



TIBCO BusinessEvents® Process Orchestration

Developer's Guide

*Version 6.2.2
June 2022*



Contents

Contents	2
TIBCO BusinessEvents Process Orchestration Overview	5
Support for BPMN 1.2	6
Project Requirements	6
Types of Users	6
Process Diagram User Interface Overview	8
Run Time Overview	10
Task Summary	12
Business Process Diagrams	13
Processes	14
Types of Processes	15
Process Variables	15
Subprocesses	16
Elements in a Process	16
Connectors (Edges)	18
Sequence Flow	18
Events	20
Start Events and End Events	21
Activities	23
Activity Loop Characteristics	23
Task Timeout	25
SOAP JMS Call in Web Service Task	25

Configuring Web Service Task For HTTPS-SSL	26
Configuring Web Service Task For JMS-SSL	27
Java Task	29
Annotations	30
Activating Annotation Processing	31
Adding Java Task	32
Debugging Java Task	33
Do's and Don'ts for Java Task	34
Gateways	35
Exclusive	35
Parallel	36
Project Configuration Prerequisite to Support Processes	38
Project Resources Used in Process Elements	38
Adding a Process	40
Startup and Shutdown Processes	42
Configuring Startup and Shutdown Process	43
Process Execution (Run Time)	44
Backing Store and Checkpointing	44
Configuring Explicit Checkpoint for an Activity	46
Locking and Unlocking	47
Exception Handling	47
Adding and Configuring Process Agent Class	48
Running Process Agent	50
Building EAR file	50
Decision Table Hot Deployment in The Business Rule Task	51
TIBCO Documentation and Support Services	52

Legal and Third-Party Notices	54
--------------------------------------------	-----------

TIBCO BusinessEvents Process Orchestration Overview

TIBCO BusinessEvents Process Orchestration software is an add-on product to TIBCO BusinessEvents software. It provides complex event processing (CEP) functionality within the context of a business process management (BPM) process.

TIBCO BusinessEvents Process Orchestration software you can simply and efficiently *partition the rule space*, that is, to *segregate* different CEP rule sets within the flow of a BPM process. TIBCO BusinessEvents Process Orchestration provides the BPM functionality within the context of a CEP application, giving the business process, access to TIBCO BusinessEvents project resources and features.

Thus Process Orchestration combines the features of CEP and BPM technologies as follows:

- Using CEP applications you can correlate and process the events on a continuous basis. The outcome is not predetermined. It depends on events that arrive, the rules triggered, and the changes to the ontology that may result in triggering more rules, and so on. All rules active in an agent can be triggered if conditions are right.
- BPM applications, by contrast, orchestrate various activities into a predefined process, to achieve a predetermined business goal, such as provisioning a cell phone. Only certain rules should execute at any given step of the process.

When to Use Process Orchestration

With TIBCO BusinessEvents Process Orchestration, you can execute different tasks, each of which may use a few or hundreds of rules to complete.

Straight through processing

TIBCO BusinessEvents Process Orchestration provides the state processing model to automate the life cycle of a concept.

State models are available in the TIBCO BusinessEvents Data Modeling add-on.

Classical business process model

A classical business process model depends on the actions and illustrates the activities that occur in a predefined order (depending on conditions) to achieve the overall business goal of the process. This includes the workflow and Case Management (Intelligence).

Support for BPMN 1.2

TIBCO BusinessEvents Process Orchestration is compliant with a subset of the functionality specified in the Business Process Model and Notation (BPMN) specification version 1.2. It supports the functionality, appropriate for its role in a CEP application.

TIBCO BusinessEvents Process Orchestration provides event-driven process execution with BPMN-based modeling semantics. It is not intended to be a complete process modeling tool. For example, it does not offer human workflow, workforce allocation, document management, asset management, or organizational model capabilities.

If you also use the TIBCO BusinessEvents Data Modeling add-on product, you can extend the functionality provided using state modeling features (provided in TIBCO BusinessEvents Data Modeling add-on), as desired.

Project Requirements

The TIBCO BusinessEvents Process Orchestration project uses the Cache OM with Store persistence, and the process agents (a special type of agent) features of TIBCO BusinessEvents.

For details on designing and configuring projects, see [TIBCO BusinessEvents Documentation](#).

Types of Users

The add-on supports development of business process models by three kinds of users: business users, analysts, and technical users.

Business Users

Defines business processes at a high level, with annotations. Business users own the processes.

Analysts

Adds more detail so that the architect or developer can implement the model. Analysts interact with business users to improve and refine the model and update it in the TIBCO BusinessEvents Process Orchestration add-on.

Technical Users

Implements the design. They add necessary functions, simple and time events, Java code, and other items needed to fully implement the process, so that it can be executed by the TIBCO BusinessEvents engine.

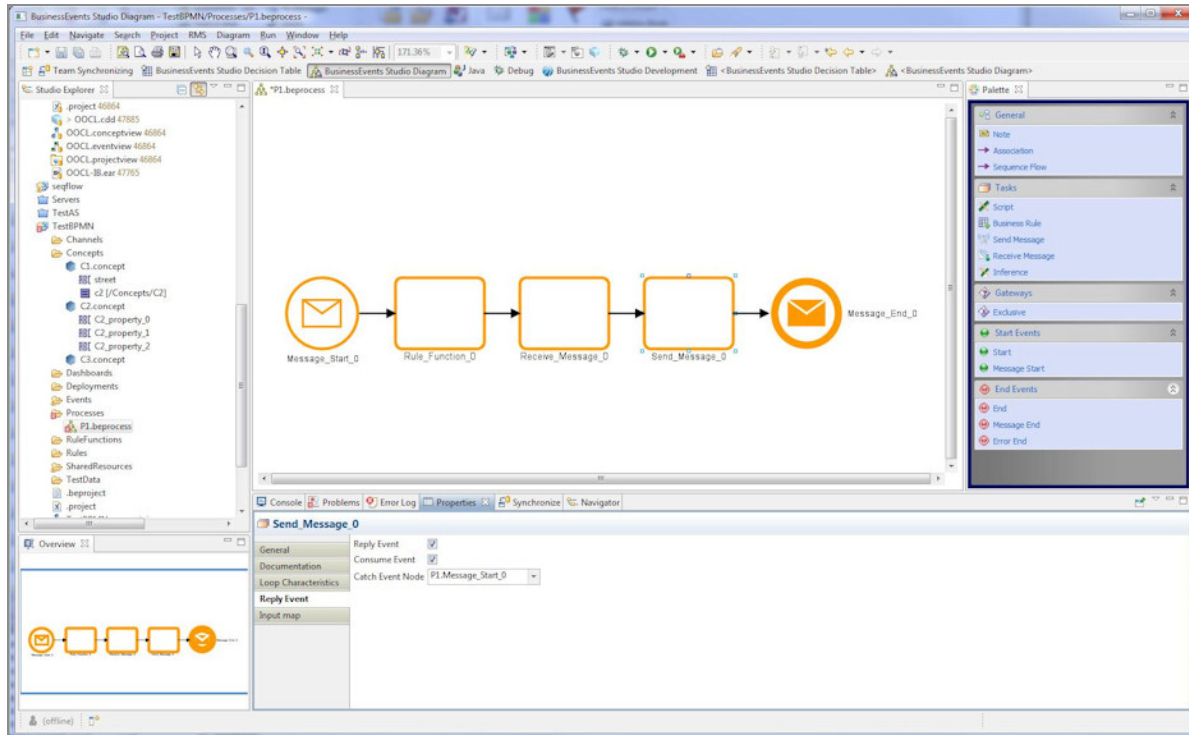
Prerequisite Skills

Technical users should be familiar with the Business Process Model and Notation (BPMN) specification, see <http://www.omg.org/spec/BPMN/1.2/PDF>.

All users should have some familiarity with business process modeling.

Process Diagram User Interface Overview

TIBCO BusinessEvents Process Orchestration uses the TIBCO BusinessEvents Studio and provides additional component to create process diagrams.



Studio Explorer

(top left) The standard project explorer as normally seen in TIBCO BusinessEvents. Processes are added as resources in the usual way.

Overview Window

(lower left) The standard overview window as seen in other diagrams within TIBCO BusinessEvents.

Editor

(upper middle) It uses a similar canvas-based user interface to other diagramming tools in TIBCO BusinessEvents.

Property Sheets

(lower middle) The context sensitive property sheets that you can use to view and edit items in the diagram that has properties. Select an item to display its properties.

Multiple left tabs switch between different sheets.

Palette

(right) It has various collapsible sections for different kinds of items you can drop onto the canvas when drawing a process. The palette is customizable and extensible.

Run Time Overview

At run time, you can use a process agent to select and deploy all processes marked public. All private processes are deployed internally.

Process Agent

A process agent is just an inference agent. The Process Orchestration engine works in cache-only mode and therefore, you must select the Cache Only OM option when configuring an inference agent. You can deploy any number of process agents, query agents, and cache agents.

i Note: A cluster can have either an inference agent or a process agent, but not both, so do not mix a stand-alone inference agent and process agent in a cluster.

i Note: The preprocessors are not supported in TIBCO BusinessEvents Process Orchestration.

Runtime Functionality

The following functionality is supported for process execution during run time:

- Process Template: is a model of a process (more details).
- Process Agent: runs all the deployed processes.
- Tasks Execution: various tasks such as Script, Business Rule, Send Message, Receive Message, Web Service, Inference, Sub_Process and Call Activity are supported during process execution at run time.
- Job: each instance of a process template is a job.
- Job Context: data plus additional state that each job maintains for status, recovery, reporting, metrics and so on.
- Checkpoint: allows persistence of the process job data to the cache and data store.

- ReplyEvent: allows the acknowledgment and reply to Start events coming through EMS,HTTP transport.
- Gateway Execution: both Exclusive and Parallel gateways are supported during process execution.

Task Summary

At a high level, some tasks are required to configure a TIBCO BusinessEvents project that contains one or more processes.

- Create a TIBCO BusinessEvents Studio project and add a JMS channel, events, and other project artifacts as needed.
- Create process diagrams with tasks and a default flow. This step can be performed at a high level by a business user, using the **Documentation** tab to “storyboard” the process, that is, describe the tasks and transitions in the process without completing the implementation.
- Link the tasks to project resources such as events, rule functions, and rules. In this step a technical team member assesses the storyboard version of the project (if any) and implements it using process and project resources.
- Validate the processes.
- In the project CDD, configure the process agent, load balancer, and other aspects of the project.
- Validate and deploy.

Business Process Diagrams

An introduction to the components of a business process diagram, and how in TIBCO BusinessEvents Process Orchestration, each component leverages the features of TIBCO BusinessEvents are explained.

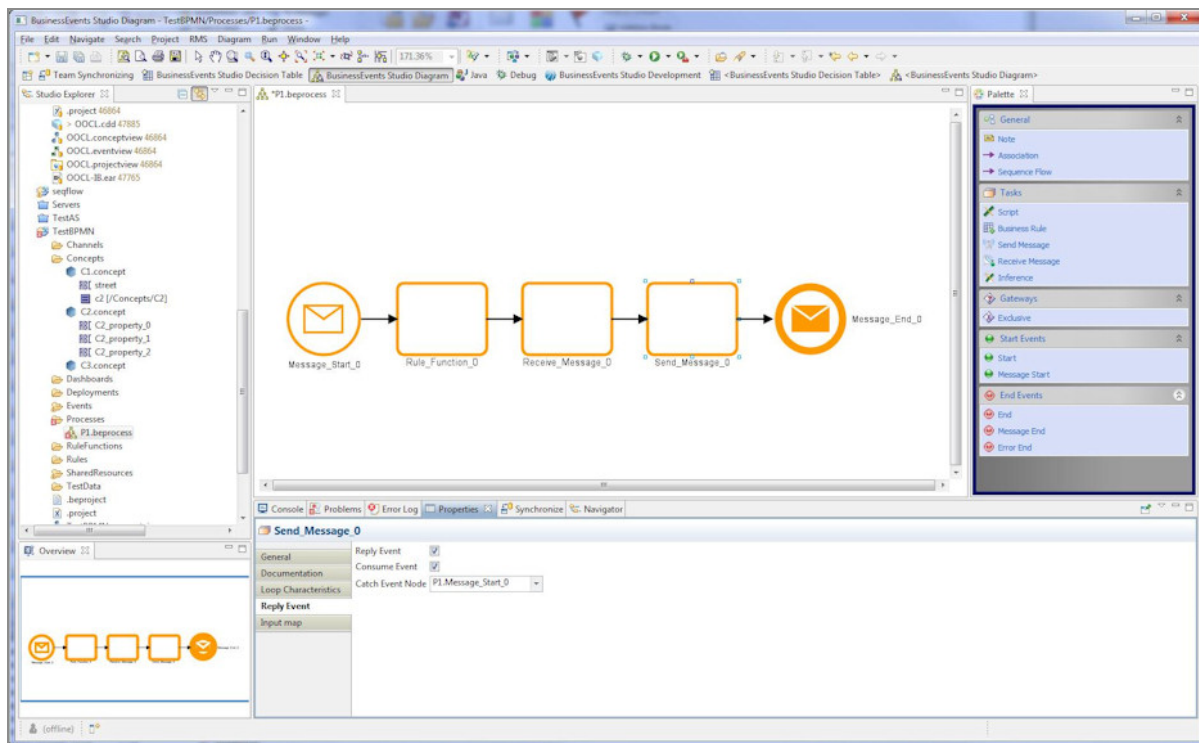
Many resources on BPMN are available and this manual does not attempt to present a BPMN primer. However, it does outline some basics so that you can understand how TIBCO BusinessEvents project resources are used to implement the functionality of different parts of a BPM process.

Processes

A business process is a flow of business activities related to one business goal, such as providing a complex service to a customer. A process diagram is a graph that describes the flow of activities in a business process. The graph is built up of connected elements of different types such as activities and events.

A process can describe different paths of which one or more paths are taken at run time, depending on conditions.

A notional *token* flows through the process, indicating which particular path or paths are taken as the process executes.



Processes are of two main types, public and private.

Public Processes

A public process can be added to a process agent in CDD and deployed. Public processes are often instantiated by the arrival of a message at a TIBCO BusinessEvents destination,

which causes an instance of a process and run the process.

Private Processes

Private processes are only used within a public process. They can be called using a *call activity* or they can be triggered by an error, or other situation.

Private processes can be shared by multiple public processes.

Private processes are automatically deployed when a public process that references them is deployed. All private processes are deployed.

Types of Processes

Processes are of two main types, public and private.

Public Processes

A public process can be added to a process agent in CDD and deployed. Public processes are often instantiated by the arrival of a message at a TIBCO BusinessEvents destination, which causes an instance of a process and run the process.

Private Processes

Private processes are only used within a public process. They can be called using a *call activity* or they can be triggered by an error, or other situation.

Private processes can be shared by multiple public processes.

Private processes are automatically deployed when a public process that references them is deployed. All private processes are deployed.

Process Variables

Process variables are similar to concept properties in TIBCO BusinessEvents. Process variables are used to hold data that enters the process during its execution at run time. Using an XPath mapper you can map data from simple events to process variables and from process variables to resources such as other simple events, rules, and so on.

You can define process variables of these types: String, int, long, double, Boolean, DateTime, and Concept. Variables can be flagged as Multiple, that is, arrays.

If you want to rename the process variables, then right click the process variable and select **Rename**. The variable name is updated everywhere in the process.

When you select Concept as the process variable type, you select a concept from the project ontology. This concept behaves like a contained concept, with respect to the process, which behaves like the containing concept.

Subprocesses

Subprocesses are processes that exist within another process.



Note: A subprocess has access to all variables in all its parent processes (including parent subprocesses).

Subprocesses are of two types, standard and event-based.

Standard Subprocess





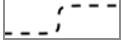

A standard subprocess can only be reached by a sequence flow in the parent process.

A standard subprocess can begin with a unique empty Start Event, or an activity or gateway that has no incoming sequence flow. It cannot start with a message, signal, or timer start event.

Elements in a Process

A BPMN process is a set of different types of *flow elements*.

The following table introduces these BPMN standard elements supported in this release, and their icons. The icon shapes are as specified in the BPMN standard. Subtypes of each element type have the same basic shape with distinguishing characteristics, also as specified in the standard. Swim lanes are not supported in this release.

Element Types	Elements and Icons		
Flow elements			
	Events (Start, End) See Events	Activities (Task, Sub-process) See Activities	Gateways See Gateways
Connectors			See Connectors (Edges)
	Sequence flow	Association	
Artifacts			
	Note		

Connectors (Edges)

Connectors in a process diagram links two objects which are either associated or need to be executed in a sequential order.

Connectors (which might also be termed connectors, edges, or transitions) are of two types:

- An *association* associates data or artifacts with flow objects. They can show how data is provided to or generated by an activity. They can link text annotations with the objects they annotate.
- A *sequence flow* is a connector that indicates the order in which activities are done. Sequence flow connectors connect events, activities, and gateways.

The next section explains more about sequence flow.

i Note:
Sequence Flows Emerging from Gateways

In this release, only sequence flows emerging from gateways can have expressions.

Sequence Flow

A sequence flow is an arrow that links flow objects. Sequence flows (along with gateways) show the order in which activities are performed in a process.

A sequence flow can be one of the following kinds:

- **Default**

When multiple paths flow from a gateway, select the **Default Flow** check box to specify one of them as the default. The default path is taken only when no other path is valid.

- **Conditional**

If you add a Boolean expression to the sequence flow's **Expression** tab, the flow proceeds to the next task only if the expression evaluates to true.

- **Uncontrolled**

A flow that is unaffected by conditions, and does not pass through a gateway. Can be used for simply connecting two activities, or can diverge from or converge to an activity, in parallel. (Similar to the lambda transition in a state model)

Note that the BPMN specification discusses additional types of flow. In TIBCO BusinessEvents Process Orchestration, various types of flow are selected internally based on context.

Events

Event is a technical term both in BPMN and in TIBCO BusinessEvents Process Orchestration. A BPMN *event* is a signal that something happened.

A BPMN event can be one of three types: start, intermediate, or end. Each type has subtypes used for different purposes.

Message and Event

In BPM, the term *message* is used to denote anything that arrives or leaves a process. In TIBCO BusinessEvents, the term *message* denotes information arriving at a destination, where it is transformed into a TIBCO BusinessEvents simple event. In TIBCO BusinessEvents, the term *event* is often used loosely to mean *message*, for example, “the event arrives at the JMS destination.”

BPMN Events and TIBCO BusinessEvents Events

BPMN event types are implemented using various TIBCO BusinessEvents resources. The Message Start BPMN event maps to a TIBCO BusinessEvents simple event, whose arrival causes an instance of a process (that is, a job) to be created.

BPMN Event Triggers and Results

In BPMN, events usually have a cause (*trigger*) and an effect (*result*) as follows:

Events that catch a trigger

Start events and some intermediate events are catching events. The trigger is the cause. The trigger can be a simple event or it could be a system alert about shutdown, for example.

Events that throw a result

End Events and some intermediate events throw events. The result they throw could be a TIBCO BusinessEvents simple event, timer event, or an event subprocess.

Start Events and End Events

Start events indicate the beginning of a process. End events indicate where a path of a process ends.

Implicit and Explicit Start and End Events

Use of start events and end events is optional. Other types of activities can be used to start a process, as long as they have no input mapping. Gateways can also be used to start a process. Similarly, other types of activities can be used to end a process, as long as they have no output mapping. These are known as *implicit start events* and *implicit end events*.

For an explicit start event, there must also be an explicit end event, and the reverse is also true. Also, if there are start and end events, implicit start and end events cannot be used. A process does not end until all start events have been visited, or in the case of implicit start events, all parallel paths have been completed.

Effect of Abnormal Termination

If a subprocess ends abnormally, the containing process does not terminate. If a multi-instance activity terminates abnormally, only the affected instance is terminated. (See [Activity Loop Characteristics](#) for an explanation of a multi-instance activity.)

Message Start and Signal Start Events

Message start and signal start events are configured using TIBCO BusinessEvents simple events. Validation checks at design time ensure that you use a correctly configured simple event:

- Message start events require a simple event with a queue destination.
- Signal start events require a simple event with a topic destination.

Example

As a simple example, suppose a process begins with a message start event and ends with a message end event. In such case, the process instance is instantiated when the simple event specified in a message start event arrives through a destination. The simple event properties appear as scope variables in the Output map, and you map them to process (job) variables, so they are available to the other activities later in the process.

At the end of the process the end event transfers the information available from the Input map (that is, values stored in the process variables) to the specified simple event's properties and sends it to its default destination or specified destination.

Timer Start Events

The start of the process is triggered by a time-date or repeating time interval, for example, every Monday at 9am. In BPMN, the timer start event uses the TIBCO BusinessEvents time event trigger, therefore the event can be a repeating time event or a rule-based scheduled time event. See the TIBCO BusinessEvents timer event definition.

Activities

A BPMN *activity* is a unit of work performed within a process. Activities are flow objects and they are shown using round-cornered rectangles. They are also known as tasks.

The following are the BPMN activity types:

- Tasks
- Call activities
- Subprocesses

Various task subtypes are available. Most of them provide TIBCO BusinessEvents-specific functionality, such as Send Event, Decision Table, Rule Function, and so on.

Activity Loop Characteristics

Activities can have looping characteristics.

When adding an activity, you can optionally use the Loop Characteristics tab to configure the activity to be a loop activity (Standard option) or multi-instance activity (Multi-Instance option). Activities configured in these ways are also known as *loop activities* and *multi-instance activities*.

Standard Looping

Standard looping behavior is based on a Boolean condition: the activity repeats (loops) as long as the Boolean condition is true. You can configure the evaluation to occur either at the start or end of each loop iteration and you can specify a maximum number of iterations.

Multi-Instance Looping

While standard looping continues until a condition, the multi-instance option lets you specify a certain collection of activity instances to execute. The number of instances is evaluated once.

The number of instances can be determined using an XPath expression.

Loop Characteristics Tab

The options on the Loop Characteristics tab let you configure the looping behavior. Types of looping behavior are as follows:

Field	Description
Loop	Standard or Multi-Instance.
Standard Loop Fields	
Count	Maximum number of loop iterations. Click Edit to use the XPath formula editor to define a Integer expression that evaluates to the variable on which the iteration will be performed. This is an optional setting. If not set, the activity loops until the Loop Condition expression evaluates to false.
Test Before	<p>If selected, the condition is checked at the beginning of an iteration of the loop.</p> <p>If cleared, the condition is checked at the end of an iteration of the loop.</p>
Condition	Click Edit to use the XPath formula editor to define a Boolean expression. The looping continues while the Boolean expression evaluates to true. You can use process variables, functions, and constants.
Multi-Instance Loop Fields	
Collections	Click Edit to use the XPath formula editor to define a XPath expression that specifies the list of objects over which the iteration happens. Collections could be of primitive data types array or concept array.
Test Before	<p>If selected, the condition is checked at the beginning of an iteration of the loop.</p> <p>If cleared, the condition is checked at the end of an iteration of the</p>

Field	Description
	loop.
Condition	Click Edit to use the XPath formula editor to define a Boolean expression. The looping continues while the Boolean expression evaluates to false and stops when the expression evaluates to true. You can use process variables, functions, and constants.

Task Timeout

If in a receive task, the process instance job is waiting for the incoming message, but the message is not arriving in appropriate time. Then the process needs to send out an notification event of such a situation after the time out period.

To setup time out in the task (for example, the Receive Message task) of a process, create a scheduler rule function task before the task (Receive Message). The scheduler rule function schedules invocation of a event, which starts another process. You can use the `Cluster.scheduleEvent` catalog function in the rule function for scheduling the event after the specified time interval. While creating the new event, pass the process ID and next activity name that would be performed in case of a timeout.

This scheduled event is the message start event for another process. The new process consists of activities that need to be performed after the time interval. For a simple time out process, create a rule-function task, and call the `Process.Activity.moveTo` catalog function to move to the specified *process* and *activity* which were passed by the scheduler (for example, the activity after the Receive Message task).

SOAP JMS Call in Web Service Task

Using SOAP/JMS calls is similar to using the SOAP/HTTP calls in a web service task. You can use a web service task for making the SOAP/JMS calls.

The WSDL file for SOAP/JMS needs to be associated with the web service task. Under the web service task properties, click **Browse** to select the properly configured WSDL file. All the other fields, such as, **Service**, **Port**, **Operation**, and **SOAP Action** are populated using the WSDL content.

For configuring the task for JMS you can configure the **Transport** Tab; otherwise the fields gets populated through the WSDL associated with the web service task.

The input mapper is used for sending values for SOAP request and the output mapper is used for getting the response value.

Configuring Web Service Task For HTTPS-SSL

You can configure a Web Service task to execute a remote service which is SSL enabled using the HTTP transport.

Procedure

1. Select Web Service task in the process.
2. Select the **Transport** tab under **Properties**.
3. In the **End Point Url** field, verify that correct https URL is displayed. This defines the URL of the web service to which a request is sent to. This field is automatically populated depending upon the WSDL selected in the **General > Resource** field. For example,

```
https://xyzserver:9000/receiverServices
```

4. Click the **Configure SSL** button and configure the following SSL parameters:

HTTPS SSL Configuration Parameters

Field	Description
Trusted Certificates Folder	Specifies the folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked to ensure trusted connection with server. This prevents connections to rogue servers that attempt to impersonate trusted servers.
Identity	Specifies an Identity resource that contains the client's digital certificate and private key. See <i>TIBCO Designer Palette Reference</i> for more information.

Field	Description
Verify Host Name	<p>When checked, this field is used to ensure that the host name of the SOAP server is checked against the host name listed in the server's digital certificate. This provides additional verification that the host name you are connecting to is in fact the desired host.</p> <p>If the host name specified in the Endpoint URL field on the Transport tab is not an exact match to the host name specified in the server's digital certificate, the connection is refused.</p> <p>Note: If you specify an equivalent hostname (for example, an IP address) in the Endpoint URL field on the Transport tab, but the name is not an exact match of the hostname in the host's digital certificate, the connection is refused.</p>
Strong Cipher Suites Only	<p>When checked, 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.</p> <p>See <i>TIBCO ActiveMatrix BusinessWorks Administration</i> for more information.</p> <p>The default value of the property disables cipher suites with an effective key length less than 128 bits.</p> <p>When this field is unchecked, only cipher suites with an effective key length of up to 128 bits can be used.</p>

5. Save the project.

Configuring Web Service Task For JMS-SSL

You can configure a Web Service task to execute a remote service which is SSL enabled using JMS transport.

Procedure

1. Select Web Service task in the process.
2. Select the **Transport** tab under **Properties**.

3. Verify the JMS and JNDI parameters are correctly configured.
4. Select the **Use SSL** check box to use SSL for JNDI server connection.
5. In **SSL Password**, enter the password any) to be used to connect to JNDI server.
6. Click the **Configure SSL** button and configure the following SSL parameters:

JMS SSL Configuration Parameters

Field	Description
Trusted Certificates Folder	Specifies the folder in the project containing one or more certificates from trusted certificate authorities. This folder is checked to ensure trusted connection with server. This prevents connections to rogue servers that attempt to impersonate trusted servers.
Identity	Specifies an <i>Identity</i> resource that contains the client's digital certificate and private key. See <i>TIBCO Designer Palette Reference</i> for more information.
Trace	Specifies whether SSL tracing is enabled during the connection. If checked, the SSL connection messages are logged and sent to the console.
Debug Trace	Specifies whether SSL debug tracing is enabled during the connection. Debug tracing provides more detailed messages than standard tracing. If checked, the SSL connection (debug level trace) messages are logged and sent to the console
Verify Host Name	When checked, this field is used to ensure that the host name of the SOAP server is checked against the host name listed in the server's digital certificate. This provides additional verification that the host name you are connecting to is in fact the desired host. If the host name specified in the Endpoint URL field on the Transport tab is not an exact match to the host name specified in the server's digital certificate, the connection is refused. Note: If you specify an equivalent hostname (for example, an IP

Field	Description
	address) in the Endpoint URL field on the Transport tab, but the name is not an exact match of the hostname in the host's digital certificate, the connection is refused.
Expected Host Name	<p>Specifies the name of the host you are expecting to connect to. This field is relevant only if the Verify Host Name field is checked.</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>
Strong Cipher Suites Only	<p>When checked, 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.</p> <p>See <i>TIBCO ActiveMatrix BusinessWorks Administration</i> for more information.</p> <p>The default value of the property disables cipher suites with an effective key length less than 128 bits.</p> <p>When this field is unchecked, only cipher suites with an effective key length of up to 128 bits can be used.</p>

7. Save the project.

Java Task

The Java task enables you to add functionality to call and execute user-defined Java code. The functionality is similar to the rule function script task and provides an integrated development and debugging environment.

In TIBCO BusinessEvents Studio integrated development environment (IDE) you can write code in Java and Rule Language in the same project. You can also seamlessly debug the code written in Java and Rule Language.

Using the Java task you can use third party Java code libraries to integrate with TIBCO BusinessEvents.

Data Type

The Java Rule API allows for easy integration with TIBCO BusinessEvents type system, such as concepts, process, and TIBCO BusinessEvents primitive data types.

The following are the accepted Java primitives:

- int
- double
- long
- boolean

The following are the accepted wrapper objects:

- Integer
- Double
- Long
- Date-Time
- Boolean

String data type is also supported. For computation of the collections, you can use Array of all of the accepted types.



Note: In the Java code you can only use `java.util.Calendar` type for referring to Date-Time data types of BusinessEvents.

Annotations

You can use the Java code in the TIBCO BusinessEvents project by using annotations, which are supposed to be used for the rule language. Without proper annotations, the Java code will not be recognized by TIBCO BusinessEvents. Only the methods that have proper annotations can be selected as Java resource.

Annotations for Java Task

Annotation and Classes	Description
@JavaTask	Identifies a class that can be used for the Java Task.
@JavaTaskContext	Identifies a task context field of a given TaskContextType.
@JavaTaskMethod	Identifies a Java task function.
@ModelTypeMap	Required annotation that must be incorporated into your project. Identifies a type mapping between Java data type and TIBCO BusinessEvent model type (ModelType).

Apart from these annotations, two enumeration classes TaskContextType and ModelType are used. For more information on the annotation types, see *TIBCO BusinessEvents® Online Reference*.



Note: The custom functions and Java functions can be used in same places but their annotations are not the same. While using the BPMN annotations the functions can be used in the Java task's input and output mapping activities, the custom catalog functions restricts you from mapping.

Activating Annotation Processing

For compile annotation error reports, enable Annotation Processing in the Studio project properties.

Procedure

1. In the Studio Explorer, right click on the project and select **Properties**.
2. Select **Java Compiler > Annotation Processing**.
3. Select the following check boxes and click **Apply** and then **OK**:
 - Enable project specific settings
 - Enable annotation processing

- Enable processing in editor

Adding Java Task

Using the Java task you can add functionality to call and execute user-defined Java code.

Procedure

1. Add the Java task from the palette to the process editor.
2. Under Properties for the Java task, select the **General** tab, and click on the **Java Resource** hyperlink.

The New Java Task Class wizard is displayed.

3. Enter or select the parent Java source folder and specify the Java class file name in the **Java Source Name** and click **OK**.

A new Java class file is created with the specified name in the selected Java source folder. The **Java Resource** field for the Java task is populated with the Java class file path. The new Java class file is populated on the basic Java task method and annotation to get you started. You can also write your own methods similar to the default method using proper annotations, as identified in [Table 3, Annotations for Java Task](#),.

4. Use the **Input Map** tab, for passing the value similar to what is done in the Java method call. The values mapped in the input mapper are used as arguments for the called function.
5. Use the **Output Map** to obtain the value returned by the function after execution. The mapping can be done to execute the job so as to use the value for future job execution.
6. To re-factor the Java class, right click the Java class from the studio explorer and select **Refactor > Move** or **Refactor > Rename**. Using this method you don't have to clean build the project for the changes to be properly reflected.

Sample BPMN Java Task

```
package com.tibco
import java.util.Calendar;
import com.tibco.cep.bpmn.runtime.activity.tasks.JavaTaskContext;
```



```

import com.tibco.cep.bpmn.runtime.activity.tasks.ModelType;
import com.tibco.cep.bpmn.runtime.activity.tasks.ModelTypeMap;
import com.tibco.cep.bpmn.runtime.activity.tasks.TaskContextType;
import com.tibco.cep.bpmn.runtime.model.JavaTask;
import com.tibco.cep.bpmn.runtime.model.JavaTaskMethod;
import com.tibco.cep.runtime.model.element.Concept;
import com.tibco.cep.runtime.session.RuleServiceProvider;
import com.tibco.cep.runtime.session.RuleSession;
@JavaTask
public class TestJavaTask {
    @JavaTaskContext(TaskContextType.NAME)
    String taskName;
    @JavaTaskContext(TaskContextType.RULE_SERVICE_PROVIDER)
    RuleServiceProvider rsp;
    @JavaTaskContext(TaskContextType.RULE_SESSION)
    RuleSession ruleSession;
    @JavaTaskMethod
    @ModelTypeMap(type=ModelType.CONTAINED_CONCEPT,
uri="/Concepts/ConceptA")
    public Concept taskFunction(
        @ModelTypeMap(type=ModelType.INT) int a,
        @ModelTypeMap(type=ModelType.STRING) String b,
        @ModelTypeMap(type=ModelType.DATETIME) Calendar c,
        @ModelTypeMap(type=ModelType.CONTAINED_CONCEPT,
uri="/Concepts/ConceptA") Concept cc,
        @ModelTypeMap(type=ModelType.PROCESS,
uri="/Processes/ProcessA") Concept pp) {
        return null;
    }
}

```

Debugging Java Task

When you debug the Java code, you can put a breakpoint in the Java class files and also view compilation annotation error reports.

Follow these steps to ensure that the class that you write for the Java task is packaged into the `be.jar` file. Ensure that catalog names and Java task function names are kept separate.

Procedure

1. Select the line of code.
2. Double click on the left margin.

Result

For more details about Debugging Java, see *TIBCO BusinessEvents® Developer's Guide*.

Do's and Don'ts for Java Task

With the Java task you can use Java at the task level, however some restrictions and guidelines apply.

- The Java task code is meant to be used with synchronous code; if asynchronous code is used, it blocks the worker threading model of the process.
- Java files are to be created inside a Java source folder for TIBCO BusinessEvents to index it. Best practice is to create a package and not use the default package. Specify the Java file in the **Java Resource** field of the Java task.
- Without proper annotations, the Java code will not be recognized by TIBCO BusinessEvents. Only the methods that have proper annotations can be selected as Java resource.
- You can enter any names for your variables, but the variable types must stay consistent with what is provided in the annotations.
- Do not use the names of reserved words or existing packages as your variable names when combined with @BE Function annotation to catalog functions. Packages that are reserved include **be.gen** and **com.tibco**, which are TIBCO created classes and objects. After the code is compiled, it should not collide with any existing packages. For a complete list of all reserved words see *TIBCO BusinessEvents® Architect's Guide*.
- Avoid static variables as they will be shared between different instances of tasks.
- Certain tasks causes the process to become dormant and are only meant to be evoked by external events.
- Practice caution when any threading operations are executed in the BPMN Java Task, as the code must be synchronous. Avoid using multi-threading between Java tasks.
- Ensure that the class that you write for the Java task is packaged into the `be.jar` file.
- Ensure that catalog names and Java task function names are kept separate.

Gateways

Gateways control how the process flows. They define interactions between sequence flows in a process.

The following types of gateways are supported:

- Exclusive
- Parallel

**Note:****Sequence Flows Emerging from Gateways**

In this release, only sequence flows emerging from gateways can have expressions. Only one path is taken, the first one whose expression evaluates to true. If no path evaluates to true, the default path is taken.

Exclusive

Exclusive gateways (also known as XOR gateways) specify multiple possible paths, only one of which is actually taken. All the outgoing flow paths are evaluated and the flow whose condition is evaluated to true is taken for process execution.

Exclusive gateways are used in the following two ways:

- Join: specifies the branching of the flow into multiple possible paths, or to merge these possible paths.
- Fork: specifies the possible multiple paths. The selection logic is specified in the sequence flows. One sequence flow is defined as the default.

Joining

A converging exclusive gateway is used to merge alternative paths.

Each incoming control flow token is routed to the outgoing Sequence Flow without synchronization. Each token arriving at any incoming sequence flow activates the gateway and is routed to exactly one of the outgoing sequence flows.

The Exclusive Gateway has pass-through semantics for a set of incoming branches (merging behavior). This behaves like the parallel gateway merging except that each incoming sequence requires an individual rule and the rule is satisfied with the message token coming through the sequence flow.

When an exclusive gateway is used where two or more sequence flows merge, the gateway does not impose synchronization or evaluate any sequential flow expression, it simply continues the execution to proceed to the next task after the gateway.

Forking

Gateway activation leads to the activation of exactly one out of the set of outgoing branches. The branch is determined by the if-then-else conditional logic using the information available from the sequence flows. To determine the outgoing sequence flow, the conditions are evaluated in order. The first condition that evaluates to true determines the sequence flow the token is sent to. The evaluation ceases at that point. If none of the conditions evaluates to true, the default sequence flow is taken, and if a default flow has not been specified, an exception is thrown.

Parallel

Parallel gateways define multiple paths, or convergence and synchronization of multiple paths. Parallel gateways can be of two types: join and fork.

Joining

The joining type of parallel gateway synchronizes concurrent branches or threads of execution. The gateway is activated if it receives an execution token from every incoming sequence flow and the token is consumed.

To model this behavior a rule is required whose condition would require the incoming messages from each incoming sequence flow to be asserted to the Process Rete and each message context providing identification of the sequence path traveled and the last flow node it originated from.

If there are excess tokens at an incoming sequence flow, these tokens remain at this sequence flow after the execution of the gateway. This could mean that the out of context message has arrived and it needs to be correlated there it should also get asserted.

Forking

Parallel gateway forking is used to spawn concurrent threads or branches.

This involves creating a job hierarchy that is, parent job spawns child jobs and each job keeps progressing on the assigned sequence flow.

If the sequence flow activities have no outgoing sequence flow and there are no end events in the containing process or subprocess, the activities terminate and termination semantics for the container are applied.

The parent job can have multiple child jobs but a child job can have only one parent job. Each job can be identified by <parent job id>.<child job id>, and it can have its own job context. All the concurrent jobs are scheduled to be executed immediately after the gateway synchronization in the rule action.

Project Configuration Prerequisite to Support Processes

At a minimum, the project must provide some configuration options to support processes.

- Cache object management
- Backing store
- Content-aware load balancer
- JMS channel and destination

See *TIBCO BusinessEvents Developer's Guide* for details on all of the configuration options. If you are new to TIBCO BusinessEvents, first gain familiarity with the core features using the *TIBCO BusinessEvents Getting Started* guide.

Project Resources Used in Process Elements

The TIBCO BusinessEvents process must provide the resources needed to support the functionality defined in the tasks of a process.

Ontology Items that Support Process Elements

Process Element	Project Resource
Script	Rule Function
Business Rule	Decision Table (Requires TIBCO BusinessEvents Decision Manager)
Inference	Rule
Send Message	Simple event
Receive Message	
Message Start	

Process Element	Project Resource
Message End	
Error End	


Adding a Process

For adding a process, the TIBCO BusinessEvents project resources required to support the process should be in place.

See TIBCO BusinessEvents Developer's Guide for details about creating project resources.

Procedure

1. In BusinessEvents Studio Explorer, right-click the folder, where you want to store the process, and select **New > Process**. The New Process Wizard is displayed.
2. In the **Process Name** field, type a name for the process. In the **Description** field, type a description.

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

3. Click **Finish**.

You see the Process Editor showing a simple process outline.

4. Click any blank part of the process canvas so that the process-level property sheets are visible, then complete them as follows:
 - In the **General** tab, enter a label and author, and specify if the process type is Public or Private.
 - In the **Documentation** tab, use the rich text editor to add documentation about the process.
 - In the **Variables** tab, add all the process variables needed for data storage. For example, data from incoming events is copied to the process variables for use in a later activity, for example by a rule function.
5. Place a **Message Start** event on the canvas and configure it as follows:
 - In the **General** tab Resource field, specify the simple event or time event that triggers the process to execute.
 - In the **Documentation** tab, use the rich text editor to add documentation about

the event.

- In the **Output map**, map the properties of the event you selected in the **General** tab to process variables. Map event properties on the left to process (job) variables on the right.
6. Click **OK**.
 7. Do any of the following as per the requirement of the process:
 - Place an activity on the canvas, such as a Script task and configure it. For example, specify loop characteristics, use the Input map to map data from process variables to the activity, use the Output map to map data from the activity (that is, values after processing has occurred) to process variables, and so on, according to the available options for each type of activity. You can use the **Documentation** tab to add documentation about the activity.
 - Connect the items with sequence flows and Add gateways to control the flow. Configure expressions for sequence flows emerging from gateways, and select one as the default flow. Specify the type of sequence flow on the **General** tab: **Uncontrolled**, **Conditional**, or **Default**. If you select **Conditional**, define the expression as explained next. (See [Gateways](#) for details on gateways.)
 - If you are configuring a Conditional sequence flow out of a gateway, in the **Expression** tab, use the XPath Formula builder to define an expression that has to evaluate to true for the next activity to execute. In the **Data** tab of the Expression editor, you can use process variables in the formula. In the **Functions** tab, you can use a library of provided functions and use the **Constants** tab to select items such as Timezone formats, quotes, and other items.
 8. Add one or more End events, as needed to end the paths of the process.
 - In the **General** tab, select an event type as the output of the process (if you selected message end or signal end)
 - In the **Documentation** tab, use the rich text editor to document the task.
 - In the **Input** tab, map process variables as needed to the event that is sent out at the end of the process, if any event is sent.
 9. Save the process.

Startup and Shutdown Processes

The Startup and Shutdown processes are executed by the Process Agent at the start and shutdown of TIBCO BusinessEvents Engine respectively.

Startup Process

When the TIBCO BusinessEvents engine starts up, it executes the startup process if defined.

Shutdown Process

When the TIBCO BusinessEvents engine shuts down, it executes the shutdown process if defined.

Consider the following conditions for a process when you configure the startup and shutdown process:

- Startup and Shutdown processes must have a start event with none.
- Startup and Shutdown processes should not have message and signal start events. They are activated on call to the process and do not depend on an event to start.
- Startup and Shutdown processes can have none, message, and signal end events as they can notify the status to the external message end points.
- Startup and Shutdown processes cannot have any receive message task, rule task, and parallel join gateways.
- If a Startup or Shutdown process contains any Call Activity task, the referred process in the Call Activity task should meet the conditions of the startup and shutdown processes.

i Note: Startup or Shutdown processes must not have any task that forces a parallel thread of execution. They can contain only those tasks or events that allow execution within the same process thread.

Configuring Startup and Shutdown Process

Startup and Shutdown processes are configured for a process agent in the Cluster Deployment Descriptor (CDD) file.

Procedure

1. Open the CDD file.
2. In the **Agent Classes** tab, expand the process agent node for which the startup and shutdown processes must be added.
3. Select the Startup Processes node to add a startup process.
4. Click the **Add** button and select the process in the Select Processes dialog box. This process behaves as the startup process for a process agent.
5. Select the Shutdown Processes node to add a shutdown process.
6. Click the **Add** button and select the process in the Select Processes dialog box. This process behaves as the shutdown process for a process agent.
7. Save the CDD file.

Process Execution (Run Time)

The Life Cycle of the process execution involves creating process template class from the process definition and executing it on an event arrival.

The Life Cycle of the process execution has the following characteristics:

- Every process definition is first compiled and converted into a process template class.
- A process template is instantiated for this process definition.
- The process template contains the process graph, task, and sequence.
- Each task model from design time is created with a task ID.
- The process template registry is maintained.
- The process is executed.
- Upon the receipt of an event, which is a start trigger point for the process, an implicit rule is sent to find the start event and match it with the process template. After that, a job is created from the process and the output mapper executed.

Task Execution

The result of input mapping and task execution is mapped to the output mapper.

Backing Store and Checkpointing

Checkpointing persists the process data to the backing store and cache for use in recovery. At a checkpoint, the values of all process variables are saved to the backing store and cache.

TIBCO BusinessEvents Process Orchestration supports only Shared All as the persistence option and only cache-aside as the backing store write option. See *TIBCO BusinessEvents Architect's guide* for more information on backing store.

i Note: Process Orchestration does not support the Shared Nothing persistent option and the write-behind backing store write option.

All entities and handles need to be preloaded using the CDD options for preload.

The following configuration in CDD should be enabled:

- Object Management[Cache]/Domain Objects/Default/Cache Only
- Object Management[Cache]/Domain Objects/Default/Preload Entities
- Object Management[Cache]/Domain Objects/Default/Preload Handles

You can checkpoint individual tasks to provide more points of recovery. Checkpointing impacts performance due to the time it takes to persist the data. You should checkpoint only important tasks whose state must be persisted in the event of a shutdown.

Values of system variables (such as `id`, `_extid`, and `messageid`) are persisted, as well as the values of user variables (those you add to a project). Stale checkpoint values are automatically replaced by fresh ones as a process instance executes, so that the latest checkpointed states are used during recovery.

In case there are two process agents (with different priorities) and the **Max Active** field in the CDD file is set to the value “1”, the Process Orchestration software still performs the checkpointing and recovery automatically. If the process agent with the high priority is killed and process is still incomplete, the second process agent, with the low priority, becomes active and performs the recovery. The second process agent continues to execute the remaining process. Later when the high priority process agent starts again, it becomes active again and the second process agent, with low priority becomes inactive.

Checkpointing can be implicit or explicit.

Explicit Checkpoints

Explicit checkpoints are done when the user configures it at individual tasks. You can enable a checkpoint explicitly for the following tasks:

- Script task
- Business Rule
- Send Message
- Web Service

- Inference task

Implicit Checkpoints

The BusinessEvents engine creates the checkpoints for the job definitions using version. Additionally, job instance states are checkpointed at the following points of execution:

- Parallel Gateway - At gateways, checkpoints are made:
 - After all forks
 - Before all merges
- Receive task
- Call Activity task
- Subprocess task.

Recovery

When an engine restarts, only jobs that were active at the time of the shutdown are recovered. They are loaded from the backing store into memory.

Jobs that were passive during a shutdown remain in the backing store until activated, for example by a message arriving at an intermediate event, or on activation by a timer event.

Configuring Explicit Checkpoint for an Activity

Explicit checkpoints are done when the user configures it at individual tasks.

Procedure

1. Select the task on the canvas.
2. Click **Properties > General** tab.
3. Select the **Checkpoint** check box. This enables the checkpoint to persist the process data for this activity to a backing store and cache.
4. Save the project.

Locking and Unlocking

There is an implicit lock on the job when the process is executed.

The developers are expected to take a lock in following objects as needed:

- Parallel Gateway
- Job clustering

Checkpointing removes locking.

Exception Handling

The catalog function `setExceptionHandler` is used in the template class to handle the exception in Processes. You can set the exception handler to invoke the exception handler process or exception handler function in the case of an exception.

You can see more details about the function through the tooltip or the catalog functions in the *Online References* documentation.

Adding and Configuring Process Agent Class

Defining a process agent class is similar to defining an inference agent class. The process agent configuration is defined in the CDD file. In the CDD file, select the project resources that the agent uses at run time, and configure other settings.

Before you begin

Some familiarity with adding and configuring inference agent classes. Refer to *TIBCO BusinessEvents Developer's Guide* for a more detailed procedure.

Procedure

1. In the **Agent Classes** tab, click **Add Agent**.
2. In the New Agent Class dialog, type an Agent Class name and select the Process Agent Class type from the list. Click **OK**.
3. The new agent class name appears on the left. Select the agent class name. Settings for the process agent type appear in the Configuration section.
 - Complete the settings as explained in [Table 5, CDD Agent Classes Tab Process Agent Settings](#). Add any properties as needed.

CDD Agent Classes Tab Process Agent Settings

Setting	Notes
Max Active	Specifies the maximum number of active agents of this class. This value is used for fault tolerance. Deployed agents that act as standbys can take over from active instances that fail. In many cases, there is no need to keep standby instances. A value of 0 indicates an unlimited number of active instances. Default is 0.

4. Add the resources you want to use in each of the collections. The collection types available are the same as for an inference agent class, with the addition of process definitions. Ensure that all project resources used in the process definitions are included. For a detailed procedure, see *TIBCO BusinessEvents Developer's Guide*.
5. If you added any individual destinations to the Input Destinations Collections

category, highlight their name on the left and configure their settings on the right. See TIBCO BusinessEvents Developer's Guide for details. (Destinations within input destination collections are configured at the **Collections** tab.)

6. Save the CDD file.

Running Process Agent

After the deployment configuration is defined for the Process Agent in the CDD file, run the process agent to execute the processes defined at design time.

You can run the process agent in the following ways:

- Run using TIBCO BusinessEvents Studio.
- Run at the Command Line.
- Deploy and run using TIBCO Administrator.

Building EAR file

After you have defined the deployment configuration for the Process Agent in the CDD file, you can build the project EAR file within the TIBCO BusinessEvents Studio software.

For a more detailed procedure for building and deploying EAR Files at the command line, see *TIBCO BusinessEvents Administration Guide*.

Procedure

1. Select the project.
2. Navigate to the menu bar, select **Project > Build Enterprise Archive**.
3. Type the value for the fields, in the Build Enterprise Archive wizard. For example, the **File Location** field specifies the location where the EAR file is built.
4. Click **OK**.
5. Verify that the EAR file is built at the location specified by the **File Location** field.

Decision Table Hot Deployment in The Business Rule Task

You can hot deploy the decision table for the process agent in Process Orchestration, when the decision table is used in the business rule task.

You can hot deploy the decision table in Process Orchestration in three different ways.

Hot Deploying Decision Table Using CDD

You can only hot deploy to an application that was enabled for hot deployment before it was deployed. For hot deployment, ensure that the **Hot Deploy** check box is selected for the process agent's Processing Unit of the project's CDD file. When enabled for hot deployment, the application listens for changes in the EAR file. When you replace an EAR file, TIBCO BusinessEvents detects the change and performs hot deployment at the next RTC cycle. Ensure that the modified EAR file has the same name and is placed in the same directory as the EAR file that was used to start the engine. See *TIBCO BusinessEvents Administration Guide* for more details.

Hot Deploying Decision Table Using WebStudio

Ensure that the project, for which you want to deploy the decision table, is in RMS. Now use the WebStudio to BuildAndDeploy the decision table class file for the process agent. See the *TIBCO BusinessEvents WebStudio User's Guide* for more details on hot deployment of the decision table.

Hot Deploying Decision Table Using JMX

Using studio tools you can generate the decision table classes with command line and deploy the classes using JMX. You need to do certain configuration before you can perform the generation and hot deployment. See *TIBCO BusinessEvents Decision Manager User's Guide* for more details on generating the decision table classes and hot deploying them using JMX.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the [TIBCO BusinessEvents® Process Orchestration Documentation](#) page.

- *TIBCO BusinessEvents® Process Orchestration Release Notes*
- *TIBCO BusinessEvents® Process Orchestration Installation*
- *TIBCO BusinessEvents® Process Orchestration Developer's Guide*

To directly access documentation for this product, double-click the following file:

`TIBCO_HOME/release_notes/TIB_businessevents-process_6.2.2_docinfo.html`

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

Other TIBCO Product Documentation

When working with TIBCO BusinessEvents® Process Orchestration, you may find it useful to read the documentation of the following TIBCO products:

- TIBCO BusinessEvents® Enterprise Edition: TIBCO BusinessEvents Process Orchestration software is an add-on product to TIBCO BusinessEvents® Enterprise

Edition software.

How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

Legal and Third-Party Notices

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 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 TIBCO logo, the TIBCO O logo, TIBCO BusinessEvents, and TIBCO BusinessEvents Process Orchestration are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

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 and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2012-2022. TIBCO Software Inc. All Rights Reserved.