starts collecting data at the beginning of next RTC for the specified duration.

This method works the same way as the `Engine.Profiler.startCollectingToFile()` function (see [Turning Profiler On and Off](#)), except that it requires you to specify an agent class.

Input arguments are the same as the engine properties shown in [Profiler Configuration Properties](#):

`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.

Profiler Column Heading Reference (Sheet of)

Column Heading	Notes
Statistics Relating to the Overall RTC	

Column Heading	Notes
RTC_Stats_Type	Type of rule evaluation cycle (RTC) There are 10 different types of RTC: <ul style="list-style-type: none"> • 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.
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: <ul style="list-style-type: none"> • CONDITION-Filter • Condition-Join
Timestamp	The timestamp of the first time the RTC begins.

Column Heading	Notes
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 <code>//... Only applies to CONDITION_Stats_Type</code>
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.
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.

Column Heading	Notes
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.
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: <ul style="list-style-type: none"> • 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.

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.

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*

You can create test data and set breakpoints before you run debugger, or using views within the Debug perspective.

Preparing to Run (Test) or Debug a Project

You can do these tasks in any order. For information on setting up see [Tester Preferences](#).

Build an EAR File

Procedure

1. In TIBCO BusinessEvents Studio, select the project in the BusinessEvents Studio Explorer panel, then select **Project > Build Enterprise Archive**.
2. If you want to build an EAR with debug information, select the **Generate Debug > Info** check box (selected by default).
3. In the File Location field provide the path to the EAR file and specify the EAR file name, for example, *c:\myprojects\myproject.ear*.
4. Generate the EAR file.

Create Test Data

You can create test data for use across multiple sessions, or you can provide rule input data while the engine is running. See [Test Data](#).

For Remote Debugging Only Configure Java Debug Interface (JDI)

To configure for remote debugging you configure the *be-engine.tra* file on the remote engine so 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](#). You must specify the same JDI port number in the TRA file and in the debug configuration.

Configure Java Debug Interface (JDI)

For each TIBCO BusinessEvents engine you want to enable for remote debugging, perform the following actions.

Procedure

1. Open the `BE_HOME/bin/be-engine.tra` file for editing.
2. Specify the port on which you want the engine to listen, using the environment variable `tibco.env.JDI_PORT`, for example:

```
tibco.env.JDI_PORT 5192
```

Where 5192 is the default value. If multiple engines run on the same machine, ensure that each has a unique port.

3. Uncomment the following line:


```
-Xrunjdpw:transport=dt_socket,address=%JDI_PORT%,suspend=na,server=y
```
4. Start or restart the engine with the `-d` (or `-debug`) command.

Adding and Working with Launch (Debug or Run) Configurations

Launch configurations are of two types: *run configurations* and *debug configurations*. All launch configurations have the same basic fields, but debug configurations have extra settings for remote debugging.

Before you begin, ensure you know the EAR file location, the CDD file location, and the name of the processing unit (configured in the CDD file) that you want to use. (The processing unit runs as an engine.)

Add and Work with Launch Configurations

A reference to the settings is provided in [Launch Configurations Reference](#).



For remote debugging you first have to set properties in the remote engine TRA file. See [For Remote Debugging Only Configure Java Debug Interface \(JDI\)](#).

Procedure

1. Perform one of the following actions.
 - To create a debug configuration, click the down-arrow to the right of the debugger button () on the toolbar. From the 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. Depending on your requirements, perform one of the following actions.
 - To edit a configuration, expand TIBCO BusinessEvents Application or Remote TIBCO BusinessEvents, and select an existing debug configuration.
 - To add a 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](#), perform the following actions.
 - Select the **Main** tab and configure values as explained in [Launch Configurations Reference](#).
 - Select the **Classpath** tab and configure the classpath for external libraries or custom functions as needed, for example if the project uses Rendezvous or JMS channels. See [Working with External Library and Custom Function Paths](#) for details.
 - Select the **Environment** tab and configure environment variables as needed, to run or debug the project in TIBCO BusinessEvents Studio. You can add new variables. You can select and then edit existing variables. You can append your edited variable to the existing environment variable, or you can replace the existing environment variable with it. For example if a custom function depends on a native library, you can add the path to that library using the `PATH`, `LD_LIBRARY_PATH`, `SHLIB_PATH`, or `LIBPATH` variable, as appropriate for your operating system.



On Linux platforms when TIBCO BusinessEvents DataGrid is used, you must set `LD_LIBRARY_PATH` to `AS_HOME/lib` in the **Environment** tab.
 - If you need to display characters that are not present in the system's default charset on the console, set the encoding to UTF-8.
 - Select the **Common** tab and set the **Encoding** to Other. Then select **UTF-8** from the drop-down menu.

This procedure is documented in **Plug-in Development Environment Guide > Reference > Launchers > JUnit Plug-in Test Launcher > Common Tab**.
 - For information on standard Eclipse features incorporated into this area, see Eclipse help.
5. If you selected Remote TIBCO BusinessEvents Application in [step 2](#), you must connect to the remote engine. Click the **Remote** tab and specify the host name or IP address and port the engine is running on. Use the same JDI port number in the TRA file and in the debug configuration (see [For Remote Debugging Only Configure Java Debug Interface \(JDI\)](#)).
6. Click **Apply** to save configuration settings.
7. Perform one of the following actions.
 - Click **Close** and save the configuration you worked on.
 - If you have done the setup as explained in this chapter and are ready to run or debug, click **Debug** to save the configuration and 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](#).

Launch Configurations Reference

See [Adding and Working with Launch \(Debug or Run\) Configurations](#) 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 <code>-V</code> , <code>-D</code> , <code>-X</code> and so on. For example to set global variables use: <code>-VVariable=value</code> .
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	

Field	Notes
Remote Connection	Enter VM arguments for running the remote VM. Alternatively, add these to the <code>java.extended.properties</code> property in the remote application's runtime properties (TRA) file. Default value is: <pre>-Xdebug - Xrunjdpw:transport=dt_socket,address=25192,suspend=n,server=y</pre>
Host	The host name or IP address of the remote computer where you are running the TIBCO BusinessEvents engine. Default is <code>localhost</code> .
Port	Communication port for the debugger, on the remote machine. Default is <code>25192</code>

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.



If the test data gets asserted in Cache Agent then the studio shows a message "Test data cannot be asserted" along with a "Cannot assert Object when activeMode is false" message on console as cache agents are not actively participating in processing inputs. Only Inference Agents processes inputs and the test data should be asserted into Inference Agent.

Working with Concept and Event Test Data

You can enter data in the Test Data editor for event payloads, and for concept, event and scorecard properties, including properties that are primitive types, array types, contained concepts, and reference concepts. If the concept, event or scorecard properties are associated with a domain model, then the test data gets populated with the values in the domain model. You can use global variables.

Creating Concept and Event Instance Test Data

Procedure

1. To create test data in BusinessEvents Studio Explorer (or in the Debug perspective BusinessEvents Studio Explorer view), you can perform either of the following steps:

- Right-click on an event or concept and click **New**, then **Test Data or Project**
 - Right-click **Project**, then **Test Data**.
2. Right-click an event or concept and click **Create Test Data**.
The Test Data editor appears showing the event or concept properties as column headers.
 3. In the Test Data editor, click **Add** to add rows for new instances. You can add your own unique **extId** values to the test data input, as needed. You can also use global variables.

You can also remove existing rows by selecting one or more rows and clicking **Remove**.
 4. 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](#)).

Edit Test Data for Concepts and Events in BusinessEvents Studio Explorer

To add more test data or edit test data you created earlier, perform the following actions.

Procedure

1. In BusinessEvents Studio Explorer, expand the `TestData` folder in the root of the project.
The `/TestData` folder is the default location.
2. Drill down to the test data you want to edit. The folder structure matches the project's event and concept folder structure.
3. Double-click the name of the test data file you want to edit. The test data editor opens. Add, remove, and edit rows of test data as desired.

Test data filenames use the format `eventName.eventtestdata` and `conceptName.concepttestdata`.

In test data editor, you can click on heading hyperlink to open the linked resource (event or concept).

Edit Concept and Instance Test Data in the Rule Input View

Procedure

1. Open the Rule Input view, if it is not already shown. To make the Rule Input View visible, perform one of the following actions.
 - Select the Debug perspective as follows: select **Window > Open Perspective**, or click the Open Perspective () button). Then select **Other > Debug**. The views associated with the Debug perspective open.
 - In the **Window** menu, click **Show View > Other**. Expand **TIBCO BusinessEvents**, and select **Rule Input**. Click **OK**.
2. Click the **Rule Input** tab and then select the **Tester Data** bottom tab.
For each concept or event for which you created test data, you see one row showing the project path to that concept or event.
3. Double-click the row for the concept or event whose test data you want to edit.
The Test Data editor appears, showing the rows of test data already created.
4. Edit as desired.

Working with Rule Data

To create and save rule Data:

Procedure

1. Open the Rule Input view, if it is not already shown. To make the Rule Input View visible, perform either of the following steps, as needed:
 - Select the Debug perspective: Select **Window > Open Perspective** , or click the **Open Perspective** () button). Then select **Other > Debug** . The views associated with the Debug perspective open.
 - In the **Window** menu, click **Show View > Other**. Expand **TIBCO BusinessEvents**, and select **Rule Input**. Click **OK**.
2. Provide input from the mapper as explained in [Using the Function Argument Mapper](#).
3. Specify the Launch Target. This is generally the locally running engine. The other fields become active.
4. Specify an event or concept in the Entity URI field. If you specify an event, then specify the Destination URI. Specify the Rule Session (agent) to use.
5. Perform either of the following steps:
 - Click **Save** to save the input values.
You can reuse these saved values for repetitive tests.
 - Click **Load** to load the input values from an existing XML file.
 - Click **Assert** to assert the data to a running engine.

Setting Breakpoints in Rules and Rule Functions

Setting breakpoints is an Eclipse feature. This section provides only basic information. You can also use advanced features such as importing and exporting breakpoints, and using class prepare breakpoints. See Eclipse help for more details on all breakpoint functionality. You can set or change breakpoints during a debug session also.

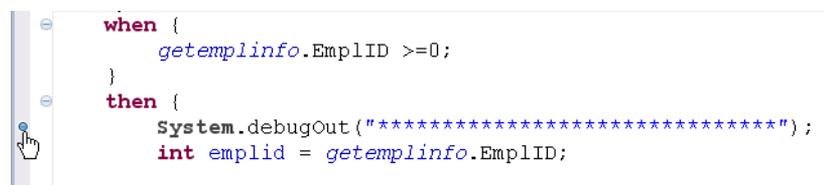


You cannot step through custom Java code.

Procedure

1. In TIBCO BusinessEvents Studio, open the source editor for a rule or rule function.
You can work with breakpoints in the debugger perspective as well as in the TIBCO BusinessEvents Studio development perspective.
2. To add a breakpoint put your cursor in the left margin (gray area) next to a row where you want to add a breakpoint. Perform one of the following steps.
 - Right click and select Toggle Breakpoint.
 - Double-click in the left margin.

A breakpoint icon appears in the left margin:



3. Set and adjust breakpoints as needed. Select a breakpoint, right-click, and perform any of the following steps.
 - To disable a breakpoint, select **Disable**.
 - To remove a breakpoint, select **Toggle Breakpoint**. Or you can double-click the breakpoint.
 - To edit a breakpoint's properties, select **Breakpoint properties**. A dialog displays (with mostly runtime options). For example, you can use a class prepare breakpoint (so the running program is suspended when the specified class or interface is first loaded by the Java VM).

A breakpoint may not be set exactly where you place it. This is because TIBCO BusinessEvents ensures that breakpoints fall on an executable statement, and moves any that do not to the nearest executable statement.



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. Then run debugger as explained below.



For projects containing an RV channel, add the path to the location of the RV libraries by adding `RV_HOME/lib` to the existing path.

As needed, switch to Debug perspective. Select **Window > Open Perspective**, or click the **Open Perspective** () button). Then select **Other > Debug**.

Alternatively, wait till TIBCO BusinessEvents prompts you to change to debug perspective. This happens when the debugger reaches the first breakpoint.

Click the down-arrow to the right of the debugger () button. You see a drop-down list. Perform one of the following steps.

- 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.

Procedure

1. Assert data as explained in [Asserting Rule Input Data](#).
2. View the results as explained in [Viewing the Results](#).
3. Use the standard Eclipse commands such as step into (F5), step over (F6), step return, step return, and so on, depending on what you want to examine.

See the options in the **Run** menu and in the **Breakpoints** tab for more options, and use Eclipse help for details with these Eclipse features.

When you are finished, click the **Terminate** () button to stop the engine.

Running Tester

Make sure you have completed setup as needed. You can create test data before you run tester, or using views within the Debug perspective. Other setup must be done before you begin.



For projects containing an RV channel, add the path to the location of RV libs:

See the following sections for details:

First do the setup needed for your testing. See the following sections:

- [Preparing to Run \(Test\) or Debug a Project](#)
- [Adding and Working with Launch \(Debug or Run\) Configurations](#)
- [Test Data](#)

Then run tester as explained below.

Run the Tester

As needed, switch to Debug perspective. Select **Window > Open Perspective** , or click the **Open Perspective** () button). Then select **Other > Debug** .

Select **Run > Run Configurations** or click the down-arrow to the right of the **Run** () button and choose **Run Configurations**.

If you have configured favorites, you can click the down-arrow to the right of the **Run** () button and choose a favorite.

Procedure

1. At the Run Configurations dialog select a run configuration and click **Run**.



If you have already started an engine using a run configuration and want to start it again, click the Run () button or Select **Run > Run** .

A TIBCO BusinessEvents engine starts, using parameters provided in the run configuration, if any were provided.

2. Assert data as explained in [Asserting Rule Input Data](#).
3. View the results as explained in [Viewing the Results](#).

When you are finished, click the **Terminate** () button to stop the engine.

Asserting Rule Input Data

You can assert either tester data, or rule data for use in Tester or Debugger.



To perform any task in the **Rule Input** tab, you must keep the engine running.

You can also send messages to destinations, as you would at runtime.

See also [Test Data](#) for details about creating and saving data.



Test data you created earlier for concepts, events, or scorecards appears in the Debugger perspective. By default the view appears in the middle on the left.



For cache application using preprocessors to load a concept from cache, if an event does not have the default destination then that destination has to be assigned to event before starting test in Rule Input View.

Assert Tester Data

Procedure

1. Select the **Rule Input** tab and then the **Tester Data** tab.
2. One entity can have multiple rows of test data. To select which row or rows of test data to assert, double click the entity's URI in the Select Test Data view. It appears in the Test Data editor. Select one or more check boxes in the Use column as desired. You can repeat this step for all the entities shown in the Select Test Data view, as needed.

You can edit the test data at this time too and you can add test data for more entities. See [Working with Concept and Event Test Data](#) for details.
3. In the Input panel, **Launch Target** field, specify which engine to use.

Multiple engines can run in tester at the same time, for example, a cache agent engine, and an engine running inference agents.
4. In the Input panel, **Rule Session** field, specify which agent to assert the data to.

One engine (processing unit) can have multiple inference agents.
5. Click **Start Test**.

You see console messages and results. See [Viewing the Results](#) for details about understanding the results of a test or debugger run.
6. As appropriate, select more test data to assert (as in [step 2](#)), and again click **Start Test**. Once again, analyze the results of asserting that data using the other views.

To Run Tester or Debugger with Rule Data
7. Open the Rule Input view, if it is not already shown. In the **Window** menu, click **Show View > Other** . Expand **TIBCO BusinessEvents**, and select **Rule Input**. Click **OK**.
8. Perform one of the following steps.
 - Provide input from the mapper as explained in [Using the Function Argument Mapper](#). Optionally you can click **Save** and save the values to an XML file.
 - Click **Load** to load the input values from an existing XML file.
9. Specify the **Launch Target**, **Entity URI**, **Destination URI** and **Rule Session**.
10. Click **Assert**.

Result

See [Viewing the Results](#) for details about understanding the results of a test or debugger run.

Viewing the Results

After data is asserted to the working memory (see [Running Tester](#)), a run to completion (RTC) cycle occurs and you see the following:

- The **Console** tab displays engine console messages.
- The result data editor appears 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](#) 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.

Understanding Result Data

The Result Data view shows the results of a single RTC cycle. On the left of the view you see the **Created**, **Modified**, and **Deleted** entities.

Expanding each entity shows the rule that affected the entity, and also the causal object that triggered the rule.

Click any entity to display its values in the **Result** panel.

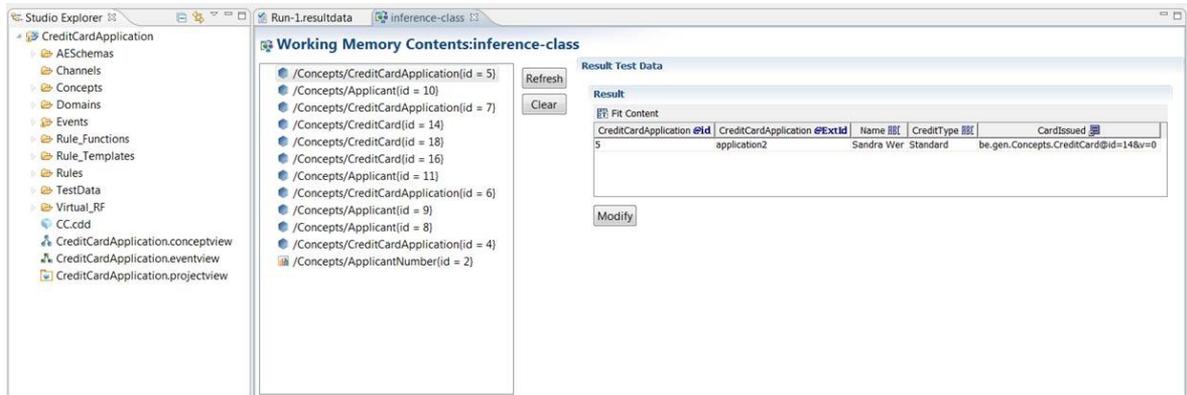


Test results will not be shown for cache applications; they will be shown only for in memory applications.

Viewing and Understanding Working Memory Contents

The Working Memory concepts or scorecards can be manipulated in Working Memory View. Option to modify Working Memory Contents is available only in the Working Memory editor.

Change the property values in Working Memory View and click **Modify**.



Procedure

1. Click the **Working Memory Contents** icon on the tool bar, or from the **Project** menu.
2. Select the appropriate rule session also known as the agent.

The number of objects in the working memory contents is specified in the **Preferences**. For more details, see [Tester Preferences](#).

Unit Testing Using JUnit

JUnit is a unit testing framework for the Java programming language. It is an instance of the xUnit architecture for unit testing frameworks. TIBCO BusinessEvents uses BEUnit (a JUnit based extension) to author unit tests for BusinessEvents applications.

JUnit Use Cases

In BusinessEvents, individual changes in business logic and data could also affect functioning of the BusinessEvents application. Using a unit testing framework, you can better understand the effect of individual changes to the BusinessEvents application.

For example, during migration to a new TIBCO BusinessEvents version, you can use the unit testing framework to test if your old application performs as expected in the new version of the TIBCO BusinessEvents software.

BEUnit (JUnit Extension) Implementation

The BEUnit test framework provides helper classes to author standard JUnit test suites, which you can invoke in the context of a running BE engine. The following classes are the major classes for the BEUnit test framework:

- `com.tibco.cep.runtime.service.tester.beunit.BETestEngine` - This class is used for setting up and configuring the BusinessEvents engine to run in a tester context. The `BETestEngine` class provides methods to assert data into the running test engine, invoke functions, reset the session between tests, and so on.
- `com.tibco.cep.runtime.service.tester.beunit.TestDataHelper` - This class provides helper methods to work with the test data. The existing test data files from previous versions of BusinessEvents can be used in BEUnit tests. You can also create individual concepts and events using the methods available in the `TestDataHelper` class.
- `com.tibco.cep.runtime.service.tester.beunit.Expecter` - This class provides a comprehensive set of methods to introspect the results of a test run.

For more information on these classes, refer to *Java API Reference*.

Unit Test Suite Flow

The general flow of a particular unit test suite is as follows:

1. The BCTestEngine is configured and the test engine is started. This is performed during the set up phase of the test suite.
2. One or more individual unit tests are run against a particular BusinessEvents application as specified in the setup.
3. For each individual unit test, the test creates and asserts some concepts/events.
4. Next, use an Expecter class method to specify a particular outcome. This can include the case such as, determining that a particular rule fired, determining that a concept has been modified, or testing whether an event has been asserted. The Expecter class is used to test the specific behavior of the BusinessEvents application. If the expectation fails (for instance, RuleB fired before RuleA for certain input) the unit test also fails.

You can reset the tester session at any time; however, it's a good practice to do it at the start of an individual test. Resetting the session clears the working memory, and resets any active timers. So, any objects that were asserted in previous tests do not have an effect on the current test.

Adding Unit Test Suite for the BusinessEvents Project

TIBCO BusinessEvents Studio provides a wizard to create a unit test suite for the project using new wizard dialog.



The Object Management type of the project should be set to InMemory as the BEUnit Test Suite runs in the API mode.

Procedure

1. In BusinessEvents Studio explorer, select the BusinessEvents project for which you want to create the BEUnit test suite and in the menu bar, select **File > New > Other**.
2. In the wizard, select **TIBCO BusinessEvents > BEUnit Test Suite** and click **Next**. The New BusinessEvents Unit Test Wizard is displayed.
3. In the New BusinessEvents Unit Test Wizard, enter the values of the following fields and click **Next**.

Field	Description
Parent Folder	<p>Select the Java source folder of the project as the parent folder in the project.</p> <p> The test suite should be placed in an existing Java source folder. If the parent folder is not a Java source folder then after test suite creation, add the folder as Java source folder.</p>
File Name	Name of the unit test suite Java file.
EAR File	The EAR archive for the selected project. The path can be a relative path as well.

Field	Description
CDD File	Select the CDD file of the project. This populates the Processing Unit and Agent Name options. The path can be a relative path as well.
Processing Unit	Processing unit for the project.
TRA File	(Optional) Filepath to the TRA file of the project. The path can be a relative path as well.
Agent Name	(Optional) Agent name for the selected CDD. If there are multiple agents, then the recommendation is to provide agent names separated by commas.
Description	(Optional) Short description for the test suite.
Concept TestData	(Optional) Select the concept test data for the project.
Event TestData	(Optional) Select the event test data for the project.

- In the next page of the wizard, select the test that you want the unit test suite to perform and click **Finish**.
The test suite Java file with name specified in **File Name** is created with code required for the test suite. Edit the file according to your requirement.

Sample Unit Test Suite File

The following is a sample unit test suite .java file generated for the CreditCardApplication project to test if a particular event has been asserted.

```

package;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import java.util.List;
import com.tibco.cep.runtime.model.event.SimpleEvent;
import com.tibco.cep.runtime.service.tester.beunit.BETestEngine;
import com.tibco.cep.runtime.service.tester.beunit.Expecter;
import com.tibco.cep.runtime.service.tester.beunit.TestDataHelper;
import com.tibco.cep.runtime.session.RuleServiceProvider;

/**
 * @description
 */
public class BEUnitTestSuite2 {
    private static BETestEngine engine;
    private static TestDataHelper helper;
    private static Expecter expecter;

    @BeforeClass
    public static void setUpBeforeClass() throws Exception {
        engine = new BETestEngine("C:/tibco/BE54v95/be/5.4/examples/standard/
WebStudio/CreditCardApplication.ear", "C:/tibco/BE54v95/be/5.4/bin/be-
engine.tra",
        "Deployments/CreditCardApplication.cdd", "default",
"inference-class", RuleServiceProvider.MODE_API);

        // Start the test engine
        engine.start();

        // Create a helper to work with test data
        helper = new TestDataHelper(engine);

        // Create an Expecter object to test rule execution, modifications,
assertions, etc.
        expecter = new Expecter(engine);
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        try {
            engine.shutdown();
        } catch (Exception localException) {
        }
    }

    @Before
    public void setUp() throws Exception {
    }

    @After
    public void tearDown() throws Exception {
    }

    /**
     * Test whether a particular Event was asserted by the engine during rule
execution
     */
    @Test
    public void testEventAsserted() throws Exception {
        engine.resetSession(); // (optional) reset the rule session, which
will clear working memory, restart timers, and clear the data from any

```

```
previous tests

    // TODO : Change test data path here to create events to be asserted
from a test data file
    List<SimpleEvent> events = helper.createEventsFromTestData("/TestData/
<test data file name>");
    if (events.size() > 0) {
        engine.assertEvent(events.get(0), false);
    }
    engine.executeRules();
    expecter.expectEventAsserted("/Events/<event name>");
}
}
```

What to do next

Right-click the new BEUnit test suite .java file and select **Run as > JUnit Test** to execute the test suite.

Shared Resources

A set of resources called Shared Resources can be used for various purposes in your projects.



A RuleServiceProvider Configuration shared resource is available In TIBCO BusinessEvents Studio. It can be used during migration to display a migrated version 3.x resource for informational purposes. For integration with TIBCO ActiveMatrix BusinessWorks projects, use the RuleServiceProvider Configuration activity, available in the TIBCO Designer palette of TIBCO BusinessEvents Activities. See [TIBCO ActiveMatrix BusinessWorks 6 Integration](#).

Adding a Shared Resource



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

Procedure

1. In BusinessEvents Studio Explorer, right-click the folder where you want to store the shared resource and select **New > Other** .
2. In the Select a Wizard dialog, expand TIBCO Shared Resources and select the resource type you want to add (see list in [step 3](#)) and click **Next**.
3. In the new resource wizard enter a name in the **File Name** field and click **Finish**.

You see the editor for the shared resource you selected. See the following sections for guidelines on completing the fields:

- [ActiveSpaces Connection Reference](#)
 - [Hawk Connection Reference](#)
 - [HTTP Connection Reference](#)
 - [Identity Resource Reference](#)
 - [JDBC Connection Reference](#), and [Enabling the Test Connection Feature](#)
 - [JMS Application Properties](#)
 - [JMS Connection Reference](#), and [Enabling the Test Connection Feature](#)
 - [JNDI Configuration Reference](#)
 - [Rendezvous Transport Reference](#)
 - [StreamBase Connection Reference](#)
4. (Optional) View the **Source** tab. The source tab shows the XML source format for the shared resource. You can edit the XML source directly, but this is not recommended because of the risk of error. The resource editor provides validation checks.

Enabling the Test Connection Feature

To make the Test Connection feature work for JMS Connection and JDBC Shared Connection shared resources, copy the relevant JAR files to the following directory:

```
BE_HOME/lib/ext/tpcl
```

(This directory is referenced in the extended classpath in the file BE_HOME/studio/eclipse/configuration/studio.tra.)



If TIBCO BusinessEvents Studio is running when you copy the file or files, you must restart it for the test connection feature to work.

For TIBCO Enterprise Message Service copy the `jms-2.0.jar` and `tibjms.jar` to the above location.

For WebSphere MQ, copy the MQ JAR files and the binding file to the above location.

For DBMS products, copy the supported driver file you are using for the DBMS product.

ActiveSpaces Connection Reference

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](#).

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements . Unlike other resource identifiers shared resource identifiers can have spaces in the name. The name then appears in the title of the resource.
Configuration		
Description	No	Short description of the resource.
Metaspace Name	Yes	Specifies a 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'.
MemberName	Yes	Name the AS channel will take when connecting to ActiveSpaces metaspace

Field	Global Var?	Description
Discovery URL	Yes	<p>Specifies how a metaspace instance discovers the current metaspace members. Multicast discovery can use either PGM - Pragmatic General Multicast or RV - TIBCO Rendezvous protocol.</p> <p>If using PGM protocol, the multicast URL is expressed in the following format:</p> <pre>tibpgm://destination port/interface;discovery group address/optional transport arguments, where</pre> <ul style="list-style-type: none"> <i>destination port</i> specifies the destination port used by the PGM transport. If not specified, the default value of 7888 is used. <i>interface;discovery group address</i> specifies the address of the interface to be used for sending discovery packets, and the discovery group address to be used. If not specified, it defaults to the default interface and discovery address, 0.0.0.0;239.8.8.8. <i>optional transport arguments</i> specifies a semicolon-separated list of optional PGM transport arguments. By default, the PGM transport is tuned to provide the best performance according to the most common deployment architectures. The values of those optional arguments should be changed only when necessary and with care, since any inappropriate values could easily result in degraded performance of the product. <p>If using TIBCO Rendezvous, the multicast URL is expressed in the following format:</p> <pre>tibrv://service/network/daemon, where</pre> <ul style="list-style-type: none"> <i>service</i> specifies the TIBCO Rendezvous service number (UDP port) that will be used. If not specified, the value defaults to 7889. <i>network</i> specifies the interface and discovery group address that will be used to send Rendezvous discovery packets. The format is: <code>interface; discovery_group_address</code> If not specified, the default interface and discovery group address will be used. (<code>tibrv://7889/;239.8.8.9/</code>). <i>daemon</i> specifies where to find the Rendezvous daemon. If not specified, it will try to connect to a local daemon on port 7500.
ListenUrl	Yes	<p>The discovery mechanism is based on pure TCP.</p> <p>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.</p> <p>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).</p>
RemoteListenUrl	Yes	<p>This field is used to configure TIBCO ActiveSpaces Channel as a remote-discovery proxy. In this case, any remote client can connect to an ActiveSpaces metaspace via the ActiveSpaces Channel node.</p>

Field	Global Var?	Description
EnableSecurity	Yes	<p>Enables security for the ActiveSpaces connection resource when selected.</p> <p>Note: Some fields are activated only for the specific security role or authorization policy.</p>
SecurityRole	Yes	<p>Security role of a node for the secure ActiveSpaces channel in the metaspace. The values are:</p> <ul style="list-style-type: none"> • <code>Controller</code> is dedicated to enforcing a security domain's defined security behavior for a metaspace associated with the security domain. Security domain controllers are the only discovery nodes in a metaspace. • <code>Requester</code> just requires access to the data in the data grid, such as a seeder or a leech, and which need to be authorized by a controller. Requesters can never be used as discovery nodes. <p>The controller nodes are configured with a security policy file. The requester nodes provide a token file and additional credentials to the controller for authentication. The controller performs authentication as defined in its policy file and using the credentials provided by the requester.</p> <p>If the <code>Controller</code> option is selected, then the following fields become active:</p> <ul style="list-style-type: none"> • Identity Password • PolicyFile <p>If the <code>Requester</code> option is selected, then the following fields become active:</p> <ul style="list-style-type: none"> • Identity Password • TokenFile • Credential
Identity Password	Yes	The password for the identity key in the security policy file.
PolicyFile	Yes	Absolute path to the policy file which contains the security settings that the controller node enforces. It is generated using the <code>as-admin</code> utility.
TokenFile	Yes	Absolute path to the token file which is used by requester to connect to a metaspace whose security is defined in the policy file.

Field	Global Var?	Description
Credential	Yes	<p>Authentication policy to be used for authentication as specified in the policy file. The values are:</p> <ul style="list-style-type: none"> • USERPWD - A user name and password based authentication is used. It activates the following fields: <ul style="list-style-type: none"> – Domain – Username – Password • X509V3 - The authentication source is an LDAP configured with certificate based authentication. It activates the following fields: <ul style="list-style-type: none"> – KeyFile – PrivateKey
Domain	Yes	Domain name for system based user authentication.
Username	Yes	User name for LDAP and system based authentication.
Password	Yes	Password for LDAP and system based authentication.
KeyFile	Yes	The absolute path for a file containing the key to use for LDAP with the certificate based authentication.
PrivateKey	Yes	The password for the identity key in the LDAP identity file specified in KeyFile .

Hawk Connection Reference

The Hawk Connection resource describes the connection to a TIBCO Hawk domain through a specific transport. This section provides a reference to the fields. For procedures see [Adding a Shared Resource](#).

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	<p>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 .</p> <p>Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.</p> <p>The name then appears in the title of the resource.</p>

Field	Global Var?	Description
Configuration		
Description	No	Short description of the resource.
Transport	Yes	Type of transport used by the Hawk domain: Rendezvous or EMS.
Domain	Yes	Name of the Hawk domain.
Transport: Rendezvous		
RV_Service	Yes	Specifies the User Datagram Protocol (UDP) service group used by the TIBCO Rendezvous daemon for session communications. A service can be specified either by its name or its port number. The default configuration uses the service port number 7474.
RV_Network	Yes	Specifies the network to be used for outbound session communications when a computer is connected to more than one network. A network can be specified by its name or by its IP address.
RV_Daemon	Yes	Specifies the TIBCO Rendezvous daemon that will handle communication for the session. A local daemon is specified by the communications type (always tcp) and a socket number (e.g., 7474). The default configuration uses the local daemon with the TCP socket number 7474. Specify a remote daemon by inserting its host name or IP address between the tcp entry and the port number of the daemon parameter, for example: <code>tcp:remote_computer:7800</code>
Transport: EMS		
EMS_ServerURL	Yes	Specifies the EMS server to connect to. The server URL is typically provided in the following format: <code>protocol://hostname:port-number</code> For example, <code>tcp://myhost:7222</code>
EMS_UserName	Yes	Specifies the user name used to connect to the EMS server.
EMS_Password	Yes	Specifies the password for the user name used to connect to the EMS server.

HTTP Connection Reference

The HTTP Connection resource describes the characteristics of the connection used to receive incoming HTTP requests. This section provides a reference to the fields. For procedures see [Adding a Shared Resource](#).

The HTTP Connection resource can specify that the HTTPS (secure sockets layer or SSL) protocol must be used by clients. If this is enabled, you can configure the SSL parameters for the HTTP server using the Configure SSL Button. See [SSL Configuration](#) for more information.



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	<p>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 .</p> <p>Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.</p> <p>This field appears on the New HTTP Connection Wizard. The name then appears in the title of the resource.</p>
Configuration		
Description	Yes	Short description of the resource.
Host	Yes	<p>Specifies the name of the host that accepts the incoming requests. For machines that have only one network card, the default value <code>localhost</code> 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.</p> <p>Note</p> <p>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.</p> <p>Note</p> <p>If there is more than one network card on the machine, and you specify <code>localhost</code> in this field, then all network cards on the machine will listen for incoming HTTP requests on the specified port.</p>
Port	Yes	Port number on which to listen for incoming HTTP requests.

Field	Global Var?	Description
Use SSL	No	<p>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.</p> <p>By enabling this field you can specify SSL parameters with the Configure SSL button (see Configure SSL).</p>

SSL Configuration

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

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



The HTTPComponent server type does not support Entrust based SSL.

Field	Description
Requires Client Authentication	<p>Selecting this field requires clients to present their digital certificate before connecting to the HTTP server.</p> <p>When this field is selected, the Trusted Certificates Folder becomes enabled so that you can specify a location containing the list of trusted certificate authorities.</p>
Trusted Certificates Folder	<p>This field is only applicable when the Requires Client Authentication field is selected.</p> <p>This field specifies a folder in the project containing one or more certificates from trusted certificate authorities. This folder is selected when a client connects to ensure that the client is trusted. This prevents connections from rogue clients.</p>
Identity	<p>This is an Identity resource that contains the HTTP server's digital certificate and private key. See Identity Resource Reference for more information.</p>
Strong Cipher Suites Only	<p>When selected, this field specifies that the minimum strength of the cipher suites used can be specified with the <code>bw.plugin.security.strongcipher.minstrength</code> 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.</p> <p>When this field is not selected, only cipher suites with an effective key length of up to 128 bits can be used.</p>

Identity Resource Reference

The Identity resource encapsulates information that may be used to authorize a user, connection, and so forth. The information you supply changes depending on the type of Identity resource you want to use. This section provides a reference to the fields. For procedures see [Adding a Shared Resource](#).

The identity certificate location, its type, and password can be specified as global variables.

Wizard and Configuration Tab

The New Identity Resource Wizard and the Configuration tab of the Identity Resource have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	<p>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.</p> <p>Unlike other resource identifiers shared resource identifiers can have spaces in the name.</p> <p>This field appears in the New Identity Resource Wizard. The name then appears in the title of the resource.</p>
General		
Description	Yes	Short description of the resource.
Type	No	The type of identity resource: Identify File, Certificate/Private Key, or Username/Password (the default). See sections below for details.
Identity File		
Use this option if the certificate includes the private key information in the same file.		
URL	Yes	Location of the certificate (which includes the private key).
File Type	No	<p>Choose the certificate file type from the drop-down list:</p> <ul style="list-style-type: none"> Entrust JCEKS JKS PEM PKCS12
Password	Yes	Password for the certificate.

Field	Global Var?	Description
Certificate/Private Key Identity Use this option if the private key and the certificate are in two separate files.		
Certificate URL	Yes	Location of the certificate. Click the browse icon or type in a URL.
Key URL	Yes	Location of the private key file associated with the certificate.
Key Password	Yes	Password used for private key.
Username/Password Use this option if you want to use a user name and password for authentication and do not want to use a certificate.		
Username	Yes	Name of the user for this identity.
Password	Yes	Password for the user for this identity.

JDBC Connection Reference

The JDBC Connection resource describes a JDBC connection. JDBC connections are used with backing stores and with database concepts (available in the TIBCO BusinessEvents Data Modeling add-on).

For procedures to add a JDBC shared resource, see [Adding a Shared Resource](#).

For using Secure Sockets Layer protocol for JDBC connection between BusinessEvents and a database server, see [Configuring SSL for JDBC Connection](#).

JDBC Connection Wizard and Configuration Tab

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements . Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.
Configuration		
Description	Yes	Short description of the resource.

Field	Global Var?	Description
Connection Type	Yes	<p>Specifies the kind of JDBC connection you wish to create. The connection type can be one of the following:</p> <ul style="list-style-type: none"> • JDBC • JNDI <p>The type of connection determines the other configuration fields that appear.</p>
<p>JDBC Connection Type Configuration Fields</p> <p>Tip: Using global variables makes the project more portable.</p>		
JDBC Driver	Yes	<p>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:</p> <pre>oracle.jdbc.OracleDriver (thin)</pre> <pre>com.ibm.db2.jcc.DB2Driver (supported for database concepts only)</pre> <pre>com.microsoft.sqlserver.jdbc.SQLServerDriver</pre> <p>When you select a driver, the Database URL field is populated with a template for the URL for the driver.</p>
Database URL	Yes	<p>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.</p> <p>You can configure Single Client Access Name (SCAN) for Oracle database in a cluster with numerous nodes.</p> <p>For more information on configuring SCAN, see <i>TIBCO BusinessEvents Configuration Guide</i>.</p>
Maximum Connections	Yes	<p>The maximum number of database connections to allocate. The default maximum is 10. The minimum value that can be specified is 1.</p> <p>See Connection Pooling for more details, including related settings that override this setting.</p>
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	<p>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.</p>

Global		
Field	Var?	Description
JNDI Connection Type Configuration Fields		
JNDI DataSource Name	Yes	The JNDI name specified for the DataSource.
Use Shared JNDI Configuration	No	When this field is selected, the JNDI Configuration field appears, allowing you to choose a shared JNDI Configuration resource. When this check box is not selected, configuration fields appear.
JNDI Configuration	No	This field only appears when the Use Shared JNDI Configuration field is selected. This field allows you to choose a JNDI Configuration shared resource that specifies the JNDI connection information.
JNDI Context Factory	No	The initial context factory class for accessing JNDI. (<code>javax.naming.Context.INITIAL_CONTEXT_FACTORY</code>). 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 (<code>javax.naming.Context.PROVIDER_URL</code>). 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 (<code>javax.naming.Context.SECURITY_PRINCIPAL</code>). 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 (<code>javax.naming.Context.SECURITY_CREDENTIALS</code>). If the JNDI provider does not require access control, this field can be empty.
Use SSL	Yes	Specifies whether you want to use secure sockets layer (SSL) protocol for connection to the database server. This protocol authenticates the server to the client, and optionally authenticates the client to the server. Select this field to specify SSL parameters using the Configure SSL button (see JDBC SSL Configuration Reference).

JDBC SSL Configuration Reference

Enter SSL parameters for a secure JDBC connection. Click **Configure SSL** in JDBC Configuration to display the SSL Configuration for JDBC page.

SSL Configuration for JDBC

Field	Global Variable?	Description
Requires Client Authentication	Yes	<p>Selecting this field requires clients to present their digital certificate before connecting to the database server.</p> <p>The Identity field becomes active when this field is selected, so that you can specify the identity file.</p>
Trusted Certificates Folder	Yes	<p>Location of the trusted certificates on client machine. The trusted certificates are a collection of certificates from servers with whom you establish connections. If the server you wish to establish a connection with, presents a certificate that does not match one of your trusted certificates, the connection is refused.</p> <p>This prevents connections to unauthorized servers.</p> <p>Trusted certificates must be imported into a folder, and then you can select the folder in this field.</p>
Identity	No	<p>The location of the identity shared resource file that contains the information to authenticate BusinessEvents client identity.</p> <p>Select the Required Client Authentication field to activate this field.</p> <p>See Identity Resource Reference for more information.</p>
Trust Store Password	Yes	Specifies the password for the truststore
Verify Host Name	Yes	Specifies whether to verify the host name

Field	Global Variable?	Description
Expected Host Name	Yes	Specifies the host name for verification Select the Verify Host Name field to activate this field.

Configuring SSL for JDBC Connection

Using the Secure Sockets Layer (SSL) protocol, you can establish secure communication between the JDBC client and DBMS servers.

Refer *TIBCO BusinessEvents Configuration Guide* for more information on JDBC backing store and database connections.

Prerequisites

- Configure database server for SSL. Refer to the respective DBMS documentation for server-side configuration steps.
- Copy the appropriate JDBC drivers file to `BE_HOME/lib/ext/tpcl`.

Procedure

1. In TIBCO BusinessEvents Studio, open the JDBC shared resource for editing.
See [JDBC Connection Wizard and Configuration Tab](#) for more details on JDBC fields.
2. Select the **Use SSL** check box for activating the SSL protocol for the JDBC connection. The **Configure SSL** button is activated.
3. Click **Configure SSL**.
The SSL Configuration for JDBC window is displayed.
4. Enter values for the SSL parameters and click **OK**.
See [JDBC SSL Configuration Reference](#) for more details on these parameters.
5. Save the JDBC shared resource.
The JDBC connection is now configured to use the SSL protocol.

Oracle Database Configuration

6. (Oracle database only) When using the Oracle wallet or .p12 file as keystore or truststore type, copy the following JAR files from the `jlib` directory in the Oracle installation to `BE_HOME/lib/ext/tpcl`:
 - `oraclepki.jar`
 - `osdt_cert.jar`
 - `osdt_core.jar`

If you get the following exception when using JKS as keystore or truststore type:

```
java.security.cert.CertPathValidatorException: Algorithm constraints
check failed: MD5withRSA
```

then perform the following steps:

Oracle 11g

- Remove MD5 from the following property value in `java.security` file:

```
jdk.certpath.disabledAlgorithms=MD2,MD5,RSA
keySize < 1024
```

- Remove MD5withRSA from the following property value in `java.security` file:

```
jdk.tls.disabledAlgorithms=SSLv3, RC4,
MD5withRSA, DH keySize < 768
```

- Store root certificate in the truststore folder.

Oracle 12c

Create wallets and certificates with SHA-256 or other than MD5withRSA as signing algorithm. Now create the JKS keystore and truststore using these wallets.



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.

Test Connection Button

Using the **Test Connection** button you can test the connection specified in the configuration of this resource. See [Enabling the Test Connection Feature](#) for a step you must take to enable the connection to work.

JMS Application Properties

The JMS Application Properties resource describes any JMS message properties that a JMS application expects. These properties can then be added.

For procedures see [Adding a Shared Resource](#).

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		

Field	Global Var?	Description
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements . Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.
Description	Yes	Short description of the resource.
Configuration		
<p>Properties Table</p> <p>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.</p>		
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 Reference

JMS Connection resource describes a JMS connection. This section provides a reference to the fields. For procedures, see [Adding a Shared Resource](#).

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
File Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifier Naming Requirements . Unlike other resource identifiers shared resource identifiers can have spaces in the name. This field appears on the New JMS Connection Wizard. The name then appears in the title of the resource.
Configuration Tab		

Field	Global Var?	Description
Description	Yes	Short description of the resource.
User Name	Yes	<p>User name to use when logging into the JMS server.</p> <p>If the JMS provider does not require access control, this field can be empty.</p> <p>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.</p>
Password	Yes	<p>Password to use when logging into the JMS server.</p> <p>If the JMS provider does not require access control, this field can be empty.</p>
Auto-Generate Client ID	No	<p>Selecting this field specifies you wish to automatically generate the client ID if no client ID is specified or if the specified ID is already in use. When this field is selected, if a value is specified 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.</p> <p>If this field is not selected, then the value specified in the Client ID field is used. If no value is specified in the Client ID field, then no client ID is set.</p>
Client ID	Yes	<p>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.</p> <p>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.</p>
Use SSL	No	<p>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.</p> <p>Note: SSL is supported only when using TIBCO Enterprise Message Service.</p> <p>See Configure SSL for more information.</p>
Use JNDI for Connection Factory	Yes	<p>Specifies whether JNDI should be used to look up the ConnectionFactory object. If this field is not selected, the Provider URL and Use XA Connection Factory fields appear. If this field is selected, JNDI configuration fields appear.</p>
Provider URL	Yes	<p>This field is only available when the Use JNDI for Connection Factory field is not selected.</p> <p>This is the URL to use to connect to the JMS server.</p>

Field	Global Var?	Description
Use XA Connection Factory	No	<p>This field is only available when the Use JNDI for Connection Factory field is not selected.</p> <p>When selected, this field specifies that an XA connection factory is to be used.</p>
Connection Factory SSL Password	Yes	<p>This field is only available when the Use SSL check box is selected, and the User Shared JNDI Configuration check box is not selected.</p> <p>The SSL configuration is specified in the ConnectionFactory object, except for the client SSL password.</p> <p>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.</p>
Use Shared JNDI Configuration	Yes	<p>When this field is selected, the JNDI Configuration field appears. Using this you can choose a JNDI Configuration Reference .</p> <p>When this field is not selected, JNDI configuration fields appear.</p>
JNDI Configuration	No	<p>This field only appears when the Use Shared JNDI Configuration field is selected.</p> <p>This field allows you to choose a JNDI Configuration Reference that specifies the JNDI connection information.</p>
JNDI Context Factory	Yes	<p>This field only appears when the Use Shared JNDI Configuration field is not selected. Required. The initial context factory class for accessing JNDI. (<code>javax.naming.Context.INITIAL_CONTEXT_FACTORY</code>). You can choose from the drop down list of supported classes or you can type in a different InitialContextFactory class name.</p> <p>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.</p>
JNDI Context URL	Yes	<p>This field only appears when the Use Shared JNDI Configuration field is not selected. Required. This is the URL to the JNDI service provider (<code>javax.naming.Context.PROVIDER_URL</code>). An example URL is provided when one of the supported JNDI context factory classes is selected.</p> <p>See your JNDI provider documentation for the syntax of the URL.</p>
JNDI User Name	Yes	<p>This field only appears when the Use Shared JNDI Configuration field is not selected. User name to use when logging into the JNDI server (<code>javax.naming.Context.SECURITY_PRINCIPAL</code>). If the JNDI provider does not require access control, this field can be empty.</p>
JNDI Password	Yes	<p>This field only appears when the Use Shared JNDI Configuration field is not selected. Password to use when logging into the JNDI server (<code>javax.naming.Context.SECURITY_CREDENTIALS</code>). If the JNDI provider does not require access control, this field can be empty.</p>

Test Connection Button

Using the **Test Connection** button you can test the connection specified in the configuration of this resource. See [Enabling the Test Connection Feature](#) for a step you must take to enable the connection to work.

When JNDI is used (that is, when the Use JNDI for Connection Factory check box is selected), the Test Connection button works only when the JNDI-related fields on the **Configuration** and **Advanced** tab are correctly specified.

Advanced Tab

The **Advanced** tab has the following fields.

Field	Global Var?	Description
Topic Connection Factory	Yes	<p>This field is only available when the Use JNDI for Connection Factory field on the Configuration tab is selected.</p> <p>The <code>TopicConnectionFactory</code> object stored in JNDI. This object is used to create a topic connection with a JMS application.</p> <p>See your JNDI provider documentation for more information about creating and storing <code>TopicConnectionFactory</code> objects.</p>
Queue Connection Factory	Yes	<p>This field is only available when the Use JNDI for Connection Factory field on the Configuration tab is selected</p> <p>The <code>QueueConnectionFactory</code> object stored in JNDI. This object is used to create a queue connection with a JMS application.</p> <p>See your JNDI provider documentation for more information about creating and storing <code>QueueConnectionFactory</code> objects.</p>
Optional JNDI Properties	No	<p>Any additional properties to supply for the connection. You specify a name, datatype, and value for each property.</p> <p>These properties are typically vendor-specific. See your JNDI provider documentation for more information about the available properties.</p>

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	<p>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.</p> <p>This prevents connections to unauthorized servers.</p> <p>Trusted certificates must be imported into a folder, and then you can select the folder in this field.</p>
Identity	<p>The location of the client certificate.</p> <p>You only need to specify the client certificate when the JMS server requires client authentication.</p> <p>See Identity Resource Reference for more information.</p>
Trace	<p>Specifies whether SSL tracing should be enabled during the connection. If selected, the SSL connection messages are logged and sent to the console.</p>
Debug Trace	<p>Specifies whether SSL debug tracing should be enabled during the connection. Debug tracing provides more detailed messages than standard tracing.</p>
Verify Host Name	<p>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.</p> <p>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:</p> <pre>com.tibco.tibjms.TibjmsSSL.setVerifyHost(false)</pre>
Expected Host Name	<p>Specifies the name of the host you are expecting to connect to. This field is only relevant if the Verify Host Name field is also selected</p> <p>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.</p> <p>This prevents hosts from attempting to impersonate the host you are expecting to connect to.</p>

Field	Description
Strong Cipher Suites Only	<p>When selected, this field specifies that the minimum strength of the cipher suites used can be specified with the <code>bw.plugin.security.strongcipher.minstrength</code> 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.</p> <p>When this field is not selected, only cipher suites with an effective key length of up to 128 bits can be used.</p>

JNDI Configuration Reference

The JNDI Configuration shared resource provides a way to specify JNDI connection information that can be shared by other resources. This resource can be specified in any resource that permits JNDI connections. For example, [JDBC Connection Reference](#) can use JNDI connections. This section provides a reference to the fields. For procedures see [Adding a Shared Resource](#).

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
Name	No	<p>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 .</p> <p>Unlike other resource identifiers shared resource identifiers can have spaces in the name.</p> <p>This field appears on the New JNDI Connection Wizard. The name then appears in the title of the resource.</p>
Configuration		
Description	Yes	Short description of the resource.
JNDI Context Factory	Yes	<p>The initial context factory class for accessing JNDI. (<code>javax.naming.Context.INITIAL_CONTEXT_FACTORY</code>). You can choose from the drop down list of supported classes.</p> <p>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.</p>

Field	Global Var?	Description
JNDI Context URL	Yes	The URL to the JNDI service provider (<code>javax.naming.Context.PROVIDER_URL</code>). 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 (<code>javax.naming.Context.SECURITY_PRINCIPAL</code>). If the JNDI provider does not require access control, this field can be empty.
JNDI Password	Yes	Password for logging into the JNDI server (<code>javax.naming.Context.SECURITY_CREDENTIALS</code>). If the JNDI provider does not require access control, this field can be empty.

Advanced Section

The Advanced section has the following fields.

Field	Global Var?	Description
Validate JNDI Security Context	No	Some application servers store the security context on the thread used to establish the JNDI connection (at the time of this release, only the WebLogic application server does this). In that case, the first activity to use this resource establishes the security context, then subsequent activities use the same security context, unless this field is selected. Selecting this field ensures that each activity that uses this resource examines the security context to determine if the activity uses the same security context as the security context established on the thread. If they are different, the activity's configured security context is used. Selecting this field causes additional overhead for activities that use this resource. Therefore, only select this field when necessary.
Optional JNDI Properties	No	The table in this field contains optional properties to pass to the JNDI server. Use the +, X, and arrow keys to add, delete, and move properties in the list. Each property requires the property name, the datatype for the property, and the value for the property. See the documentation for your JNDI provider for more information about properties that can be passed to the JNDI server.

Rendezvous Transport Reference

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](#).

Wizard and Configuration Tab

The Wizard and Configuration tab have the following fields.

Field	Global Var?	Description
Wizard		
Name	No	<p>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 .</p> <p>Unlike other resource identifiers, however, shared resource identifiers can have spaces in the name.</p> <p>This field appears on the New Rendezvous Transport Wizard. The name then appears in the title of the resource.</p>
Configuration		
Description	Yes	Short description of the resource.
Daemon	Yes	<p>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.</p> <p>For example:</p> <pre>ssl:acct:5785</pre>
Network	Yes	<p>This field contains the host name, IP address, network name, or interface name.</p> <p>For example:</p> <pre>;224.34.103.4</pre>
Service	Yes	The TIBCO Rendezvous service name, specified in one of the following formats: <i>service name</i> or <i>port number</i> .
Use SSL	No	The Use SSL check box specifies that Secure Sockets Layer (SSL) should be used when communicating with the TIBCO Rendezvous daemon. When this field is selected, the Configure SSL button is enabled. See Configure SSL for more information about configuring SSL parameters.

Configure SSL

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	<p>File containing one or more certificates from trusted certificate authorities. This file is selected 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.</p> <p>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.</p>
Identity	<p>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.</p> <p>See Identity Resource Reference for more information.</p>

Advanced Section

The Advanced section of the UI has the following fields.

Field	Global Var?	Description
RV Type	Yes	<p>The type of TIBCO Rendezvous connection to use. This can be reliable (standard RV transport), certified (RVCM), or Distributed Queue (RVCMQ).</p> <p>The fields of the Advanced tab correspond to the value selected for this field.</p>
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.
Require Old Message	Yes	Select this box if you wish to require the retention of messages for which delivery has not been confirmed. These messages will be resent.

Field	Global Var?	Description
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.
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.

StreamBase Connection Reference

The StreamBase and BusinessEvents connection details are configured in the shared resource.

For procedures see [Adding a Shared Resource](#).

Field	Global Var?	Description
Description	No	Short description of the resource

Field	Global Var?	Description
StreamBase Server URI	Yes	<p>The URI at which TIBCO BusinessEvents can contact the StreamBase server</p> <p>Example: sb://localhost:10000/</p>
UserName	Yes	<p>A valid user name for the StreamBase server.</p> <p>Note: This field is required only when the basic authentication is enabled at the StreamBase server.</p>
Password	Yes	<p>The password assigned to the user name specified in UserName, for accessing the StreamBase server.</p> <p>Note: This field is required only when the basic authentication is enabled at the StreamBase server.</p>

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.

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.

Set Up a Domain Model

Setting up domain models has two steps:

Procedure

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](#) for details.

You can also import (and export) domain models. See [Importing and Exporting Domain Models](#).

2. Associate domain models with properties. See [Associating Domain Models with Properties](#) for details.

Result

When creating a decision table, domain models control the values you can use for a given property.

Domain Model Value Descriptions for User Friendly Presentation

All domain model values can have optional descriptions that appear in the domain model editor. A preference determines whether domain model values or their descriptions appear in decision table cells. For some applications, displaying descriptions can make the table easier for users to understand. For example, suppose the value is a code such as 23, and the description is North West. Users 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](#)

Add a Domain Model

Procedure

1. Right-click the folder where you want to store the domain model, and select **New > Domain Model** . You see the New Domain Model Wizard.
Alternatively, right-click a resource, and select **New > Other** . In the New dialog, select **Domain Model** under TIBCO BusinessEvents.
2. In the **Domain Model Name** field, type a name for the domain model. In the **Description** field, type a description.



You cannot change the name in the editor. To change the name of any project element, right-click the element in BusinessEvents Studio Explorer and select **Refactor > Rename** . See [Element Refactoring Operations](#) for more details.

3. Click **Finish**. You see the Domain Model editor.
4. In the **Domain Type** field, select the data type for this domain model. See [Supported Data Types](#) for supported types.
5. As needed, complete the **Inherits From** field. If this domain model extends another domain model, browse to and select that domain model.
6. In the Domain Entries section, click **Add**.
An empty row appears in the table of entries.
You can also select rows and click **Duplicate** to duplicate (and then modify) selected rows.
You can select a row and click **Remove** to remove individual rows that are not needed.
7. For each row you add, enter a description (optional) and in the Details section, define the domain model entry.
The Details section presents appropriate fields for defining the type of domain model entry you selected in [step 4](#).
8. When you have created the entries for the domain model, save the domain model resource.

Adding Domain Entries

When you add a domain model, first select its data type. The Domain Model editor then displays an appropriate user interface for defining domain model entries of that data type.

The Domain Entries section is the same for all types. The Details section changes depending on what data type you select for the domain model.

Supported Data Types

Domain Models support the following data types:

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

Sections below show the user interface for each type.

String

String entries are simple text strings.



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.

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.

Boolean

Boolean entry values are always `true` or `false`. The description can give the meaning of the pair of choices, such as male or female, supported or unsupported, eligible or ineligible and so on.

DateTime

In a single domain model, you can enter single values, range values, or a mixture of both. Using a date and time picker, you can specify a date and time.

If you do not want to specify a time of day, set the time to midnight (12:00 AM) for the start date, and to a minute before midnight (11:59:59 PM) for the end date of an inclusive range, or for a single date.

The calendar shows a 12-hour clock. PM numbers are converted to a 24 hour clock format in the value table.

Importing and Exporting Domain Models

You can import domain model information from a database, from a Microsoft Excel spreadsheet, and from the source of a database concept property. (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](#)

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*.

Exporting 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.

Importing Domain Model Entries from Excel

Procedure

1. In BusinessEvents Studio Explorer, perform one of the following actions.
 - Right-click the folder where you want to create the domain model and select **Import > TIBCO BusinessEvents > Domain Model**.
 - In 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.

Export Domain Values to Excel

Procedure

1. In BusinessEvents Studio Explorer, perform one of the following actions.
 - Right-click the domain you want to export and select **Export > TIBCO BusinessEvents > Export Domain to Excel**.
 - In 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**.

Importing Domain Model Entries from a Database Table

When you import domain model information from a database, the result set from the SQL query is transformed for use as domain model entries. This feature is supported with Oracle Database.

If imported from a database concept, database table, or a database query, you can update the domain model by using the **Reload Domain Model** button available on the top bar of the Domain Model editor.

Procedure

1. Add a JDBC Connection resource and configure it to connect to the database from which you want to import the domain model. See [JDBC Connection Reference](#) for details.
2. In BusinessEvents Studio Explorer, perform one of the following actions.
 - Right-click the folder where you want to create the domain model and select **Import > TIBCO BusinessEvents > Domain Model**.
 - In 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. Perform one of the following actions.
 - Select one or more columns, then click the **Create Domain for Selected Columns** button. Values of all columns are used for the domain model entries.
 - Click **Advanced** and enter an SQL query whose 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.

Associate Domain Models with a Property

Procedure

1. Perform one of the following actions.
 - 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.

Mismatching Range Values Not Allowed

If you accidentally enter mismatching values in the Lower and Upper fields for range values, the Problems view displays helpful information.

Display Models

Display Models provides a mechanism to define the display text for elements in WebStudio, such as, properties. Instead of displaying the default element name, a more user readable name in any selected language could be displayed.

You can provide a name for the artifact and its properties in your preferred language using Display Model. Thus, you can see the artifact and its properties name in your language while using the WebStudio. See *TIBCO BusinessEvents WebStudio User's Guide* for how it would appear in the WebStudio.

Display Models are defined inside the TIBCO BusinessEvents Studio, similar to Concepts, Events, and Rules. A Display Model is a specialized type of property file, and follows the familiar name/value pair syntax for property files in Java. Display Models are the first class citizens inside the BusinessEvents Studio project, and their file extension is `.DISPLAY`.

Naming

BusinessEvent assigns the Display Model name that matches with the associated BusinessEvents project artifact (Concept or Event). For instance, for the Concept named `Person.concept`, the corresponding Display Model artifact is named `Person.display`. This naming method simplifies the syntax and allow for a loose coupling of the artifacts. However, the Display Model follows the standard Java I18N naming conventions if language and country values are specified, that is the name of the file is `<artifact_name>_<language_code>_<country_code>.display`. For example, if English is selected as language and United States is selected as country, for the Concept named `Person.display`, the Display Model file is saved with the file name `Person_en_US.display`

The Display Model also supports the code refactoring. If artifact is renamed then the Display Model file associated with the artifact is automatically renamed. If the properties of the artifact is renamed then the properties are also updated in the Display Model.

Syntax

The syntax for the Display Model files follows the standard Java property file syntax, that is, specifying name and value pairs. There are three major parts of the file:

- For each artifact property, there is an optional text that is displayed in the WebStudio UI. The simple syntax is as follows:

```
<property_name>=<display_text>
```

For example,

```
Name.displayText=User Name
```



In the case of Arabic characters, do not use accents, as they are not supported in TIBCO BusinessEvents.

- Additionally, there is an optional property that allows properties to be hidden entirely from the WebStudio UI, so that they are not accessible from the builder based Business Rule editor or Decision Table editor.

```
<property_name>.hidden=true | false
```

For example,

```
Name.hidden=true
```

- Finally, there is an optional property to specify the text to be displayed wherever the artifact itself is shown in the WebStudio UI, for instance inside of the Group Contents window. The syntax for this setting is:

```
displayText=<display_text>
```

For example, for an **Applicant** concept

```
displayText=Credit Card Applicant
```

Overriding Behaviors

As Concepts and Events can inherit properties from parent Concepts and Events, any visible property from the artifact's hierarchy is also visible to the Display Model. For example, if the Concept **Employee** inherits a property called **Name** from the Concept **Person**, it is valid to specify the display text for the **Name** property inside of the Display Model for the **Employee** Concept. The **Name** property can have different meaning for different concepts, and you can use display text to differentiate them.

You can also override a display text inside a Rule Template. This overriding behavior provides another level of meaningful, context-aware specification of properties. A new `display` section could be added to the Rule Template source, and the context-sensitive display text for each accessible property can be specified using the dot notation. For example,

```
display {
    account.AccountID = "Personal Checking Account ID";
}
```

Creating a Display Model

A Display Model provides the more readable name to the artifact properties and you can create it using the TIBCO BusinessEvents Studio.

Procedure

1. Right click on any folder in the project and select **New > Other**. In the New dialog select **TIBCO BusinessEvents > Display Model**. The New Display Model Wizard window is displayed.
2. Enter the values of the following fields in the wizard and click **Finish**.

Field	Description
Project	Name of the project
Target Entity	Project path of the artifact (concept or event) for which the display text is created. Click Browse to browse and select the artifact of the project.
Language	(Optional) Language for the display model. If none is selected, the default language is English.
Country	(Optional) Country for the display model. If none is selected, the default country is United States.

The Display Model editor is displayed for adding the display text for the artifact and its arguments. See [Editing a Display Model](#) for more details.

Result

If no language/country is chosen, a file with the default name of `<artifact_name>.display` is created. If language and country values are selected, the name of the file follows the standard Java I18N naming conventions, that is the name of the file is

`<artifact_name>_<language_code>_<country_code>.display`. For example, if English is selected as language and United States is selected as country, for the Concept named **Applicant**, the Display Model file is saved with the file name `Applicant_en_US.display`.

Editing a Display Model

You can define display text for each property of the artifact and also can hide a property from displaying in WebStudio using the Display Model editor.

The following fields are not editable but can be setup only while creating Display Model.

- Target Entity
- Language
- Country
- Variant

Procedure

1. In TIBCO BusinessEvents Studio, double click the Display Model file to open it in the form editor.
2. Enter the display text for the artifact (Concept or Event) in the **Target Entity Display Text** field.
3. In the **Display Properties** section, configure the display text for the properties of the artifact.
 - a) Click **Add**, select the properties for which you need to assign the display text, translate, or hide, and click **Finish**
The selected properties are listed.
 - b) Select a property and enter the required display text in the **Display Text** field.



In the case of Arabic characters, do not use accents, as they are not supported in TIBCO BusinessEvents.
 - c) Select the **Hidden** checkbox, if you do not want the property to appear in the WebStudio UI.
4. After you have added the display text for the required properties, save the Display Model.

Understanding Entity Caches

Understanding of entity caches is needed when working with the internal structures of the Oracle Coherence caches.

This information is provided for those who want to understand the internal structures of the Oracle Coherence caches used in Cache object management. It 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.

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: Internal entity names and caches are listed and described in [Internal Entity Caches](#).
- Ontology entities: 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

Specific 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

Specific internal caches use a pre-defined scheme in the cache configuration file.

Do not change this scheme. This information is provided for reference only.

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	TypeIDs are internal numbers used to store the short number to identify the URL of a concept instead the large string.
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 (<i>extId</i>).
AgentTable	Stores all the agents and their respective states across all cluster nodes and identifies the currently active and standby nodes.
AgentTxn- <i>agentId</i>	Each agent in the cluster has an AgentTxn- <i>agentId</i> cache. The <i>agentId</i> 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.

Handling Null Properties

There are several ways of handling null concept property values:

Enabling Use of the Nillable Attribute

The presence of the `xsd:nillable` attribute in an XSD element means that the corresponding element in the XML file permits null values.

Setting `tibco.be.schema.nil.attrs=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.

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.attrs
tibco.be.schema.exclude.null.props
```

If Null Properties are Excluded

```
tibco.be.schema.nil.attrs= 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.attrs` property has no effect on concept serialization.

If Null Properties are Included and the Nillable Attribute is Set

```
tibco.be.schema.nil.attrs=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.attrs=false
tibco.be.schema.exclude.null.props=false
```

In this case, each null property is considered to be an empty string, and is represented, for example, as follows:

```
<CustomerName/>
```

Special Treatment of Numeric and Boolean Null Values

If you enable null values to be output to XML (see [Enabling Null Property Values to Appear When Serializing Concepts to XML](#)), 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 [Properties for Null Property Handling](#)).

The special numeric values that indicate null are used in TIBCO BusinessEvents when serializing and deserializing a concept to and from its XML representation, and when performing various operations on database concepts and the database tables to which they are linked.

The special values that indicate null appear only in TIBCO BusinessEvents. The appropriate null value is used in the XML or database representation of the concept property.

Conversely, when deserializing or importing a concept instance, TIBCO BusinessEvents represents null values in the source using these special values that indicate null in the concept instance.

To enable special treatment of numeric null values, set the following property in `studio.tra`:

```
tibco.be.schema.treat.null.values=true
```

At runtime you can override the default values that indicate null. See [Setting Runtime Properties for Special Treatment of Null Values](#).

Summary

Set the following properties in `BE_HOME/studio/eclipse/configuration/studio.tra` as desired:

- To enable null property values to appear when serializing concepts to XML, add the following property and set it to false:


```
tibco.be.schema.exclude.null.props=false
```
- To enable use of the nillable attribute in the concept XSD, add the following property and set it to true:


```
tibco.be.schema.nil.attrs=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](#)).

To override the default values perform the following steps.

Procedure

1. In TIBCO BusinessEvents Studio, open the CDD editor for the project and add the following properties at the cluster level properties sheet. Provide the special values as desired:

```
tibco.be.property.int.null.value=value
```

```
tibco.be.property.long.null.value=value
```

```
tibco.be.property.double.null.value=value
```

```
tibco.be.property.boolean.null.value=value
```

Choose values that will not be misinterpreted as literal values.

2. Save the file.
3. Deploy the project with the updated CDD file.

Property Reference for Null Property Handling

Set the following properties in the `studio.tra` file as needed to configure the output for your needs before you generate the EAR file.

Properties for Null Property Handling

Property	Notes
Properties Set in <i>BE_HOME</i> /studio/eclipse/configuration/ studio.tra	
<code>tibco.be.schema.nil.attrs</code>	<p>Setting this property to true causes the <code>xsd:nilable</code> attribute ("<code>xsd:nilable=true</code>") to be set on all elements in the TIBCO BusinessEvents concept XSD.</p> <p>See Enabling Use of the Nilable Attribute</p> <p>Possible values are true and false.</p> <p>Default is false.</p>
<code>tibco.be.schema.exclude.null.props</code>	

Property	Notes
	<p>When this property is set to true, null-valued concept properties are not output when the concept is serialized to XML.</p> <p>When set to false, null-valued concept properties are output to XML.</p> <p>See Enabling Null Property Values to Appear When Serializing Concepts to XML.</p> <p>Possible values are true and false.</p> <p>Default is true.</p>
<code>tibco.be.schema.treat.null.values</code>	
	<p>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.</p> <p>Possible values are true and false.</p> <p>Default is false.</p>
Properties Set in CDD at the cluster level	
<code>tibco.be.property.int.null.value</code> <code>tibco.be.property.long.null.value</code> <code>tibco.be.property.double.null.value</code> <code>tibco.be.property.boolean.null.value</code>	
	<p>These properties define a special value that indicates null. Use a value that will not be confused with an actual numeric value.</p> <p>These properties are used only if <code>tibco.be.schema.treat.null.values</code> is set to true.</p> <p>Default values for each numeric datatype are the following Java constants:</p> <pre>int: Integer.MIN_VALUE long: Long.MIN_VALUE double: Double.MIN_VALUE boolean: Boolean.FALSE</pre> <p>For Integer and Long these constants represent the most negative value. For Double the constant represents smallest positive nonzero value (4.9e-324).</p>

Diagrams

TIBCO BusinessEvents provides several kinds of diagrams, which are visualization tools that help you 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.

Using the diagrams you can show or hide details. You can open editors for any project component in a diagram. You can also create snapshot images of diagrams to share project information with other personnel.

The main types of diagrams, and the project elements that use them, are as follows:

Selected Entity Project diagrams

These diagrams display all the selected project resources, and how they interact with one another. External resources such as XSD and WSDL files are not shown. You can either create a project diagram for an entire project, or only for the selected resources of a project. You can also choose to run the project analyzer while creating a project diagram, which analyzes the project resources and gives a report in the Problems view.

Dependency diagrams

These diagrams show the relationships between the selected project resource and its dependent resources. They are available for channels, concepts, state models, scorecards, events (all types except advisory events), rule functions, and rules.

Sequence diagrams

These diagrams show how project resources are called into use at run time. They are available for events (all types except advisory events), rule functions, and rules.

Concept model diagrams

These diagrams show the concept model for a project. You can view a model diagram for all the concepts in a project.

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.

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			

Project Resource	Project	Dependency	Sequence	Concept Model	Event Model
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
- 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

Method of creating diagrams	Diagrams you can create
<p>In a resource editor, click the appropriate diagram button in the top right corner.</p> <p>Concept Model Diagram </p> <p>Event Model Diagram </p> <p>Dependency Diagram </p> <p>Sequence diagram </p>	<p>Concept Model Diagram, Event Model Diagram, Dependency Diagram, Sequence Diagram</p>
<p>In a Selected Entity Project diagram, right-click a resource.</p>	<p>Dependency Diagram</p>
<p>In a Selected Entity Project diagram, right-click anywhere on the canvas.</p>	<p>Selected Entity Project diagram</p>

Common Tasks

You can perform the following common tasks on all types of diagrams:

- Hover the mouse over a resource to view a tooltip with details.
- Double-click any resource represented in the diagram to open its editor.
- Drag resources to any location in order to change the layout as desired.
- Drag any link to change the layout as desired.
- Click any arrowhead to move the arrow to any other element in the project. When you hover the mouse over an arrowhead, you see a small graphic on it().
- Double-click the black and white plus signs in a dependency or sequence diagram to unfold the next level of dependency.

Diagram Tools

You can work with diagrams with the help of the following tools available from the Diagram menu:

- [Interactive Tools](#), such as Select, Pan, Magnify, Marquee Zoom, Interactive Zoom, Link Navigator
- [Layout](#), such as Default Layout, Circular Layout, Orthogonal Layout, Symmetric Layout, Rectilinear Hierarchical Layout, and Oblique Hierarchical Layout
- Print tools: Print Setup, Print Preview, Print
- Export to Image
- Labeling
- Link Routing
- Incremental Layout
- Fit In Window

The above tools are also available on the toolbar when you create or display a diagram. The toolbar also has three additional tools listed below:

- Refresh Diagram
- Zoom Percentage

- Search Diagram Entities

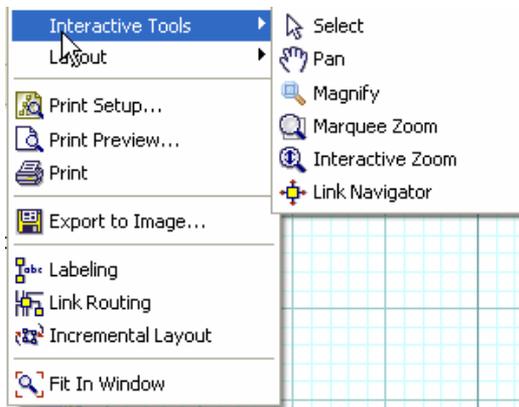


Diagram menu and the toolbar described in this section are available only when you create or open a diagram.

For more details about the diagram tools, see [Diagram Options and Tools Reference](#).

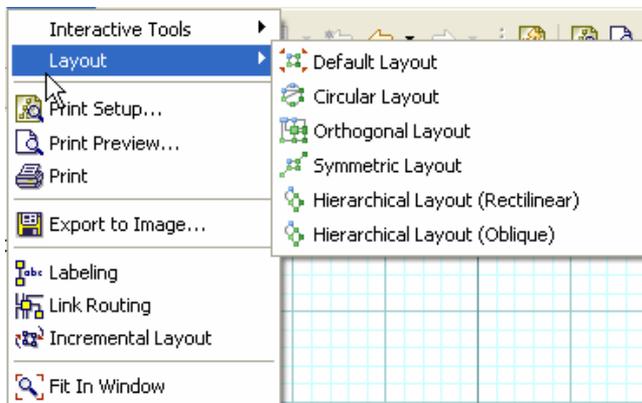
Interactive Tools

To access the Interactive Tools, select **Interactive Tools** from the **Diagram** menu. They are also available on the toolbar when you create or display a diagram. Interactive Tools let you select, pan, and magnify diagrams. They also include tools such as marquee zoom, interactive zoom, and link navigator. For more details about these tools, see [Diagram Options and Tools Reference](#).



Layout

You can access the **Layout** menu from the **Diagram** menu. They are also available on the toolbar when you create or display a diagram. Layout options let you change the layout of a diagram to Circular, Orthogonal, Symmetric, or Hierarchical. For more details about these tools, see [Diagram Options and Tools Reference](#).



Context Menu Diagram Tools

Diagram tools are available as context menus for diagram canvas and objects. [Context Menu Options for Canvas and Objects](#) lists these context menu options.

Context Menu Options for Canvas and Objects

Context Menu Option	Available for
Selected Entity Project Diagram	Canvas – Selected Entity Project Diagram
Export to Image...	Canvas – All diagrams
Print Setup...	Canvas – All diagrams
Print Preview...	Canvas – All diagrams
Fit In Window	Canvas – All diagrams
Create Dependency Diagram	Object – Selected Entity Project Diagram
Fold One Level	Object – Selected Entity Project Diagram
Fold N Levels...	Object – Selected Entity Project Diagram
Fold All Levels	Object – Selected Entity Project Diagram
Edit	Object – All diagrams

For more details about the diagram tools, see [Diagram Options and Tools Reference](#).

Exporting a Diagram to an Image

Using the Export to Image option you can save the diagrams locally for later use.

Procedure

1. From the top menu, select **Diagram > Export > to Image** . You see the Export Drawing to Image dialog.



2. Select the type of the image from the Type list.

3. In the **File Name** field, specify the file name along with the location.
4. Select the 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.

Result

The diagram is exported to an image, and saved on the system.

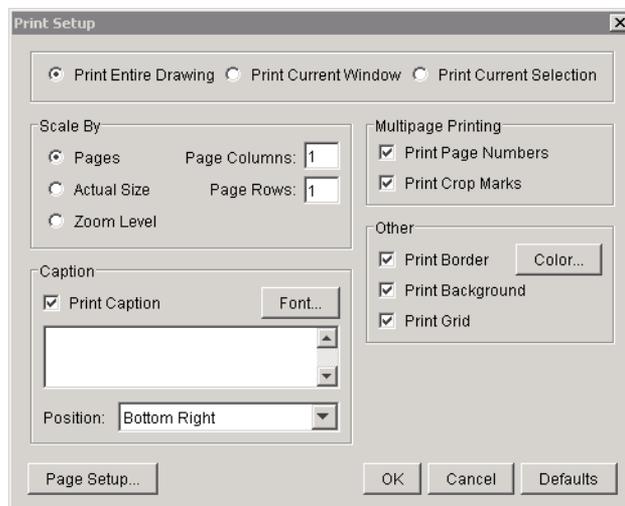
Printing a Diagram

You can set up preferences, preview the print job, and then print the diagram.

Using the Print Setup dialog you can specify the print setup for a diagram.

Procedure

1. Display the diagram, and from the top menu select **Diagram > Print Setup** .



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** check box. 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** check box. 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** check box.
8. Use **Page Setup** to specify the size, orientation, and other page properties.

9. If you want to reset all the options to the defaults, click **Defaults**.
10. Click **OK**.
11. To preview and print the diagram, select **Diagram > Print Preview** . An image of the diagram to be printed is displayed.
12. From the top menu select **Diagram > Print** .

Setting Diagram Preferences

You can set the preferences for all types of diagrams.

Procedure

1. From the top menu, select **Window > Preferences** .
2. In the Preferences dialog, expand **TIBCO BusinessEvents**, and then expand **Diagram**.
3. Select any type of diagram.

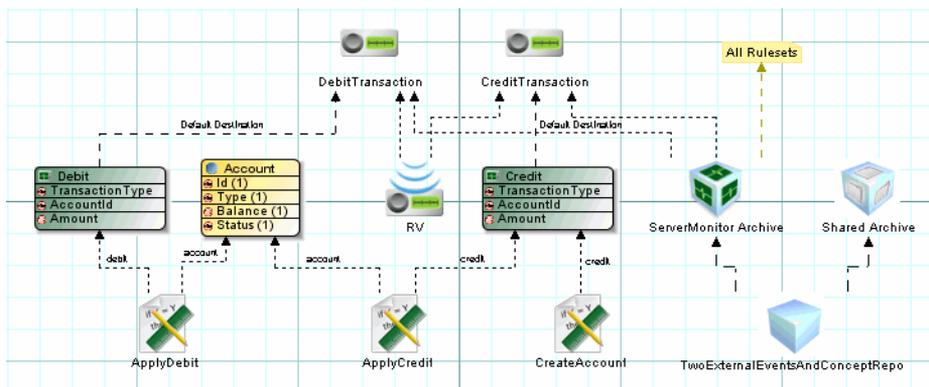
On the right side, you see preferences for the selected diagram type.

4. Edit the preferences, and click **Apply**.

OR

To set all preferences to their default values, click **Restore Defaults**.

Project Analyzer and Selected Entity Project Diagrams



Project Analyzer and the Selected Entity Project diagram work together to provide insights into your project. Project Analyzer is a document generated in the Problems view area. Selected Entity Project diagram provides a visualization of the whole project or of selected project elements, including dependencies. Selected Entity Project diagrams can display all the resources in a project, how they interact with one another, and how they use or update other resources at run time. You can see concepts, events, rules, rule functions, channels, destinations, 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 TIBCO BusinessEvents Studio, as well as print the entire project according to page setup.

The Project Analyzer report helps you analyze the project by performing the following tasks:

- Finds rules that can never be fired
- Finds rule functions that are never used
- Finds events that are never used/may never be fired
- Finds destinations with no default events
- Finds domain models that are never used, or that are not associated with entity properties
- Finds state models that are orphaned, or that are not associated with any concepts

Project Analyzer and Selected Entity Project Diagrams

You can show the diagram for an entire project or for selected elements only. You can also filter the diagram to show only certain types of project element.



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** check box, on the right.

You can also run **Project Analyzer** separately in either case. See [Project Analyzer and Selected Entity Project Diagrams](#).

To Run Project Analyzer Without Selected Entity Project Diagram

Right-click the top level project folder, and click **Analyze**. Project Analyzer displays the report in the Problems view.

Create a Selected Entity Project Diagram

Procedure

1. Do one of the following to create a Selected Entity Project diagram for the entire project:
 - In BusinessEvents Studio Explorer, right-click a project resource and select **Create Selected Entity Project Diagram**.

- Double-click the *.projectview file in the Studio Explorer.
OR, to create a Selected Entity Project diagram for selected resources
- In Studio Explorer, select resources of interest, right-click, and select **Create Selected Entity Project Diagram**.



When you select a folder all elements in that folder are selected.

2. The Selected Entity Project diagram appears in the editor area. The diagram tab label reflects the project name and the type of diagram: *projectName.projectview*, for example, *ExternalEventRepo.projectview*.

By default, the Selected Entity Project diagram displays the selected objects and their immediate dependencies.

3. You can change the depth of dependencies shown using preferences.
4. To filter what you see, select and clear the options in the Project Filter list within the Palette view.
5. Click **Apply**.

Result

Options show or hide elements of the specified type, or group them:

Show Options

Show Concepts, Show Events, Show Decision Tables, Show Domain Model, Show State Models, Show Archives, Show Rules, Show Rule Functions, Show Scorecard, Show Channels, Show Scope Links, Show Usage Links, Show Archived Destinations, Show Archived Rules, Show Archived Rules (All), Show Rules in Folders, Show Tooltips. If add-on products are used additional options may appear.

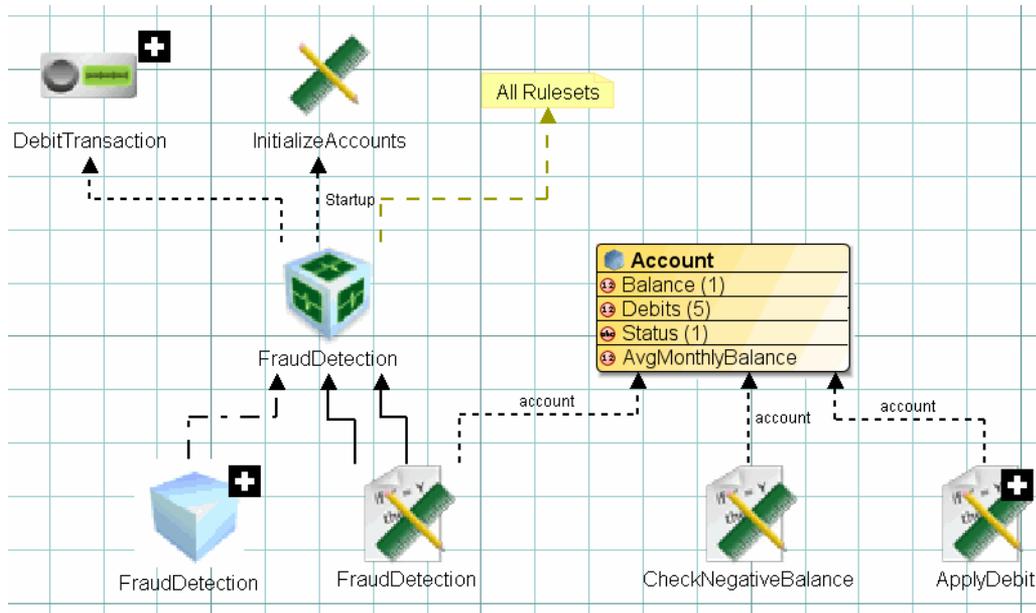
Group Options

Group Concepts, Group Events, Group Rules, Group Rule Functions. The filter options in the Palette view are loaded from diagram preferences.

Dependency Diagrams

A dependency diagram shows the dependencies for a selected project resource or for the entire project (Selected Entity Project Diagram).

When two resources are dependent and one changes, it can cause changes to the other resource as well. You can control how many levels of dependencies to view in a dependency diagram - one, two, or all levels.



Rule actions are not represented in rule dependency diagrams. Use Debugger to analyze rule actions. See [Testing and Debugging Projects](#).

Create a Dependency Diagram

Procedure

1. Do one of the following:

- In BusinessEvents Studio Explorer, right-click a project resource and select **Create Dependency Diagram**.
- Open the project element for editing and click the **Dependency Diagram** (📊) button in the top right of the editor.
- In a Selected Entity Project diagram, right-click a resource and select **Create Dependency Diagram**.

The dependency diagram appears in the editor area. By default, one level of dependency is shown which displays only direct dependents. White crosses indicate hidden dependencies. The default appearance is also driven by the diagram preferences.

The diagram tab label reflects the item being graphed and the type of diagram:

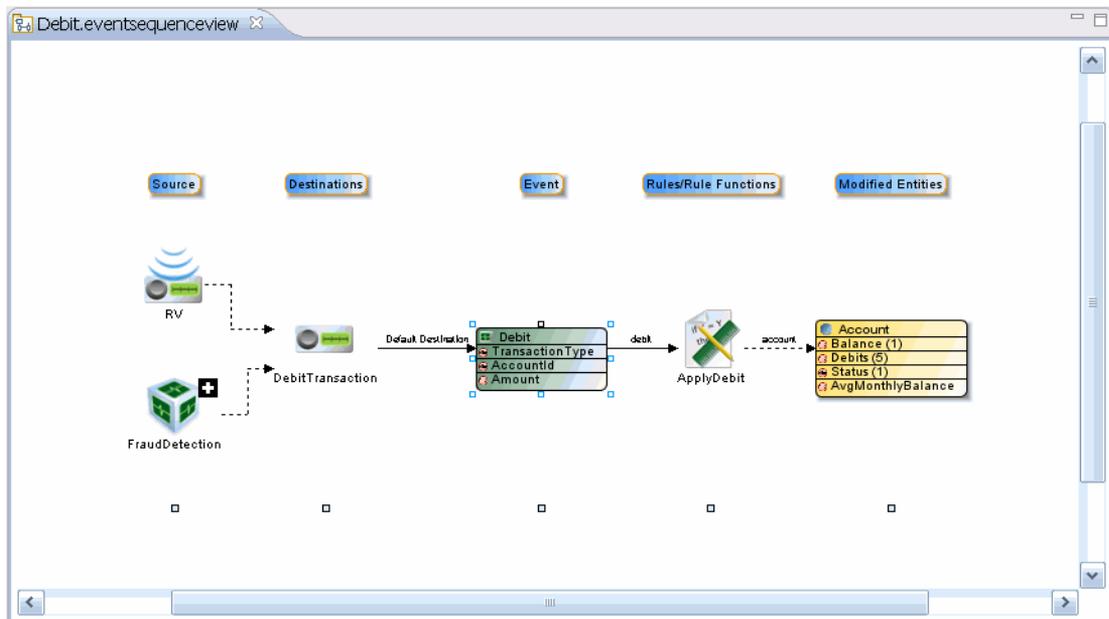
itemName.item-typedependencyview, for example, *Transaction.eventdependencyview*.

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.



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.

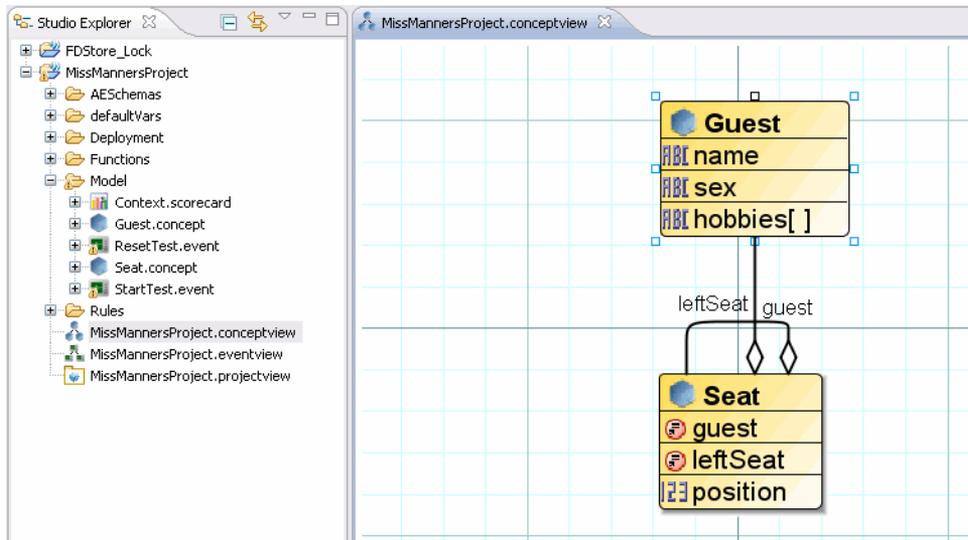
The sequence diagram for the item appears in the editor area.

The diagram tab label reflects the item being graphed and the type of diagram: *itemName.item-typesequenceview*. 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).



Create a Concept Model Diagram

Do one of the following:

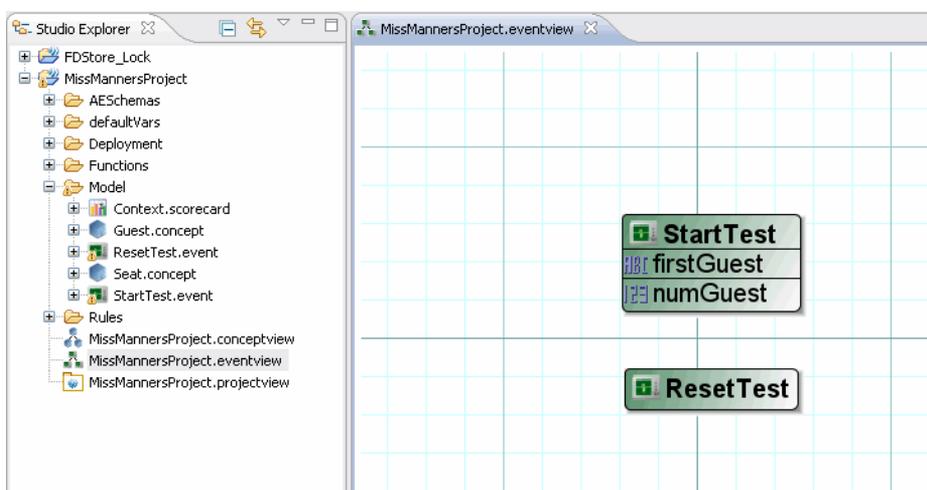
- Right-click a resource or a project in BusinessEvents Studio Explorer and select **Create Concept Model Diagram**
- Open a concept editor and click the Concept Model Diagram () button in the top right of the editor
- Double-click the *.conceptview file in the Studio Explorer.

The concept model diagram for that project appears, showing only its direct dependents (Dependency Level One).

Event Model Diagrams

The event model diagrams show inheritance relationships between all types of events, except for advisory events.

Event model diagrams are created fresh each time you view them. They do not persist on the disk.



Create an Event Model Diagram

Do one of the following:

- Right-click a resource or a project in BusinessEvents Studio Explorer and select **Create Event Model Diagram**.
- Open an event editor and click the **Event Model Diagram** () button in the top right of the editor. The event model diagram for that project appears, showing only its direct dependents (Dependency Level One).
- Double-click the *.eventview file in the Studio Explorer.

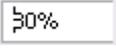
Diagram Options and Tools Reference

Most diagram options and tools are self-explanatory, and are just shown in the images. This section provides details for a few that require some explanation.

TIBCO BusinessEvents Diagram Tools Reference

Button	Description
Interactive Tools Menu Options and Toolbar Buttons	
	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.

Button	Description
	<p>Circular Layout It is available in the layout list. See Circular Layout.</p>
	<p>Orthogonal Layout It is available in the layout list. See Orthogonal Layout.</p>
	<p>Symmetric Layout It is available in the layout list. See Symmetric Layout.</p>
	<p>Hierarchical Layout (Rectilinear) It is available in the layout list. See Hierarchical Layout.</p>
	<p>Hierarchical Layout (Oblique) It is available in the layout list. See Hierarchical Layout.</p>
Other Diagram Menu Options and Toolbar Buttons	
	<p>Print Setup Sets options for printing the diagram or elements of the diagram. See Printing a Diagram.</p>
	<p>Print Preview Lets you see how the printed diagram would look.</p>
	<p>Print Prints the diagram.</p>
	<p>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.</p>
	<p>Labeling Prevents labels from overlapping with other elements and labels in the diagram.</p>

Button	Description
	<p>Link Routing</p> <p>Redraws the diagram, attempting to redraw only the links, leaving resource nodes in the same size and position where possible.</p> <p>The behavior of link routing is affected by the preference options Fix Node Size and Fix Node Position.</p>
	<p>Incremental Layout</p> <p>See Incremental Layout.</p>
	<p>Fit In Window</p> <p>Zooms the diagram to fit the size of the current diagram editor view.</p>
Additional Toolbar Buttons	
	<p>Refresh Diagram</p> <p>If you make changes in the project, clicking Refresh Diagram updates the diagram with the changes.</p>
	<p>Zoom Percentage</p> <p>Zooms the diagram to the specified percentage.</p>
	<p>Search Entities</p> <p>Allows you to search for TIBCO BusinessEvents entities in the diagram.</p>

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.)

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](#) 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](#) for a property that affects whether symmetric or hierarchical layout is used as the default, depending on number of edges. You can use Layout Preferences to determine in the direction of the hierarchy.

Preferences and options specific to the hierarchical layout are:

- Orientation Options (**Edit > Preferences**)—Left to Right, Top to Bottom, Right to Left, Bottom to Top
- Routing Options (**View > Layout**)—Orthogonal or normal (polyline) routing
- Routing Options (**Edit > Preferences**)—Orthogonal or polyline routing.

Incremental Layout

Choose Incremental Layout if you want the main arrangement of the diagram to remain stable when you make changes to your project and only re-route the entities and links that have changed since the last rendering.

Keeping most things in the same place makes it easier to see how changes you have made in the project affect the diagram.

If you want to refresh the entire diagram, click the desired layout option. The program has greater freedom to optimize the layout. The result is likely to be a more pleasing arrangement.

To perform an incremental layout update, do one of the following:

- Select **View > Layout > Incremental** Layout.
- Click the **Incremental Layout** button  on the toolbar.

Preferences

You can set up preferences in TIBCO BusinessEvents Studio on different levels and for different resources.



BusinessEvents Studio supports only left to right languages. If your operating system is configured to use any right to left language such as Arabic and you are using BusinessEvents Studio, you must not use the default layout of the operating system. Also, in **Window > Preferences > General > Globalization**, ensure that **Enable bidirectional support** is not selected and **Graphical layout direction** is selected as Left to right.

- TIBCO BusinessEvents (Top Level)

Using the top level preferences, you can do the following:

- Show or hide hover (tooltip) information in Catalog Functions view.
- Show or hide the confirm open perspective message pop-up.

- Switch or not switch to the default perspective every time an editor opens.
- Clear all the "do not show again" settings and show all hidden dialogs again.
- **Build Path and Classpath Variables**
Maintains a list of the classpath variables that have been defined. You can add, update, or remove classpath variables.
- **Code Generation**
 - Compilation Mode options are File System or In Memory
 - Source Java Version options are 1.5 or 1.6
 - Target Java Version options are 1.5 or 1.6
- **Code Generation and Ignored Resources**
Maintain the list of ignored resource patterns here: **Code Generation > Ignored Resources**. These preferences define what file types are excluded from the EAR file shared archive section at build time.
You can add and remove patterns, and select which existing patterns to ignore or stop ignoring.
- **Compare/Merge, Decision Table**
Compare/Merge preferences are used for comparing two decision tables. Decision tables are available only when the TIBCO BusinessEvents Decision Manager add-on is in use.
- **Diagram Preferences**
In addition to a global preferences section, there are sections for each type of diagram.
- **Error/Warning Preferences**
You can enable and disable error and warning messages for each type of resource in a project. Resource extensions are shown so you can select which ones to enable or disable. By default all are selected. If you make changes, a prompt appears asking if you want to rebuild the project.
- **Printing Preferences**
These preferences apply to printing of diagrams, windows, and selections.
- **Rules and Rule Debug Preferences**
In addition to the standard Eclipse Run/Debug preferences, TIBCO BusinessEvents requires some rule debug preferences.
- **Rules Management Server Preferences**
These preferences are used with TIBCO BusinessEvents Decision Manager.

Setting Preferences

To access preferences select **Window > Preferences** and expand **TIBCO BusinessEvents**.

Procedure

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.

Decision Table Related Preferences

Decision Table Preferences

The Table View preferences set the initial state of the following features. Text in parentheses shows the names of corresponding buttons in the Decision Table editor. The settings can be toggled in the UI, but their initial state is set by preferences you set:

- Show column alias if present.
- Automatically resize columns to fit content (Fit Content button.)
- Automatically resize rows to fit content.
- Automatically merge rows (Merge Rows button.)
- Show expanded text (Show Text button.)
- Show domain descriptions if present.
- Show column filter.
- Show titled border.
- Use existing IDs when importing.
- Export column alias.
- Automatically update decision tables on change in virtual rule function arguments.
- Show condition area string.
- The Use Non-resizable Editor sections option prevents or allows users from resizing editor sections.

Decision Table Analyzer Preferences

You can control the following behavior of the decision table analyzer, as well as its appearance:

- Highlight partial ranges
- Use domain model for table completeness
- Show analyzer contents while opening table.

Decision Table Appearance Preferences

These preferences let you set the color scheme for decision tables. For each item in the list, you can set color preferences. For condition data and action data, you can specify a font.

Compare/Merge Preferences

The tabbed dialog is in the TIBCO BusinessEvents preferences. It applies to decision tables.

In the **Structure Compare** tab (shown above), set preferences for initial settings and colors. The effect of your choices is shown in the lower panel.

In the **Text Compare** tab (shown below), you can set preferences for text. The effect of your choices is shown in the lower panel.

The **Merge** tab has one option: Allow columns to be automatically merged.

Diagram Preferences

The diagram preferences apply to the following types of diagrams:

- Concept

- Dependency
- Event
- Process (This option is available only if TIBCO BusinessEvents Process Orchestration is installed.)
- Project
- Sequence
- State Model (This option is available only if TIBCO BusinessEvents Data Modeling is installed.)

Preferences set using a diagram's palette are applicable only to the displayed diagram.

Preferences set using the Preferences dialog define the default preferences for diagrams of that type.

Preferences for specific diagrams override global preferences, where the same settings exist.

Global Diagram Preferences

Global Diagram Preferences

Option	Description
Reset Tool After Changes	If enabled, resets the tool to the Select tool after you add a node to the diagram. Default: Enabled.
Auto Hide Scrollbars	If enabled, then when the complete contents can display in the space available, scrollbars are hidden. If not enabled, scrollbars remain. Default: Auto hide scrollbars is enabled.
Show Tooltips	If enabled, you can see the tooltips when you hover the mouse on an element in the diagram. Default: Show Tooltips is enabled.
Link Types	Shows the links as straight lines or curved lines. Default: Straight.
Run Layout On Changes	After adding or deleting a node or an edge, refreshes the layout of the diagram. The layout options are None, Incremental, and Full. Default: None.

Option	Description
Animation	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>By default, animation is enabled along with Fade, Interpolation and Viewport Change.</p> <p>Duration: The time taken by animation to complete.</p> <p>Layout: Time, in milliseconds, for the layout change to complete. Applicable to the interpolation and fade options.</p> <p>Default for Layout: 500</p> <p>Viewport Change: Time, in milliseconds, for the viewport change to complete.</p> <p>Default for Viewport Change: 1000</p>
Opaque Movement	<p>Interactive Zoom Sensitivity specifies the sensitivity of the mouse in interactive zooming.</p> <p>Pan Sensitivity specifies the sensitivity of the mouse in panning.</p> <p>Default: Opaque Movement is enabled, Interactive Zoom Sensitivity is 200.0, and Pan Sensitivity is 1.0.</p>
Magnify Tool	<p>The size the zoom window, and the level of zoom.</p> <p>Default: Window Size is 250, and Zoom Level is 3.</p>

The preferences for all types of diagrams are listed in [Specific Diagram Preferences \(Alphabetical\)](#). The preferences are organized alphabetically and not grouped according to the diagram types.

Specific Diagram Preferences (Alphabetical) (Sheet of)

Option	Description
Cluster Layout Style	<p>Sets whether to show Selected Entity Project diagrams in a circular or symmetric clustered layout.</p> <p>Default: Circular.</p>
Create view when analyzing	<p>If enabled, creates a Selected Entity Project diagram when you choose to run the project analyzer.</p> <p>Default: Disabled.</p>

Option	Description
Dependency Levels	Sets how many levels of dependencies to view in a dependency diagram - One, two, or all. Default: One.
Filter Options	Sets whether to show various project resources in a Selected Entity Project diagram: Show Concepts, Show Events, Show Decision Tables, Show Domain Model, Show State Machines, Show Archives, Show Rules, Show Rule functions, Show Scorecard, Show Channels, Show Scope Links, Show Usage Links, Show Archived Destinations, Show Archived Rules, Show Archived Rules (All), Show Rules in Folders, Show Tooltips, Group Concepts, Group Events, Group Rules, and Group Rule Functions. Default: All enabled except for Archived Rules, Group Concepts, Group Events, Group Rules, and Group Rule Functions.
Fix node and edge labels	Keeps the labels of the nodes and edges with their graphics. Default: Enabled.
Grid	Sets the display either to without grid, or grid with lines, or grid with points. Default: Lines.
Layout Style	Sets either the orthogonal or hierarchical layout. Default: Orthogonal.
Layout Quality	Options are Draft (lowest quality), Medium, and Proof (highest quality). It 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.

Option	Description
Link Routing— Fix Node Sizes	<p>This setting affects link routing behavior for all layout options except hierarchical layouts.</p> <p>If enabled, node sizes do not change when you use the link routing feature.</p> <p>If disabled, link routing changes node sizes as needed for clarity.</p> <p>Default: Disabled.</p>
Orientation	<p>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.</p> <p>Options are: Top to Bottom, Bottom to Top, Left to Right, and Right to Left.</p> <p>Default: Top to Bottom.</p>
Orthogonal Fix Node Sizes	<p>This setting affects link routing behavior for the hierarchical layout option only.</p> <p>If enabled, node sizes do not change when you use the link routing feature.</p> <p>If disabled, link routing changes node sizes as needed for clarity.</p> <p>Default: Disabled.</p>
Run analysis when creating view	<p>Runs the Project Analyzer when creating a Selected Entity Project diagram.</p> <p>Default: Enabled.</p>
Run fast layout for large diagrams	<p>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.</p> <p>Default: Disabled.</p>
Show all properties in Concept Node	<p>If enabled, a diagram shows all properties of a concept node, instead of showing only four default properties.</p> <p>Default: Disabled.</p>
Show all properties in Event Node	<p>If enabled, a diagram shows all properties of an event node, instead of showing only four default properties.</p> <p>Default: Disabled.</p>
Show Catalog Functions	<p>Sets whether to show catalog functions in a Sequence diagram.</p> <p>Default: Enabled.</p>
Show Catalog Function Return Links	<p>Sets whether to show return links of catalog functions in a Sequence diagram.</p> <p>Default: Disabled.</p>
Show Expanded Names	<p>Sets whether to show expanded names in a Sequence diagram.</p> <p>Default: Enabled.</p>

Option	Description
Snap to grid	If you move nodes in a diagram and the grid is shown, the nodes snap to the grid lines if Snap to grid is enabled. Default: Disabled.
Undirected layout	Sets no orientation for the diagram. Default: Disabled.

Tester Preferences

The Tester preferences section lets you configure the tester behavior. See [Testing and Debugging Projects](#).

Reference to Tester Preferences

Option	Description
Output Directory	The directory in which the result files are stored. Default: /TestData/<Projectname>/<Processing Unit name>
No of WM Objects	Maximum number of objects in working memory using a preference. Default: 50
Auto scroll Modified Result Tables	If enabled, you see scroll bars for the cells in Result Test Data after clicking Fit Content. These scroll bars move together for both the After and Before sections. Default: Enabled

When you test a project, the tester shows the values changed while running the engine and the instances created in Result Test Data. To set the text color preferences for background and foreground of the modified values, go to **Window > Preferences > TIBCO BusinessEvents > Tester > Appearance** .

Reference to Tester Appearance Preferences

Option	Description
Background of the modified value	The background color of the changed values. Click the color box, which opens a color palette. Select the new color from it. Default: Light Pink
Foreground of the modified value	The text color of the changed values. Click the color box, which opens a color palette. Select the new color from it. Default: Blue
Font for the modified value	The font in which the modified value appears. Click Change to change the font. Default: Tahoma Regular 11