



TIBCO Business Studio™ - BPM Implementation Guide

Software Release 4.3

April 2019

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.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

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, Two-Second Advantage, TIBCO Enterprise Message Service, TIBCO Business Studio and TIBCO ActiveMatrix are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or 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. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

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

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

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

This 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 © 2004-2019 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Figures	9
TIBCO Documentation and Support Services	10
Using Projects and Processes	11
Projects Assets and Project Organization	11
Distribution of Assets across Multiple Projects	12
TIBCO Business Studio Workspace Folder	12
Setting BPM as the Destination Environment	12
Creating or Obtaining a Project for BPM	13
Creating a Project Using the BPM Developer Project Wizard	13
Adding New Assets to an Existing Project	15
Referencing Other Projects	16
Deployment of a Project	16
Using Live Development	16
Pageflow Processes and Business Services	18
Create a Pageflow Process	18
Generate a Business Service	20
Trigger an Incoming Message Activity from a Business Service	21
Managing Work Using Organization Models	22
Using Organization Models in a Process	22
Creating or Obtaining an Organization Model	22
Organization Model Entities as Process Participants	23
Using Capabilities and Privileges in Allocating Work to Process Participants	23
Using System Actions for Processing Work	24
Deploying an Organization Model	25
Mapping Resources to the Organization Model	25
About Participants	25
Assigning Participants to a User Task	26
Using a Performer Data Field or Parameter to Dynamically Define a Participant	26
Dynamic Organization Identifier Mapping	28
Using a Participant Expression to Define a Participant	28
Using Organization Entities in Performer Data Field or Parameter	29
Defining How Work Will be Assigned to Users	29
Offering and Allocating Work	30
Distributing Work Within the Target Pool	31
Allocating a Work Item to a Member of an Offer Set	32
Using Resource Patterns to Control How Work is Assigned	33

Chained Execution	33
Separation of Duties	34
Retain Familiar	34
Piling	34
Using Forms with User Tasks	35
Creating a New Form	35
Creating a New Form for an Existing User Task	35
Creating a New Form Manually from the Project Explorer	36
Creating a Free-standing Form	36
Switching Back to the Default Form	37
Updating Forms with the Synchronization Wizard	38
Using Data Fields and Parameters with Process User Tasks	41
The Mode Property of User Task Parameters	42
Using Data Fields and Parameters	42
Data Types for Data Fields and Process Parameters	42
Using Presentation Channels to Display Tasks to Users	46
Identifying an Appropriate Presentation Channel	46
Viewing the Available Presentation Channels	47
Adding a Channel Type to the Default Channel	47
Adding a Presentation Channel	48
Editing Email Attributes	48
Editing Email Attributes at Workspace Level	48
Editing Email Attributes at Project Level	50
Sending an Email Message from a Process	52
Configuring Service Tasks to Send Email Messages from a Process	52
Defining an E-Mail Service Type from a Service Task	52
Setting up Dynamic Data Inputs to an Email Message	56
Example of Setting up Dynamic Data Inputs to an Email Message	57
Calling a Database From a Process	59
Defining and Using a Database Connection Profile	59
Creating a Database Connection Profile	60
Creating and Using a Local Copy of the Database Connection Profile	60
Working Online or Offline To Connect to the Database or to the Local Copy	61
Using a System Participant to Identify the Target Database	61
Creating a System Participant and Mapping it to a Target Database	61
Assigning the System Participant to the Database Service Task	62
Configuring a Service Task to Call a Database	62
Creating a SQL Query	62
Using SQL Query Builder	64

Changing the SQL Statement	65
Building a Query - A Simple Example	65
Testing the SQL Statement	66
Selecting a Stored Procedure	67
Manually Entering a Stored Procedure Name	67
Selecting a Stored Procedure From the Database	67
Updating a Stored Procedure Used in a Database Task Activity	67
Mapping Data Between the Process and the Database	68
Mapping Data Parameters	68
Mapping an Externally Referenced Class Attribute to a Database Parameter	69
Automatically Creating a Business Object Model to Store Returned Data	70
Mapping the Result Set	72
JDBC Driver Connection Details	72
Using Web Services	74
Web Service Definition Language (WSDL) Documents	74
WSDL Document Requirements	74
XSD Constructs	76
Using Service Registries	76
Importing a WSDL Document Into a Project	78
Calling a Web Service	79
How to Call a Web Service	79
Calling a Service on a Virtualization Binding (Contract First)	80
Calling a Service on a SOAP Binding (Contract First)	81
Calling a Service on a Virtualization Binding (Contract Last)	82
Calling a Service on a SOAP Binding (Contract Last)	82
Configuring a Task or Event to Call a Web Service	83
Invoking a One-Way Operation on a web service	83
Invoking a Request-Response Operation on a web service	84
Selecting the Web Service Operation to Invoke	84
Selecting an Operation From a WSDL That Exists in the Workspace	85
Importing a WSDL and Selecting an Operation from the WSDL	86
Generating a WSDL and Creating an Operation from your Process Data	87
Updating a Generated WSDL File	88
Using a System Participant to Identify the Web Service Endpoint	88
Configuring Security on an Outgoing Service Call	88
Defining Input and Output Data	89
Defining Input and Output Mappings	90
Creating a Mapping	90
Points to Note About Mappings	91

Using A Script to Define a Mapping	91
Automapping	91
Catching WSDL Fault Messages on a Request-Response Operation	93
Using a Catch Intermediate Error Event to Catch a Fault Message	93
Handling SOAP/JMS Message Timeouts on a Request-Response Operation	94
Deploying a Process That Calls a Web Service	95
Exposing a Web Service	97
Exposing a Service	98
Exposing a Service (Contract First)	98
Exposing a Service (Contract Last)	99
Configuring a Task or Event to Expose a Web Service	99
Exposing a One-Way Operation	100
Exposing a Request-Response Operation	100
Using the Default Generated Web Service Operation	101
Updating the Default Web Service Operation	102
Exposing Multiple Default Web Service Operations	102
Selecting an Alternative Web Service Operation to Expose	103
Selecting an Operation From a WSDL That Exists in the Workspace	104
Importing a WSDL and Selecting an Operation	105
Using a System Participant to Define the Endpoint Provided by the Web Service	105
Setting a Common Context Root for Web Service Endpoint URIs	106
Exposing the Web Service Operation as a REST Service	107
Defining Input and Output Data	108
Defining Input and Output Mappings	109
Creating a Mapping	109
Points to Note About Mappings	110
Using A Script to Define a Mapping	110
Throwing WSDL Fault Messages on a Request-Response Operation	111
Using an End Error Event to Throw a Fault Message	111
Deploying a Process That Exposes a Web Service	114
Arbitrary Length Tasks and Request-Response Operations	114
Handling a Process that Includes Arbitrary Length Tasks	115
Using a Process as a Service Provider and as a Service Consumer	117
Authenticating Access to an Exposed Service	118
Calling the Service from a SOA Application	119
Example 1 - Single Sign-on Using a Virtualization Binding	120
Example 2 - Single Sign-on Using a SOAP Binding	121
Example 3 - Impersonation Using a SOAP Binding	122
Calling a REST Service	123

Defining the Interface to an External REST Service	123
Creating JSON Schemas	126
Creating JSON Schemas From a JSON Sample	127
Configuring the Process Project from Which you Want to Call the REST Service	128
Defining Input and Output Mappings	128
Creating a Mapping	129
Configuring Security	129
Custom Policy Set	130
Fault Handling and Propagation	131
REST and Authentication	131
WSDL Change Considerations for Application Upgrade	132
Application Upgrade	132
Reverting to the Original Version of an Upgraded Application	132
Making Changes to the Service Interface	133
Changes that Do Not Change the Service Interface	133
What Changes the Interface Using the Contract Last Approach	134
Changes That Apply to Both Contract First and Contract Last Approaches	137
Development vs. Production	138
Using Scripts	139
Implementing Script Tasks	140
Unsupported Script Types	141
Scripts on Other Tasks	142
Supported Script Types	142
Sample Scripts	143
Adding an Action Script to a Task	144
Associating a Script with a Conditional Flow	144
Associating a Script with a Loop	145
Timer Event Scripts	146
Task Scripts on Events	147
Editing Scripts	147
Assistance for Action Scripts	149
Using Process Data as Script Variables	149
Using Structured Data Types	149
Dynamically Created Factory Methods	150
JavaScript Exclusions	151
Customizing JavaScript Presentation Preferences	152
Customizing XPath Presentation Preferences	153
Scripts at Runtime	155
Data Mapping	157

Array Mapping Strategies	157
Like Mapping	159
Mapping Contents in Data Mapper	160
Mapping Process and Work Manager JavaScript Class Attributes	162
Executing Java Classes from a Process	163
Complete the Parameter Mapping	164
Web Service Definition Language (WSDL) Documents	166
Abstract and Concrete WSDL Documents	166
WSDL Document Structure	166
Abstract WSDL Documents	166
Concrete WSDL Documents	167
Web Service Configuration Properties	169
Web Service Implementation Properties	169
System Participant Shared Resource Properties	171
SOAP over HTTP Binding Details (Provider)	171
Endpoint Address Construction	172
SOAP over JMS Binding Details (Provider)	173
SOAP Over HTTP Binding Details (Consumer)	176
SOAP Over JMS Binding Details (Consumer)	177
Generating a DAA from the Command Line using an Ant Task	183
Using External Tools to run an Ant task within TIBCO Business Studio	184
Resource Query Language	186
RQL Expression Evaluation	186
Best Practices When Using Resource Query Language (RQL)	187
RQL Structure	188
Keywords	188
Operators	189
Organization Entities	189
Navigating the Organization Model with RQL Queries	191
Using the . Operator	192
Using only children and all	193
Combining Expressions	193
RQL Cleanup Configuration	194

Figures

Allocating a work item to a single user from the pool	30
Offering a work item to every user in the pool	30
Problem Marker: Form out of sync	38
Problem Marker in the Project Explorer View	39
Problem Marker in the Outline View	39
Synchronization Quick Fixes	39
First Page of Synchronization Wizard	40
Preferences for TIBCO Forms Process Support	41
Data Fields at the Process Level	44
Data Fields at the Process Package Level	45
Using two one-way operations to handle the problem of arbitrary length tasks	116
Using correlation data to connect separate one-way operations	117
A Process Providing and Consuming Web Services	118

TIBCO Documentation and Support Services

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. To access the latest documentation, 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_business-studio-bpm-cloud-edition_version_docinfo.html`

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

- TIBCO Business Studio™ Concepts
- TIBCO Business Studio™ Modeling User's Guide
- TIBCO Business Studio™ - Analyst Edition User's Guide
- TIBCO Business Studio™ - BPM Implementation
- TIBCO Business Studio™ Forms User's Guide
- TIBCO Business Studio™ Simulation User's Guide
- TIBCO Business Studio™ Customization
- TIBCO Business Studio™ - Analyst Edition Installation
- TIBCO Business Studio™ - BPM Edition Installation
- TIBCO Business Studio™ iProcess to BPM Conversion

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- 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 <https://support.tibco.com>. 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 <https://community.tibco.com>.

Using Projects and Processes

This section describes what you need to know to enable projects for use with BPM.

Projects Assets and Project Organization

A project is a container for all the assets that may be required when deploying an application to BPM. Projects help to facilitate the sharing and organization of resources. Before modeling your business process or defining your organization model, you must create a project to contain your assets.

A project can contain the following assets:

- **Business Assets.** There are the following categories of project-related business assets in TIBCO Business Studio:
 - Quality process. Business cases, project plans, and so on.
 - Ad-hoc assets. Supporting documents, spreadsheets, and so on that are not part of the quality process.
- **Processes.** A process models the business process.
- **Forms.** You can define forms to collect user input in a user task within a business process.
- **Business Object Model.** A business object model is a set of business terms and relationships specific to your corporate environment (for example, in a financial environment, broker, counterparty, and so on).
- **Emulations.** You can add emulation files to emulate a process and add test data at certain activities, thus enabling you to check the data flow at various activities in a process.
- **Organization models.** An organization model defines the organizational structure of your enterprise and the relationships between the different components (for example, organization units and positions) within your organization.
- **Workforce management.** Workforce management stores your library of tasks. Having a library of tasks enables you to reuse tasks in different processes.
- **Service descriptors.** Service assets include the WDSL files for any web services that you import into your project.
- **TIBCO SOA platform.** If you want to deploy an application to the TIBCO SOA Platform, you must create special folders to contain the SOA assets. Refer to the appropriate BPM Composite Development Guide for more information about developing applications for use with the TIBCO SOA platform.

When creating a project for use with BPM, you must do the following:

- Decide what assets you want and how they should be distributed across projects. See [Distribution of Assets Across Multiple Projects](#).
- Create or obtain a project. See [Creating or Obtaining a Project for BPM](#).
- Create and develop the assets that the project must contain.
- Deploy the project to BPM for use at runtime. See [Deployment of a Project](#), and refer to *TIBCO ActiveMatrix BPM Deployment Guide*.



When creating projects and deciding what to put into a project remember that one project is the deployment package and you cannot deploy a subset of it. If you have a main process and dependent sub-processes (which are not used from other main processes as well) you can put them into the same project, but it is recommended that you put them into different xpdl files so users can work on it in parallel. If you have processes which you want to deploy separately, you need to have them in different projects.

Distribution of Assets across Multiple Projects

When creating a project for use with BPM, you may want to include some or all of the assets, depending on your requirements. The assets can be all in one project or split amongst multiple projects.

For information about assets, see [Projects Assets and Project Organization](#).

For example, if you are going to have several applications executing against one organization model, you should create one project for your organization model and spread your business processes across multiple projects.

This is because it makes the organization model and business processes easier to maintain. For example, you may have amendments to the organization model that only affects one business process. If the assets are all in separate projects, you need only make amendments and redeploy the projects that contain the organization model and the affected business process.



When creating an organization model in a separate project:

- Versioning is used to control the interaction of different organization models across multiple projects. See [Setting BPM as the Destination Environment](#).
- The projects that contain business processes that use the organization model must reference the project that contains the organization model. See [Referencing Other Projects](#).

TIBCO Business Studio Workspace Folder

TIBCO Business Studio stores your projects in a folder called *workspace*. When you start TIBCO Business Studio, the Workspace Launcher is displayed, asking you to specify a location for your workspace.

When you install hotfixes or upgrade TIBCO Business Studio, you should specify a new workspace location and migrate your existing projects from your old workspace to your new workspace, using **File > Import**. See *TIBCO Business Studio Modeling User's Guide* for more information about migrating projects.

This is because your workspace points to the target platform specific to your installation. This target platform is used when building applications, and if it's defined incorrectly, it results in validation and build errors. For example, **Problem with DAA generation for project test: The feature com.example.test cannot be fully resolved**.

Setting BPM as the Destination Environment

When creating a project for use with BPM you must select BPM as the destination environment. Selecting BPM as the destination environment enables TIBCO Business Studio to utilize special features.

For example, for processes, TIBCO Business Studio performs validation according to the selected destination environments. Other artifacts (for example, organization models) are not affected by project destination settings.

If you select BPM as a destination, the following validation is performed:

- The processes are validated for BPMN.
- The processes are validated against BPM.

Procedure

1. When creating a new project, select **BPM** in the Project part of the wizard (see [Creating or Obtaining a Project for BPM](#))
2. For an existing project, do the following:
 - a) Enable the **Solution Design** capability from the toolbar.

- b) In **Project Explorer**, select the process.
- c) On the **Destinations** tab of the **Properties** view, select **BPM** as the destination environment for this process.

Creating or Obtaining a Project for BPM

You can create a project for use with BPM, either creating a new project, or importing an existing project.

- To create a new project, see [Creating a Project Using the BPM Developer Project Wizard](#).
- To import an existing project, see *TIBCO Business Studio Modeling User's Guide*.

You can select several default projects when creating a new project. When creating a project for use with BPM, TIBCO recommend that you select either an Analysis or BPM Developer project. The Analysis and BPM Developer projects are each created with different combinations of assets. However, you can add new assets to an existing project whenever required, see [Adding New Assets to an Existing Project](#). The following table describes the Analysis and BPM Developer projects:

Project	Default Assets	Description
Analysis	<p>The following assets are created by default:</p> <ul style="list-style-type: none"> • Business Assets • Business Processes • Forms • Business Object Model • Organization Model • Task Libraries 	<p>Select this project if you want to describe the business and its process, or an aspect of them, using business terminology. This may be useful if, for example, you want to create a project that contains plain English indications of what a script is intended to achieve, rather than JavaScript. See the <i>TIBCO Business Studio Modeling User's Guide</i> for a description of using an analysis project to model a business.</p>
BPM Developer	<p>The following assets are created by default:</p> <ul style="list-style-type: none"> • Business Processes • Forms 	<p>Select this project if you want to create a new project from the start without using an explicit analysis stage. See Creating a Project Using the BPM Developer Project Wizard for details of creating a BPM Developer project.</p>

You can create a new project from the **File > New** menu option.

Creating a Project Using the BPM Developer Project Wizard

Procedure

1. Select **File > New > BPM Developer Project**.
2. The Project dialog of the wizard is displayed.

The information that you enter in this dialog, and the subsequent dialogs of this wizard, is mostly the same as for the Analysis Project wizard. See *TIBCO Business Studio Modeling User's Guide* for a description.

Note that:

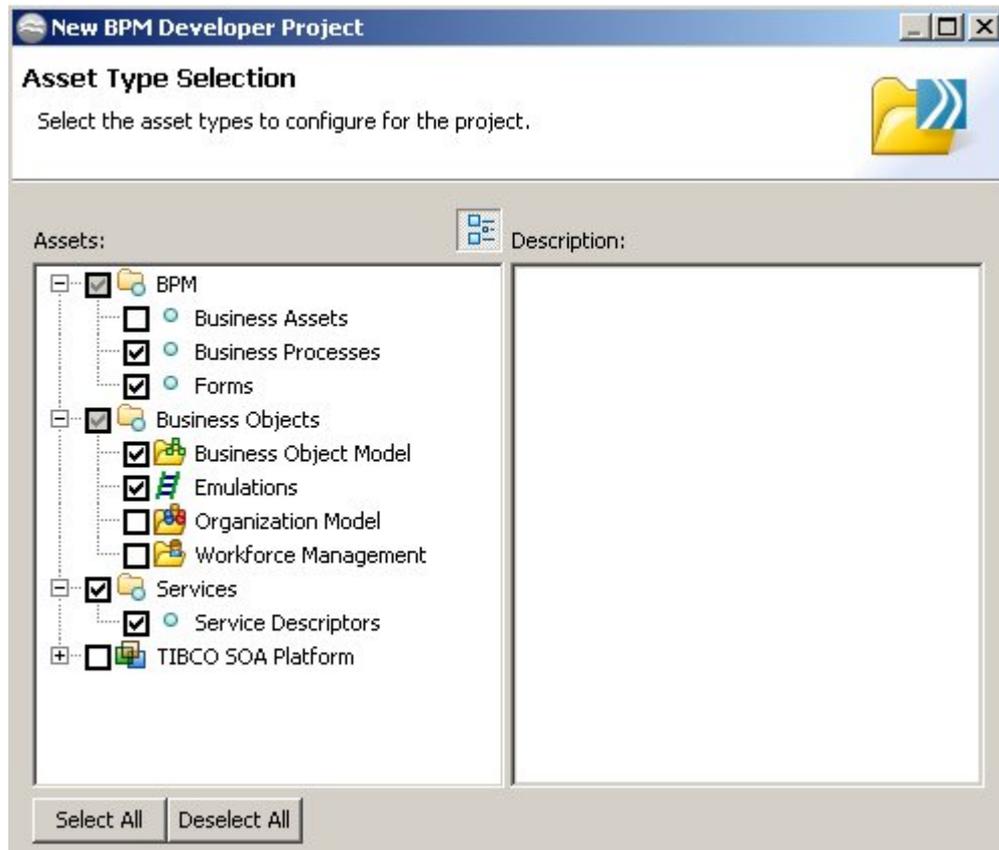
- The **Id** field specifies a unique Id for the project. This defaults to **com.example.projectname**, with *projectname* being a lower-case version of the name you entered in the **Project name** field. You can either accept the default Id or enter a new one. However, project Ids must be unique in Workspace at runtime. If you copy a project, the Id is also copied and a validation error is generated in TIBCO Business Studio. See the *TIBCO Business Studio Modeling Guide* for information on how to fix this.
- The **Version** field specifies the version of the project. Either accept the default version (**1.0.0.qualifier**) or enter a version for the project in the standard Eclipse format:
major.minor.micro



If you enter a qualifier it is ignored, as the qualifier is a date and time stamp that gets created when the project is deployed.

The specified version becomes the default for project artifacts such as process packages and organization models, and can be used to track revisions to the project. The version can be changed later. See [Setting BPM as the Destination Environment](#) for more information about versioning.

3. Set **BPM** as the destination environment.
4. The Asset Type Selection dialog displays the types of assets that you can include in your project.



The illustration shows which assets are selected by default. The assets that you select determine which dialogs are displayed subsequently. Select the types of assets you want to include in your project and click **Finish** when you are done, or **Next** to specify more options.

5. Complete the dialogs relating to your selected assets as required.

The newly-created package, process, and project are displayed in the Project Explorer.

Adding New Assets to an Existing Project

After you have created a project, you can add new assets to the project whenever required. Assets are stored in special folders. Storing assets in special folders means that you can utilize special features of the Project Explorer.

See *TIBCO Business Studio Modeling User's Guide* for a list of all the special folders.

Procedure

1. In the project where you want to add the new asset, create a new folder for the asset.
2. Right-click the new folder and select **Special Folders > Use as > asset > Folder** where *asset* is the asset you want to store in the folder.
3. Select the special folder you have just created. Select **File > New** and create the new asset that you want to add.

Referencing Other Projects

Projects can refer to other projects. You can invoke a process or use an organization model or BOM in another project.

For example, if you have an organization model that is used by processes in several different projects, you should create a separate project to contain the organization model and reference that project from the process projects that use it.

See *TIBCO Business Studio Modeling Guide* to find out how to create project references.



When you reference one project from another, it is important that their major version numbers match. See [Setting BPM as the Destination Environment](#).

Deployment of a Project

After you have developed a BPM application, its constituent elements must be deployed to the BPM runtime so that the application can be run.

Refer to *TIBCO ActiveMatrix BPM Deployment Guide* for detailed information on deployment.

The following elements must be deployed:

- The process (or processes, including both business processes and pageflow processes)
- Any organization model used by the process
- Any forms used by the process
- Any structured data used by the process

Using Live Development

Use Live Development when you want to make quick changes to elements of your project, and test the results immediately without having to redeploy an entire project.



This is especially relevant to Forms development, where small changes require a rapid turnaround for retesting.



Modifications to assets like process, BOM, and organization model are not supported in BPM Live Dev mode. If you change them, the project needs to be redeployed. If you change the Forms data interface, the project must be redeployed or the form will not open.

Procedure

1. Deploy the entire BPM application in the default **BPM Modeling** (or **Modeling**) perspective (if not already deployed).
2. Switch to the **BPM Live Dev** perspective (select **Open Perspective > BPM Live Dev** in the top right of the pane).
3. Log in to Openspace (in the embedded view, or open an external browser).
Any BPM server can be used, not just the Local Development Server. (For more information about the Local Development Server, see *TIBCO Business Studio BPM Implementation*.)
You can edit the information in **Openspace View Connection** for **Openspace/Client Base URL** and then refresh the view for the development server.
The Openspace view provides a **Copy openspace url** button that copies the openspace url, which can be pasted into an external browser to login to openspace in **Live Dev** mode.
4. Start a process and proceed until the required work item arrives (or start a business service and progress to the appropriate form).

5. Access a form on opening a work item/business service. The form from your design-time workspace is used instead of the deployed one.
6. Iteratively:
 - Test the form.
 - Edit the form in the local workspace and click **Save**.
 - Either use the **Refresh** button provided on the form which reloads the form with the latest changes without the need to re-open it or restart the pageflow for instance, or close the work item or business service in Openspace or browser. A **Cancel** button is also provided on the form in cases where Form loading fails. The **Cancel** button cancels the form if the form fails to load.
 - Reopen the work item / business service in Openspace or browser.

The process flow works as normal and you can complete/edit data in the same way as in normal Openspace.

Pageflow Processes and Business Services

Pageflow processes and business services are short-lived processes designed to display user interface pages to desktop and mobile users.

It is only possible to publish a business service, not a pageflow process. A business service can start a business process. A pageflow process requires a trigger from a task within a business process. TIBCO Business Studio includes a **Publish** option for business services to set the **Target Device** to either **Desktop** or **Mobile**.

TIBCO Mobilespace is an app that is available from the Apple App Store, and Google Play Store. Mobile users who use Mobilespace can access business services that have the **Target Device** set to **Mobile**.

Create a Pageflow Process

All tasks that are available in a business process are available within a pageflow process with the exception of a *business process user task*. Pageflow processes have a special variant of a business process user task that does not have participants (because the participant is implied by the person who opens the work item), and does not generate work items. These are referred to as *pageflow user tasks*.

A pageflow process is stored under the **Processes** branch of the Project Explorer alongside business processes.



To create a Process and its containing package and Project in one operation, see "Projects, Packages, and Processes" in *TIBCO Business Studio Modeling User's Guide*.



To convert a business process to a pageflow process, right-click the business process in Project Explorer, and then click **Convert To Pageflow Process**. Alternatively, a pageflow process can be converted to a business process in a similar way (**Convert To Business Process**).

Procedure

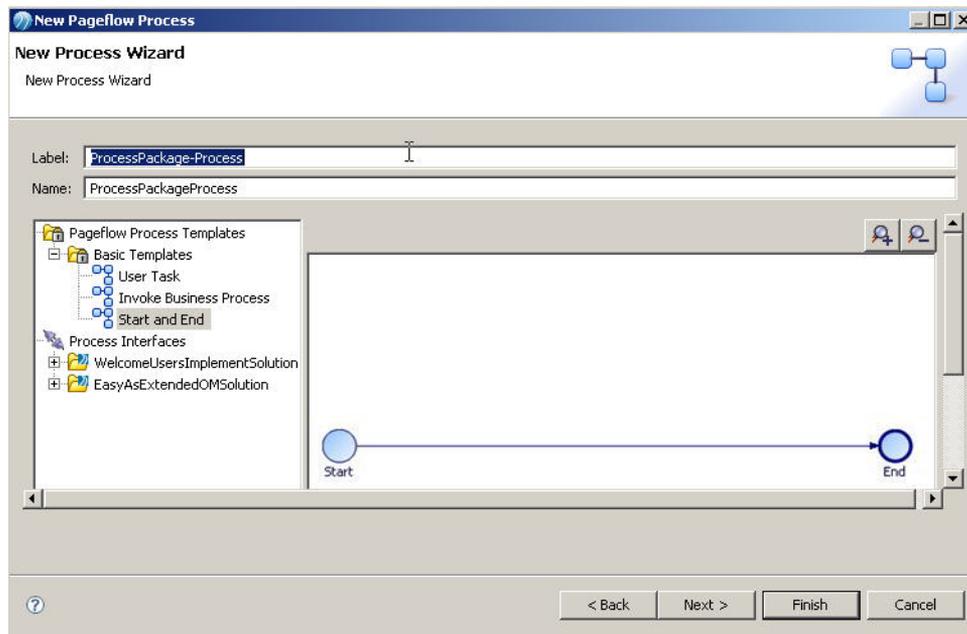
1. In the Project Explorer, under Process Packages, select the package you created, or Processes within an existing package, right-click and select **New > Pageflow Process**.
2. The **New Pageflow Process** wizard is displayed.



If you start this wizard from the **File > New** menu, the first dialog is Project and Package, where you must specify a valid project and package. This dialog is not displayed if you right-click at the package level to start the dialog; however, you can click **Back** to display it if necessary.

3. Enter the **Label** of the process. If you want to use a template to create the process, select the template and click **Next**.

In addition to the process templates, you can select a process interface as the basis for your new pageflow process. This creates a process with the necessary events and parameters that are specified in the process interface.



4. In the Description dialog, add optional text that describes the process, an optional URL that links to documentation about the process, and click **Next**.



The **Documentation Url** field is intended for design-time collaboration; it is not displayed in the runtime environment.

5. In the Destinations dialog, select the **Destination Environment** (optional). This controls the validation that TIBCO Business Studio performs when you save the process:



- The exact destination environments that are displayed depend on the edition of TIBCO Business Studio that you have installed. Select **BPM** for deployment to BPM.
- The specific destination components that make up a destination environment can be viewed by selecting **Window > Preferences**, and selecting **Destinations**.

If you do not select a destination environment, basic BPMN validation will be performed.



To avoid error messages and warnings associated with modeling constructs that cannot be executed in the runtime environment, set the appropriate destination environment on the process.

You can change or select the destination environment after the Process has been completed on the **Destinations** tab of the Process Properties.

6. Click **Finish**.
7. The process that you created is displayed in the Process Editor:



When you first start the Process Editor, the palette (on the right side of the diagram) might be collapsed; if so, expand it. You can expand this window to fill your screen by double-clicking the title bar. A pageflow process has a different default color scheme from a business process.



Pageflows do not contain pools or lanes because they are short lived and do not span different parts of the organization.

Generate a Business Service

A business service can call a business process. A business service provides input data to the business process with parameters. A Send Task in the business service sends the parameters to a Start Event in a business process.

Prerequisites

- A business process with a Start Event, and any required tasks.
- The business process should have at least one input parameter.
- The Start Event must have a Trigger Type set to None or Message.

The generated business service will finish with a Task that calls the business process, and matches the Trigger Type for the Start Event:

- A Call Sub-Process matches a Trigger Type set to None.
- A Send Task matches a Trigger Type set to Message.

Decide which Target Device the published business service will have. Mobile users who use ActiveMatrix Mobilespace can only access business services for a Mobile Target Device.

Refer to the *How to Design a Simple WelcomeUsers Business Service* tutorial for a worked example of this process.

Procedure

1. Right-click the Start Event, and select **Generate > Business Service**.

A new business service opens, with a default **Label** similar to ProcessPackage-Process.

The business service consists of a Start Event, a User Task, and a final Task.

2. Rename the business service with a more descriptive name, so that users can easily identify it from the list of available business services.
3. Add any additional User Tasks, as required.
4. From the **General** tab, use **Select the privileges required to use this service** to specify all of the organization model privileges that a resource must have to be able to use this business service.

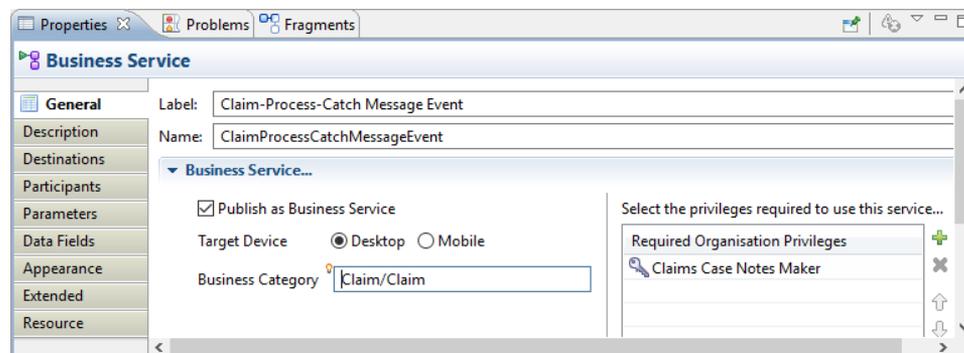


If there are no required privileges for the business service, then all resources use the business service.

5. Choose the **Target Device**:

- **Desktop**
- **Mobile**

Publish as Business Service is already checked, and the default is **Desktop**.



6. Press **CTRL + S** to save the changes.

Trigger an Incoming Message Activity from a Business Service

You can trigger any incoming message activity in a business process (that is, In-flow Receive Task and Intermediate Catch Message event, and Message Event handler).

The purpose of this business service is to collect the data needed to trigger the incoming message activity or event handler, and send the data to the business process.

Procedure

1. Right-click on the incoming message activity and select **Business Service > Generate**.
2. A new Business Service process is generated, using a default name similar to **ProcessPackage-Process**. You should rename this to a more descriptive name, to allow you to identify it easily in the list of Business Services, when you have deployed.

The business service process consists of a start event, a user task and a send task.

- The **User Task** will contain a datafield generated with the same name as the formal parameter defined in the business process.
- The **Send Task** calls the business process from which it is generated.



A system participant is automatically created for the send task that calls the business process. This participant has the name:

```
ProcessName_consumer
```

where: *ProcessName* is the name of the business process being called.

See [System Participant Shared Resource Properties](#) for more information about the configuration of this system participant.

3. Add additional user tasks to meet your requirements.



By the time that the Send Task is invoked, all parameters to the business process must have been collected. In the generated case, all are mandatory. On the user task, more realistically you might have several user tasks that all collect a proportion of the data.

4. A business service is generated.

When you deploy this process it will be presented as a business service in a list of business services. See "Business Services" in the *TIBCO Openspace User's Guide* or *TIBCO Workspace User's Guide* for more information.

Managing Work Using Organization Models

An organization is a collection of people, grouped and related to each other in different ways according to the needs of the enterprise. An organization model formalizes and defines the different elements of the enterprise organization (the organization's entities, their attributes and the relationships between them) that are available for use by a process.

These elements comprise:

- structural elements - organizations, organization units and positions.
- groups, which define the specification for a job of work to be performed, providing a *functional* view of the organization.
- descriptive elements - capabilities, privileges and locations, which provide additional information about other organizational elements, or about the resources (users) that belong to them.
- resources, which can represent items such as people, equipment or buildings.



Resources in the Organization Model are organization model entities that represent real users. To execute a process successfully, you must map real users to organization model entities using the *Organization Browser*. Once resources have been mapped, when a process is executed, BPM can translate a user task participant into the real user or users who should receive the corresponding work item, see [Mapping Resources to the Organization Model](#).

There are two Organization Browsers, one that is used if you are using the Openspace client, and another that is used if you are using a Workspace client (or custom WCC client). See the appropriate *TIBCO Organization Browser User's Guide* for your client.



See the *TIBCO Business Studio Modeling User's Guide* for more detailed information about organization model elements.

Using Organization Models in a Process

You can use organization model entities in a process: either as user task participants (at runtime, these entities will be converted into one or more real users who will receive the work items resulting from user tasks) or to make decisions about how to route work.

Procedure

1. Create or obtain an organization model. See [Creating or Obtaining an Organization Model](#).
2. Create process participants and map them to organization model entities. See [Using Organization Model Entities as Process Participants](#).
3. Deploy the organization model. See [Deploying an Organization Model](#).
4. Map resources (users) to the organization model. (This is not, however, a design-time activity.) See [Mapping Resources to the Organization Model](#).
5. Deploy the process. See "Deploying BPM Applications" in the *TIBCO ActiveMatrix BPM Deployment Guide*.
6. Run the process. See the *TIBCO Workspace User's Guide* for more information.

Creating or Obtaining an Organization Model

You can create or obtain an organization model to use with a process by creating it from scratch using the Organization Modeler or by refactoring a process that already uses **Basic Type** participants or by importing an existing organization model.

See the *TIBCO Business Studio Modeling User's Guide*.



- Creating an organization model from process data is useful for rapid development and prototyping of an application. However, for any significant work, TIBCO recommend that you create an external organization model from scratch.
- You can only use refactoring to create a new organization model. You cannot refactor to update an existing organization model (whether created by refactoring or not).

An organization model can exist in the same project as a process that uses it, or it can reside in a different project. In the latter case, the project containing the organization model should be selected as a **Project Reference** (see [Referencing Other Projects](#)).



An organization model can only be placed in a special folder marked as an **Organization Models Folder**. Organization Models folders are marked in Project Explorer with this icon

See the *TIBCO Business Studio Modeling User's Guide* for information about how to create the special folder.

Once the organization model has been created or imported, it can be used to define participants in the process. See [Assigning Participants to a User Task](#).

Importing an Existing Organization Model

You can either:

- import a complete project that contains an organization model, or
- import an organization model (.om) file into an **Organization Models** special folder in a project.



This folder must be a special folder defined as an **Organization Models** folder. If the folder is not already properly configured, right-click it and select **Special Folders > Use as Organization Models Folder**.

See the *TIBCO Business Studio Modeling User's Guide* for more information.

Organization Model Entities as Process Participants

Every user task in a process that is to be executed on BPM must have at least one participant. The participant can be defined in various ways.

The participant can be defined as the following:

- as a particular organizational entity in an organization model used by the project. See [Assigning Participants to a User Task](#) for more details.
- as an **organization model query**, a statement which uses Resource Query Language to create an expression that locates the required participant within the organization. See [Using a Participant Expression to Define a Participant](#) for details of how to do this.

To be available for use by a process, an organization model must be defined either in the same project as the process, or in a referenced project (see [Referencing Other Projects](#)).

Using Capabilities and Privileges in Allocating Work to Process Participants

Using capabilities and privileges allows you to allocate work at runtime.

For example,

- allocate a user task to an accountant if the value to be signed off is less than \$5000, but allocate it to an Accounts Manager if the value is \$5000 or more.
- allocate a user task to a single loss adjuster who holds at least level 2 motor insurance certification and is based in the Chicago office.

To allocate work to process participants using capabilities and privileges, you need to use an organization query. This allows you to enter a query using a script or expression. This is evaluated

when a referenced task is executed at runtime, so the actual participant is resolved and the activity dispatched and offered to the participant. A query could resolve to:

- a participant in the package/process. For example, you could define a capability called French speaker and then create a query that resolves a user task to any user who possesses the French speaker capability.
- an entity in the organizational model. For example, you could define a query that resolves a user task to Accounts Managers who can sign off values of \$5000 or more.

See [Assigning Participants to a User Task](#) for more information.

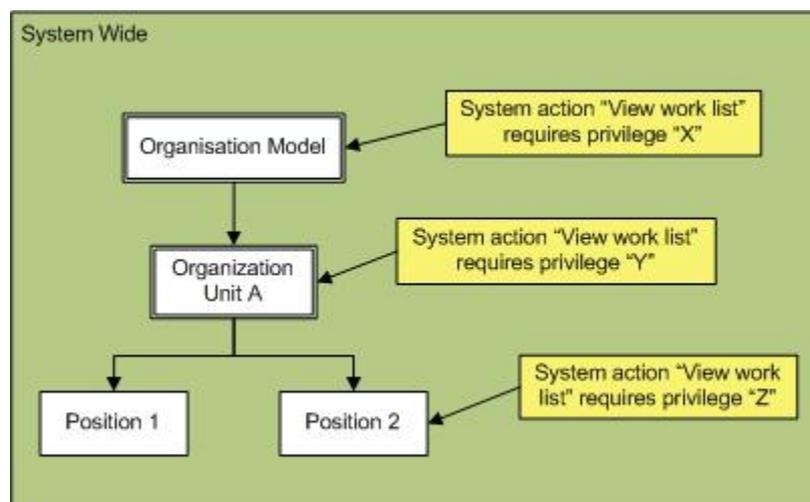
Using System Actions for Processing Work

System actions are actions that a user may wish to perform at runtime but that need to be authorized, or need to be restricted to users with a certain level of authority. These actions might include, for example, re-allocating work-items, skipping work-items, viewing another user's work list, or administering resources.

This authorization is implemented by associating system actions with privileges within Organization Modeler.

- Privileges can be assigned to any system action at the organization model level.
- Privileges can also be assigned to some system actions at the level of the organization unit, position or group.

For example, in the following diagram the "View work list" system action has been associated with three different privileges, "X", "Y" and "Z", at three different levels - the organization model, organization unit A, and position 2.



This means that a user must hold privilege "X", "Y" or "Z" to view the work list of a user who holds Position 2.

If a user wants to view the work list of a user who holds Position 1, they must hold privilege "X" or "Y". This is because no privilege has been associated with Position 1, so any privileges associated with the parent entity are used instead. If privilege "Y" had not been associated with Organization Unit A, the user would instead need privilege "X", defined in the parent Organization Model.



The **Organization Admin** system action allows users to see organization models other than the one they are part of, and allows them to view process instances started by members of these organizations.

As well as assigning different privileges at different levels, as shown above, qualifiers on the same privilege can be used to refine how access to a particular system action is controlled. (When comparing a required privilege to a held privilege, if either side is not qualified the comparison is positive. If both sides are qualified, the qualifications must match for the comparison to be positive.)

Controlling access to system actions by the application of (user-defined) privileges within the organization model provides an organization with a powerful and completely flexible way to customize and tailor users' access to system functions.

For more information:

- See the appropriate BPM Concepts Guide for an introduction to system actions.
- See the *TIBCO Business Studio Modeling User Guide* for a list of system actions that can be associated with privileges and how to assign them.

Deploying an Organization Model

To enable it to be used by a process at runtime or by Workspace to map users, an organization model must be deployed to BPM. You deploy an organization model by deploying the project that contains that organization model. A project may contain the organization model alone, or may also contain business objects and/or business processes.



When creating an organization model in a separate project, then:

- Versioning is used to control the interaction of different organization models across multiple projects. See [Setting BPM as the Destination Environment](#) for more information.
- The projects that contain business processes that use the organization model must reference the project that contains the organization model. See [Referencing Other Projects](#).

For instructions on deploying a project, see "Deploying BPM Application" in *TIBCO ActiveMatrix BPM Deployment*.

Mapping Resources to the Organization Model

Once you have deployed an organization model, you can then map resources - real users - to the model's groups and positions. You must do this before attempting to run any process that uses this organization model. Once resources have been mapped, when a process is executed, BPM can translate a user task participant into the real user or users who should receive the corresponding work item.

You map resources to the organization model using the *Organization Browser*. There are two Organization Browsers, one that is used if you are using the Openspace client, and another that is used if you are using a Workspace client (or custom WCC client). See the appropriate *TIBCO Organization Browser User's Guide* for your client.

About Participants

Participants are the entities assigned to carry out the tasks in a process.

BPM uses two types of participant:

- *user task participants* represent the users who perform the work defined in user tasks. These participants must be defined as *external references* in an organization model used by the process, not as *basic types*.



If you do not already have an organization model you can instead create participants as basic types and then refactor the process. This creates a simple organization model and converts the basic participants to external references in that organization model. See [Creating or Obtaining an Organization Model](#) for more information about this method.

- *system participants* are used to identify a task that is performed by the system.

When using participants as part of a process, you can:

- create a participant, see the *TIBCO Business Studio Modeling User's Guide*.
- define who will receive the work, by assigning one or more participants to a user task, see [Assigning Participants to a User Task](#).

- define how to distribute work to users, see [Defining How Work Will be Assigned to Users](#).
- delete a participant, see the *TIBCO Business Studio Modeling User's Guide*.

Assigning Participants to a User Task

Participants define who will receive work items generated from a user task.

A user task must have at least one participant, and can have as many as are required by the process. (A validation error is flagged on a user task that does not have at least one participant defined.)

Participants can only be assigned to a user task if they are defined in the process package (at process or package level) and are valid entities in an organizational model used by the process - see [About Participants](#).

You can use a combination of the following methods to define a participant for a user task:

- selecting a named participant. See the *TIBCO Business Studio Modeling User's Guide*.
- using a performer data field or parameter. See [Using a Performer Data Field or Parameter to Dynamically Define a Participant](#)
- using a participant expression. See [Using a Participant Expression to Define a Participant](#).
- using organization entities in performer data fields/parameters to allow you to deliver work dynamically to an organizational entity

Using a Performer Data Field or Parameter to Dynamically Define a Participant

A performer data field/parameter is a special type of data field/parameter that you can select as a participant for a user task. By assigning a value to the performer data field/parameter earlier in the process, you can dynamically define a participant for a user task. You can populate a performer data field/parameter with one or more organization entity GUIDs, or a single valid non-array RQL expression.



A parameter can only be defined at the process level.

Using a performer field, you can deliver work dynamically to an organizational entity, so that the work items appear in managed work lists. For example:

- Dynamically deliver to a group by name (e.g. using an organization entity GUID:
`Process.getOrgModel().groupByName('MyGroup')`)
- Dynamically deliver to a position within an organization unit by name (for example:
`orgunit(name='KEYTeam').position(name='AdditionalStaff') union
orgunit(name='Agency').position(name='Contractor')`)



You can also use non-RQL scripting to identify dynamic performers. For example:
`performer=Process.getOrgModel().getGroupByName('MyGroup')`.



If you use the presentation channel settings (push destinations) to deliver notification of work items via email, on the Work Resource tab for the user task, you must set the Distribution Strategy to **Allocate to One** rather than **Offer to All**. For example, if you have a performer field set to:
`resource(name='tibco-admin')`, `tibco-admin` will receive an email notification of a work item **only** if the Distribution Strategy is **Allocate to One**.

Procedure

1. Add the performer data field/parameter as a participant to the user task (using the method described in "Using Properties View to associate a participant with an activity" in the *TIBCO Business Studio Modeling Guide*).

2. Make sure that the process logic assigns a suitable value to the parameter before the user task is executed - for example, by using a script.

You can assign a script as described in "Creating Scripts" in the *TIBCO Business Studio Modeling Guide*.

You can populate a performer field with one of the following:

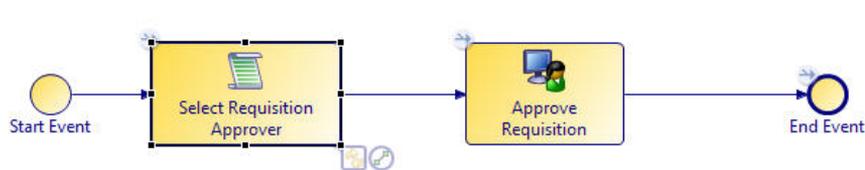
- an organization entity GUID
- multiple organization entity GUIDs (using a performer array field that contains multiple GUIDs - each of the performer fields in the array contains an organization entity GUID)
- a single valid non-array RQL expression (that resolves to one or more organization entities). See [Resource Query Language](#).
- a resource GUID. For example:

```
performer = Process.getOrgModel().resourceByName('JohnSmith').getGuid();
```



Using an organization entity GUID or multiple organization entity GUIDs means that dynamic performers allow references to specific organizational entities, so that they will appear in the relevant managed work lists. You should only need to use RQL if you want to offer work based on capabilities, privileges or intersections of organization entities. In most cases, writing a script to identify the organization entity GUIDs you need and populating the performer fields with these will be much more efficient than using RQL.

For example, the annotated process extract below shows part of a purchasing process that contains a user task to approve a requisition. The process requires that if the requisition value is less than \$50000, this task can be performed by an Accountant. If it is more than or equal to \$50000, it must be performed by the Finance Manager.



The screenshot shows the configuration for the 'Select Requisition Approver' script task. The 'Label' is 'Select Requisition Approver' and the 'Name' is 'SelectRequisitionApprover'. The 'Activity Type' is 'Script Task'. The 'Script Defined As' is 'JavaScript'. The 'Describe Task Script' field contains the following JavaScript code:

```

if (Req_value < 50000)
{
  Req_approver = "position(name='Accountant')";
}
else
{
  Req_approver = "position(name='Finance_Manager')";
}
  
```

The Select Requisition Approver script task assigns a value to the Req_approver performer data field, based on the value of the requisition which is held in Req_value (and which we assume to have been defined earlier in the process).

The Approve Requisition task assigns the work item to the Req_approver performer data field - the value of which will be either "Accountant" or "Finance_Manager" (both of which are also defined as participants for the process).

What to do next

Using a performer field, you can deliver work dynamically to an organizational entity, so that the work items appear in managed work lists.

Dynamic Organization Identifier Mapping

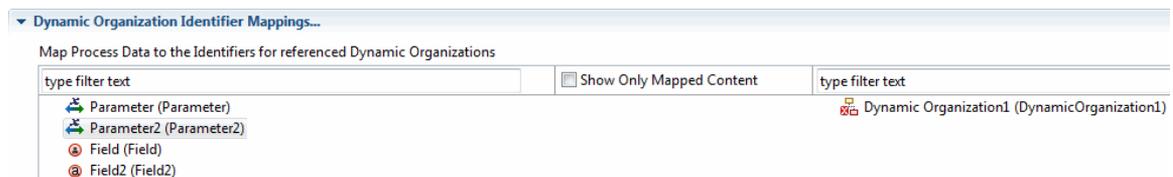
When a Dynamic Organization Participant is assigned to a task you need to identify the correct instance of the Dynamic Organization to use to resolve this participant at runtime. This is done using Dynamic Organization Identifiers which are mapped to process data.



The mappings are between the process data and the Dynamic Organization Identifiers of the referenced Dynamic Organization (**not** the Dynamic Organization).

Procedure

1. In the business process, select the **Work Resource** tab, and expand **Dynamic Organization Identifier Mappings...**



2. Map your process data (data fields and parameters) to a Dynamic Organization Identifier (which you set up when you created the Dynamic Organization Model). See "Dynamic Organizations" in *TIBCO Business Studio Concepts* and "Creating a Dynamic Organization" in *TIBCO Business Studio Modeling User's Guide*.

What to do next

A Dynamic Organization declares its Identifier Fields. These are arbitrary fields that are used to uniquely identify a generated instance of the Dynamic Organization at runtime.

When a Dynamic Organization is assigned to an Extension Point, those Identifier Fields must be mapped/assigned to named LDAP Attributes. This is done after deployment.

A generated instance of a Dynamic Organization takes its Identifier values from those named Attributes; of the LDAP Entry from which it originates.

A Dynamic Organization Participant carries values for the Dynamic Organization Identifiers. These values are derived from process data (data fields and parameters) mapped to those Dynamic Organization Identifiers. With this information the User Task can identify the instance of the Dynamic Organization in which the Participant can be found.

Using a Participant Expression to Define a Participant

You can use the **Organization Model Query** option to create an *expression* to define a participant. The expression sets out a definition that the participant in question must meet. This defines the participant in terms of organization model entities.

For example, this enables you to:

- allocate a work item to the manager of the person who carried out a particular named task.
- allocate a work item to one named position if the value of the data field x is >2000 , and allocate it to a different named position otherwise.

Participant expressions enable dynamic definitions of participants, since the participant who fits the criteria on one occasion may not be the same on another occasion.

Expressions can be used to describe concepts such as the following:

- The manager of a given organization unit
- A member of a given organization unit
- A given group
- The manager approving a given task



A push destination assigned to an organization entity (group, position, organization unit, etc.) will only work when the organization entity is explicitly identified as the participant, and not when it is defined as part of a participant reference.

These expressions can be made up of references to organization model entities:

- Lists of all members of a given organization unit
- Lists of all members in a named position in a given organization unit
- Lists of all members of a given named group
- List of all resources in a named position
- A specific named resource

For instance you can select a group named "HealthSafety" this way:

```
group(name="HealthSafety")
```

See the *TIBCO Business Studio Modeling User's Guide* to find out how to assign expressions to participants.

Using Organization Entities in Performer Data Field or Parameter

You can use organization entities in performer data fields or parameters to allow you to deliver work dynamically to an organizational entity, so that the work items appear in managed worklists.

You can do this by populating a performer field with one of the following:

- an organization entity GUID
- multiple organization entity GUIDs (using a performer array field that contains multiple GUIDs)



A Process Task can reference multiple Participants. However, if any of those Participants are Dynamic Organisation Participants, then all references to Dynamic Organisation Participants by that Process Task must refer to entities within the same Organisation Model Template.

- a valid DRQL expression that resolves to one or more organization entities. See [Best Practices when using Dynamic Resource Query Language \(DRQL\)](#)

Procedure

- Add the performer data field/parameter as a participant to the user task (using the method described in the *TIBCO Business Studio Modeling User's Guide*).

Defining How Work Will be Assigned to Users

TIBCO Business Studio offers a number of design-time mechanisms to refine and control how work items generated from user tasks will be assigned to users.

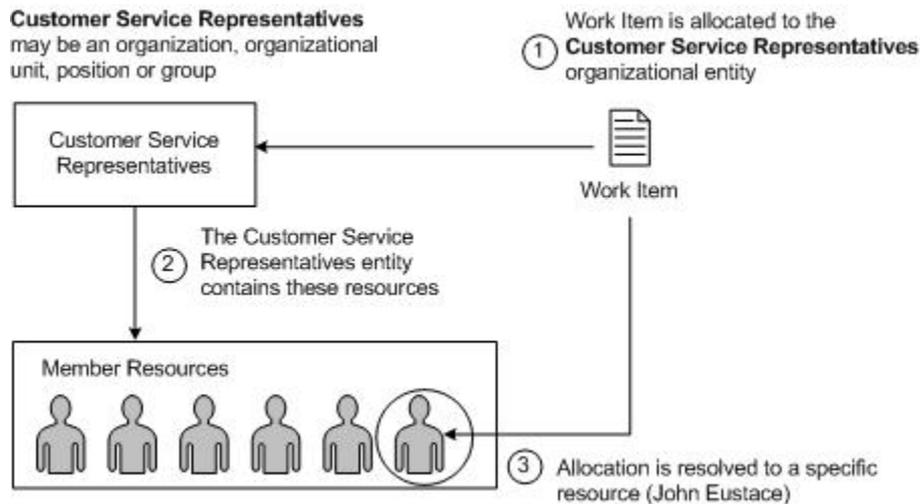
Offering and Allocating Work

The participants assigned to a user task define the pool of users who are eligible to receive work items generated by that user task. You must also define whether a work item generated from a user task will be *allocated* to or *offered* to the pool of eligible users:

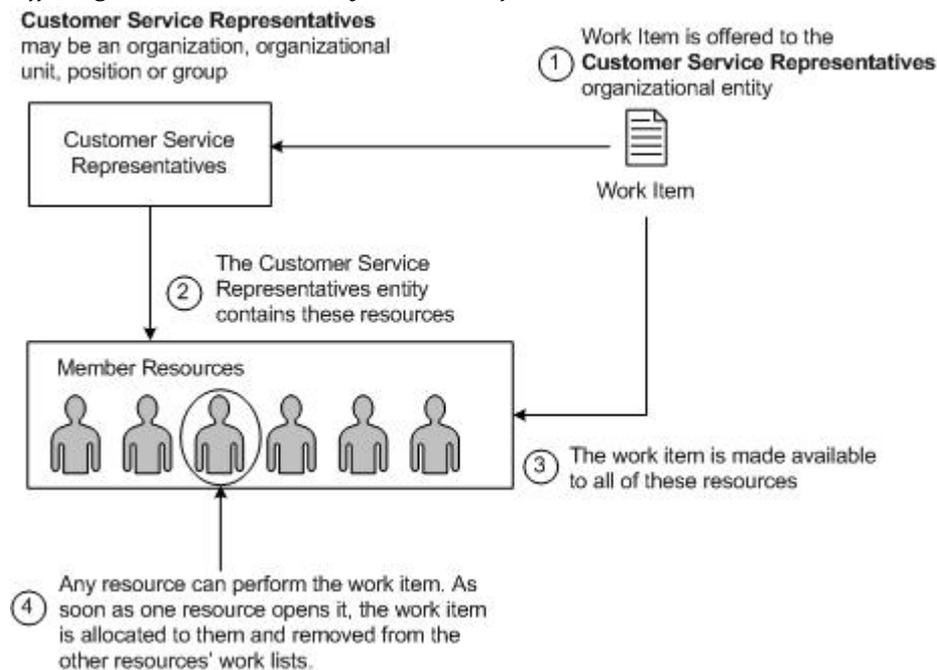
- **Allocated work** is only sent to a single user, selected from the pool.
- **Offered work** is sent to every user in the pool.

The following diagrams show the difference between these methods.

Allocating a work item to a single user from the pool



Offering a work item to every user in the pool



Procedure

1. Select the user task.
2. Select the **Resources** tab of the **Properties** view.



3. In the **Distribution Strategy** area, click:

- **Offer To All**, to offer a work item generated by this user task to all users in the pool of designated participants. This is the default option.
- **Offer To One**, to offer a work item generated by this user task to a single user in the pool of designated participants. **This option is not currently supported. Do not use it.** A validation error is displayed if this option is selected.
- **Allocate To One**, to allocate a work item generated by this user task to a single user in the pool of designated participants.

Distributing Work Within the Target Pool

Once you have identified the participants who are eligible to receive work items generated by a user task, BPM must identify the users within the target pool to send the work item to.

If the user task's specified distribution method is:

- **Offer.** BPM offers the work item to all users who are members of the specified organizational entity (or entities).
- **Allocate.** BPM allocates the work item to the pool of users that the participant definition resolves to. It determines which user to use by selecting an allocation method. These are:
 - **Round-robin.** Work items are allocated to members in strict rotational order.
 - **Random.** Work items are allocated to members in random order.

Assign the allocation method from the **General** tab of the **Properties** view of the requisite organizational entity. BPM uses the allocation method you have assigned. If that entity does not have an allocation method, the allocation method defaults to Random.

Work items can be assigned to an organizational entity (such as a position) using an RQL statement. RQL is dynamic, which means that if the items referred to by the RQL changes in some way (for example if the resources mapped to an organizational position are changed) then this will be reflected in the set of resources associated with the work item.

The work items assigned using an RQL statement do not appear in the Supervised Work List for the organizational entity (because the work items are offered to individual resources, not to the group being listed in the Supervised Work List). For example, if you define a Customer Services Representative position using resource query language, the work item is offered to each resource in the Customer Service Representative position but does **not** appear in the supervised work list.

If work items are offered to organization entities that contain no resources, the work item is scheduled, but will not appear in any worklists until a resource is mapped/added to the organization entity that the work item was offered to.

If work items are offered to organization entities that do not exist, the work item is scheduled to the undelivered group.

If you undeploy an organizational entity that is referenced by RQL and if work items are scheduled to that entity (or if work items are offered to organization entities which are deleted due to an organization model upgrade) then the work items transition to the pending state. To correct this the organizational entity must be redeployed. In this scenario an error will be logged and audited.

Allocating a Work Item to a Member of an Offer Set

Use the Allocate to offer-set member distribution strategy to specify both an offer set for a work item, and a specific user to whom that work item should be allocated. When the work item is scheduled, if that user is *not* a valid member of the offer set, the work item is instead offered to the remaining members of the offer set, *as if* the **Offer to all** distribution strategy had been used instead.

The **Allocate to offer-set member** strategy allows you to support, for example, a case handler/account manager pattern, so that although the work item is originally allocated to a member of a team, the team manager can still:

- see all items that were originally offered to the team.
- re-allocate the work item to another member if required - for example, if the user who started the case is off work due to sickness.
- report on work from a team perspective.

The **Allocate to offer-set member** distribution strategy cannot be used with the Chained Execution, Separation of Duties or Retain Familiar workflow patterns.

Procedure

1. Define a Performer field that will be used to identify the user to whom the work item should be allocated.

The Performer field must not be an array field and must not contain an initial value defined as an RQL expression. In either case, an error marker will be displayed against the field.

2. On the **Properties** tab of the User task, click **Work Resource > Distribution Strategy > Allocate To Offer-set Member**.
3. In the **Member Identifier** field, enter the name of the Performer field you defined earlier.
4. Modify the process so that it populates the Performer field with a suitable value before the user task is executed.

The performer field must, at runtime, contain the GUID of a single resource who is a valid member of the work item's offer set, as defined by the user task's participant.



You can use scripting methods and attributes to obtain the resource's GUID. See "Process Manager and Work Manager Scripting" in the *TIBCO ActiveMatrix® BPM Business Data Services Guide* for more information.

The performer field must not be populated by an RQL expression. (This will result in a runtime error.)

Result

At runtime, the work item will be allocated to the user identified by the GUID value in the performer field. If the performer field returns an invalid value:

- The work item will instead be offered to all members of the offer set.
- One or more of the following error messages will be returned (in the BPM log file and in relevant event or audit views).

Message ID of Error	Possible Causes
BRM_RESOURCE_INVALID_DESCRIBE_ENTITY	<p>The provided GUID belongs to an organizational entity that is not a resource.</p> <p>The performer field has been populated using an RQL expression (for example, <code>resource(name="Leon")</code>), so does not contain a GUID.</p>
BRM_WORKITEM_SCHEDULE_MEMBER_INVALID	The provided GUID belongs to an invalid resource.
BRM_WORKITEM_SCHEDULE_NOT_A_MEMBER	The provided GUID belongs to a resource who is not a member of the offer set.

Using Resource Patterns to Control How Work is Assigned

You can use the following resource patterns in your process to further control how work is assigned to users. Resource patterns are formal representations of the various ways in which resources are represented and utilized in workflows, as identified by the **Workflow Patterns initiative**.

- [Chained Execution](#)
- [Separation of Duties](#)
- [Retain Familiar](#)
- [Piling](#)

Chained Execution

Chained Execution allows you to specify that a user will automatically start the next work item in the same process instance as soon as the previous one has completed.

To set up a chained execution pattern, you refactor the required user tasks as an embedded sub-process, and mark the sub-process to use chained execution.

User tasks that are to be chained together must meet the following requirements:

- They can be assigned to different Participants, but they will still be offered to the same user *if* that user qualifies under the participant definitions for each task. For example, if one task is assigned to a position and the second to a group, a user who both holds that position and belongs to that group can be given both tasks. If the user is not a member of the group, the second task will be offered to each of the group members.

- They must have **Offer to All** as their **Distribution Strategy** (see [Offering and Allocating Work](#)).

For instructions on chained execution, see the information in the tutorial *How to Ensure that a Sequence of Tasks is Performed by the Same User* and the *TIBCO Business Studio Modeling User's Guide*.

For information on the properties that influence how a chained execution pattern operates at runtime, see the `WPProperties.properties` file in the "BPM Properties Files" section of the *TIBCO ActiveMatrix BPM - BPM Administration* guide.

Separation of Duties

Separation of Duties allows you to specify that particular user tasks must be performed by different *users*, even though the tasks are assigned to the same *participant*.

For instructions on implementing this pattern in a process, see the information in the tutorial *How to Ensure Specific Tasks in a Process are Performed by Different Users* and the *TIBCO Business Studio Modeling User's Guide*

Retain Familiar

Retain Familiar stipulates that you want a specific task to be executed by the same resource that executed a previous task in the same process instance. For example, the resource that handles the initial customer contact is the same one that handles the follow-up call.

For instructions on implementing this pattern in a process, see the *TIBCO Business Studio Modeling User's Guide*.

Piling

Piling allows you to specify that a particular participant completes work items that relate to a specific task, sequentially. The work items may be in different processes. Once the work item is completed, if another work item that corresponds to the same task is present in the user's work list, it is immediately started. The benefit of this is that the same work is given to a specific resource who then gains experience in processing this particular task.

For instructions on implementing this pattern in a process, see the *TIBCO Business Studio Modeling User's Guide*.

Using Forms with User Tasks

This section explains how to work with forms associated with user tasks and page flows within the TIBCO BPM environment. It provides information that is specific to using forms within a BPM environment.

Refer to the *TIBCO Forms User's Guide* for general information about designing forms.

This information is for users responsible for designing and implementing business processes that include user interfaces for presenting and capturing information from users. The forms and business processes you design can be deployed to a BPM node and can be accessed via Openspace, Workspace, or a custom client application.

Creating a New Form

When you create a BPM project, a default form is created for each user task defined in the business process. You can use the default form or if required edit the default form.

You can also use any of the methods listed below to create a new form for a user task:

- [Creating a New Form for an Existing User Task](#)
- Creating a New Form Manually from the Project Explorer
- [Creating a Free-standing Form](#)
- If required, [Switching Back to the Default Form](#)

Creating a New Form for an Existing User Task

For forms that are used by a business process, editing the default form for an existing user task is the most efficient way to create a form

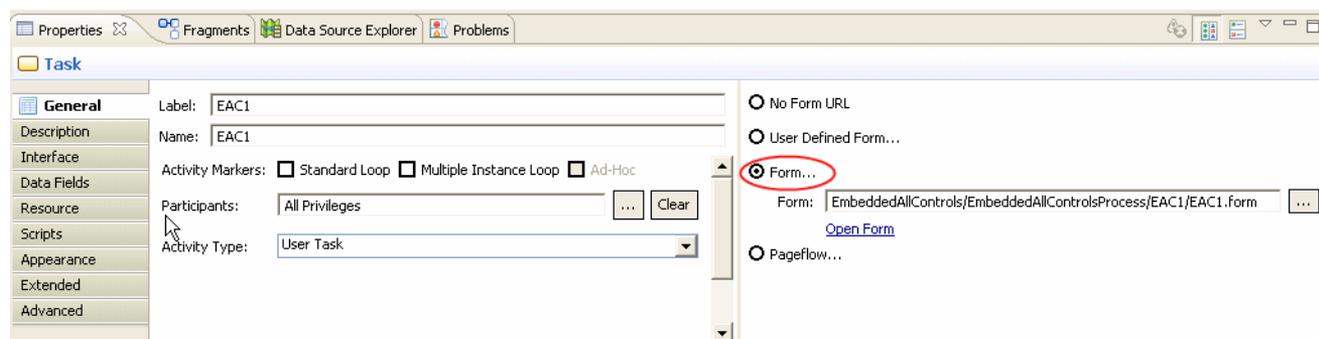
This is efficient for the following reasons:

- From the beginning, the default form exists in the context of the user task with which it will be associated in the business process. This means that the **Form** field for the user task (on the **General** tab of the user task's Properties view) will be automatically filled in with the URL of the new form.
- If there are existing parameters associated with the user task, the new form will include user interface components, bound to the corresponding form parameters.

You can create a new form for any user task in TIBCO Business Studio in one of the following ways:

- Go to the context menu of a user task in a business process and click **Form > Open**.
- On the **General** tab of a user task's Properties view, select the Form... radio button.

In both of these cases, the existing default form is set on the user task as the custom form. You are prompted with a warning message informing you that "the customized form will no longer be automatically kept in sync with the activity interface."



At this point, you will be using the automatically created default form as a starting point for further modifications. If there are any changes to the user task interface, those will need to be synchronized with the form. The form will display error-level problem markers when the parameters defined for the form are out of sync with the activity interface from which it was created.

Form Generator Preferences

TIBCO Business Studio provides two **Form Generator Preferences** settings that can be accessed from **Window > Preference > Form Designer > Generator** .

- **Generate text area for a string exceeding this length:** Used to specify the text length threshold value above which the form generator will generate a text area rather than a text input. The default value is 100.
- **Generate text area for an XSD string with no length restriction:** Used to generate a text area for a BOM primitive type or property that was originally defined in an XML schema as base type xsd:string with no length restriction.
 - Checked: A text area is generated.
 - Unchecked: A text input is generated.



You can set the **Form Generator Preference** both globally at the workspace level and override it locally at the project level. Local preference overrides must be applied to the project in which the form is generated, not the project containing the BOM.

Creating a New Form Manually from the Project Explorer

A form may be created manually from the Project Explorer within any **Forms** special folder.

Procedure

1. Go to the context menu of the **Forms** special folder, or any child folder under the Forms special folder in the Project Explorer and click **New > Form** . This will trigger the opening of the New Form dialog.
2. Specify the **Form type** on the New Form dialog. The type of form that is selected here determines the components that are initially part of the form model. The form types are as follows:
 - **Process task:** This creates a form that is the same as one created from a User Task in a process definition. It will contain a root pane, a toolbar with **Cancel**, **Close**, and **Submit** buttons, and a messages pane for displaying error messages.
 - **Pageflow task:** This creates a form that is the same as one created from a User Task in a Pageflow Process. The only difference to a Process task form is that the toolbar contains only **Cancel** and **Submit** buttons. The **Close** operation is not supported in pageflows since there is no way to re-open a step in a pageflow once it has been closed.
 - **Embeddable:** This creates a form that is suitable for embedding within another form. This will only contain a single root pane. This is because the parent form would typically contain the toolbar and messages pane, so these components are not needed in an embeddable form. Refer to the *TIBCO Forms User's Guide* for more information about Embeddable Forms.

Creating a Free-standing Form

Although it is convenient to generate new forms from existing user tasks, there are situations where a forms designer prefers to create a free-standing form before the corresponding user task is available.

There is no impediment to doing this. A form can be created by either of the methods listed below, and linked to a user task later.

- Go to the context menu of the **Forms** special folder, or any folder under the Forms special folder in the Project Explorer and click **New > Form** .
- On the **File** menu, click **New > Other > TIBCO Forms > Form** .

Procedure

1. Go to the context menu of the user task and click **Form > Use Existing** .

The Select Form dialog opens.

2. Select an existing form from the Project Explorer folder structure and click **OK**.

Note that you must choose a form within the same project as the user task.

3. Synchronize the user task parameters with the form by going to the context menu of the user task and clicking **Form > Synchronize...** This establishes a permanent link from the form to the user task. This also enables the form synchronization validator to check that the form remains in sync with the user task's data interface and for the form synchronization wizard to modify the form as necessary to preserve consistency.



The bidirectional linkage between the user task and form means that any given form may only be used by a single user task. The pairing of a user task to a form can be subject to automatic synchronization validation. If you share a form between multiple user tasks, you need to manually synchronize the form with the different user tasks. By doing this, the synchronization validation link to the previously associated user task is broken. Hence the second approach is not recommended. A better way to share forms is by embedding the reusable parts.



Forms must be created only under a Forms special folder, or within folders under a Forms special folder. If you choose a location other than the default location, be sure that the folder you select meets this requirement.

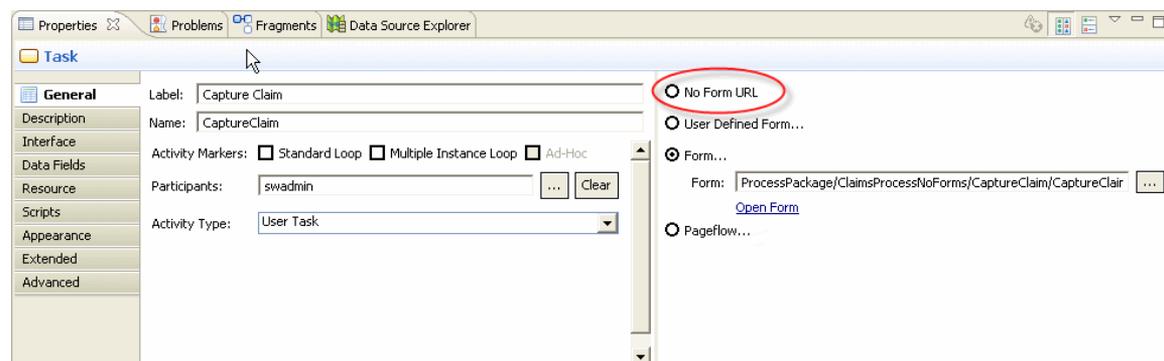
Switching Back to the Default Form

After creating an auto-generated form using the **Form > Open** context menu action or the user task General Properties tab **Form...** radio button and manually customizing it, there can be a situation where you would like to start over again with a default form.

Procedure

1. Go to the context menu of the user task and click **Form > Use Default** . This option is enabled only if the **Form...** radio button is selected on the user task General Properties tab.

You can also select the **No Form URL** radio button in the user task General properties tab as shown below.



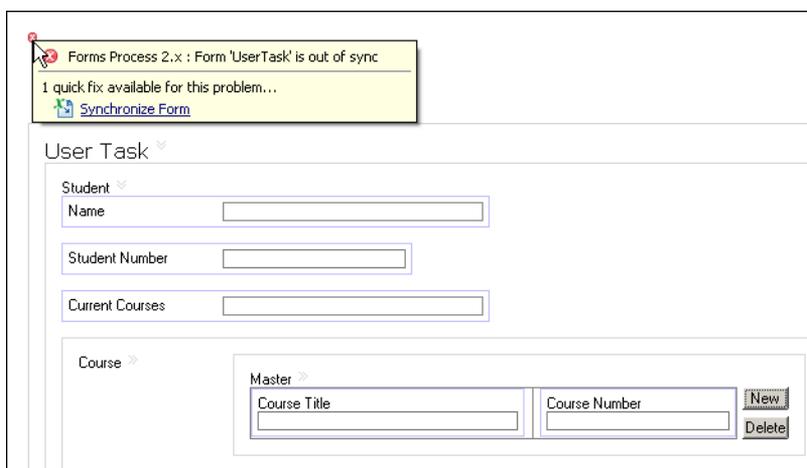
2. The Confirm Delete dialog appears.
 - Choose the **Delete** option if you need to delete the existing customized form. The customized form in the Forms folder in the Project Explorer is deleted.
 - Choose the **Keep** option if you need to retain the customized form. You would select this option if the Form were still to be used by other user tasks.
3. After providing the delete options, you need to generate the new form. Go to the context menu of the user task and click **Form > Open** . You can also click the '**Form...**' radio button in the user task General Properties tab.
4. If you previously specified the **Delete** option on the Confirm Delete dialog, a default auto-generated form is created.
5. If you previously specified the **Keep** option on the Confirm Delete dialog, the Overwrite Form dialog appears. Choose the **Overwrite** option if you need to overwrite the existing customized form with a default form or choose the **Reuse** option to continue using the existing customized form.
6. The system remembers your responses; the next occasion on which it presents either of these dialogs your previous choice becomes the 'default action' (the one executed if you press the Enter key). If you select the **Do not ask this question again** check-box, the dialog will not be displayed on subsequent occasions and the system will automatically take the action that you choose on this occasion. These settings may be changed using **Window > Preferences > Form Designer > Process** .

Updating Forms with the Synchronization Wizard

If the set of task parameters changes after a form has already been created, you can use the **Synchronize Form** wizard to update the set of form parameters, controls, and bindings in the linked form. Likewise, if a change is made in the business object model that impacts the form (for instance, if an attribute is added to a class that defines a type on the form), the synchronization wizard will update the form appropriately.

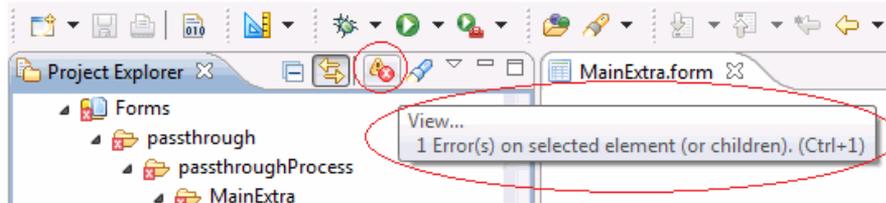
When a form needs to be synchronized, a problem marker appears on the form in the Form Designer, usually near the element that is out of sync, as well as in the Properties and Content Outline views. Mouse over the problem marker on the form to see a tool tip containing a description of the problem and a link to launch the synchronizer, as shown in [Problem Marker: Form out of sync](#).

Problem Marker: Form out of sync



You can also use the  button available in the Project Explorer view, Outline view, and Properties view. When the problem markers are in scope for the active view's input, this action is visible and can be clicked to inspect the markers and execute quick fixes.

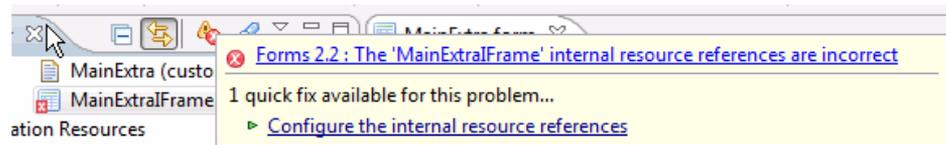
Problem Marker in the Project Explorer View



On clicking the  button, two hyperlinks are available:

- The marker link (opens the offending resource in its appropriate editor)
- The quick fix link (opens the Quick Fix wizard)

Problem Marker in the Outline View



If there are multiple markers, the tooltip window displays left and right arrow buttons to navigate the markers. If a marker issue has multiple quick fixes, they are all listed in the lower section.

Options for Synchronization

If multiple problem markers are associated with selected element on the form, the tool tip includes left and right arrows for navigating through them. Also, in the case of multiple synchronization problems, the tool tip can present different quick fixes for handling the problems.

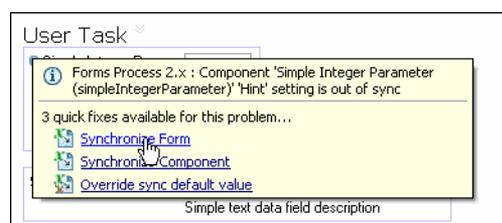
The first two quick fixes are: choose to synchronize the entire form, or synchronize just the component whose problem marker was used to launch the wizard. In the latter case, although fixes for all out-of-sync components will be displayed, *only* the component whose problem marker you "moused over" will have the checkboxes checked for its available synchronization actions, that is, selected for updating by the wizard.

There is a third option when multiple components are out of sync. You can choose to *override* the sync default value, that is, to leave the form as it is (perhaps because the problem is minor or because you intend to fix the user task parameter, instead, to bring the user task and the form into sync). This option will cause the wizard to ignore the suggested fix or fixes and remove the problem marker for the component whose marker was used to launch the wizard.

If the wizard is subsequently launched with the **Synchronize component** option using the problem marker of *another* component, fixes for the component for which the override was done will still be displayed, but will be grayed out and unselected to indicate the override. Nonetheless, the grayed-out fixes can be selected, undoing the earlier override. In this case, the wizard displays a warning message to the effect that user overrides will be overwritten with generator defaults.

Synchronization Quick Fixes shows an example of three possible synchronization quick fixes presented by the tool tip: synchronize form, synchronize component, and override sync default value.

Synchronization Quick Fixes

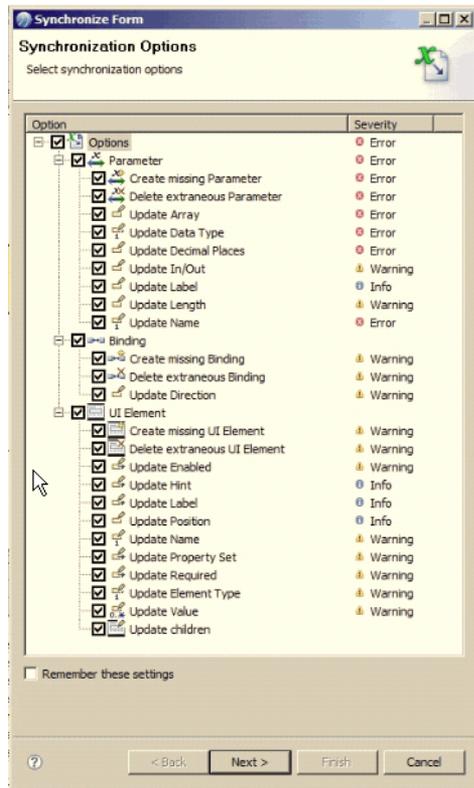


Click the desired link in the tool tip to launch the **Synchronize Form** wizard.

To launch the wizard directly, without using a tool tip, right-click the user task, and click **Form > Synchronize**. The menu for launching the wizard is shown in [#GUID-F42DD98A-A17F-4D9F-9E19-B52F7E369F04/ID-149-0000055C](#) on page 40.

The first screen of the wizard, shown in [First Page of Synchronization Wizard](#), presents a list of all the possible conditions on which the wizard can check and report, and, in each case, the severity associated with the item being out of sync. Normally, all of the checkboxes can be left checked.

First Page of Synchronization Wizard



If you check the **Remember these setting** checkbox, the synchronization options wizard will not start on page one in the future, **Synchronization Options**, but will take you straight to page two, **Synchronization Actions**. From page two, you can still reach page one by clicking the **Back** button.

Click **Next** to advance to the **Synchronization Summary** page, which displays the changes that can be made, such as parameters to be added, modified, or removed from the form. Clicking the Expand All (+) or Collapse All (-) button on this page will expand or collapse the list of actions displayed. The checkboxes enable you to select the actions the wizard will perform when you click the **Finish** button.

Actions performed by the wizard will respond to the **Undo** and **Redo** commands invoked on the **Edit** menu. Note that if a synchronization fix is undone, recreating the out-of-sync condition, the problem marker may not be displayed again until the synchronization wizard is invoked again.

Synchronization Wizard Preferences

The list of conditions that are checked by the wizard (shown on the wizard's first page), and the severity level indicated by the problem marker icon (and displayed in the tool tip) for each item, can be modified by editing the **Synchronization Options**. In the menu bar, click **Window > Preferences > Form Designer > Process** to view and edit the synchronization settings. You can also click **Window > Preferences > Form Designer > Process** to open the dialog and check or uncheck the checkboxes to remove or add an item to the list of items that will be checked when the wizard is run. Click the

severity for an item to display an option list for selecting a different severity. In most cases, the default severity choices are suitable.

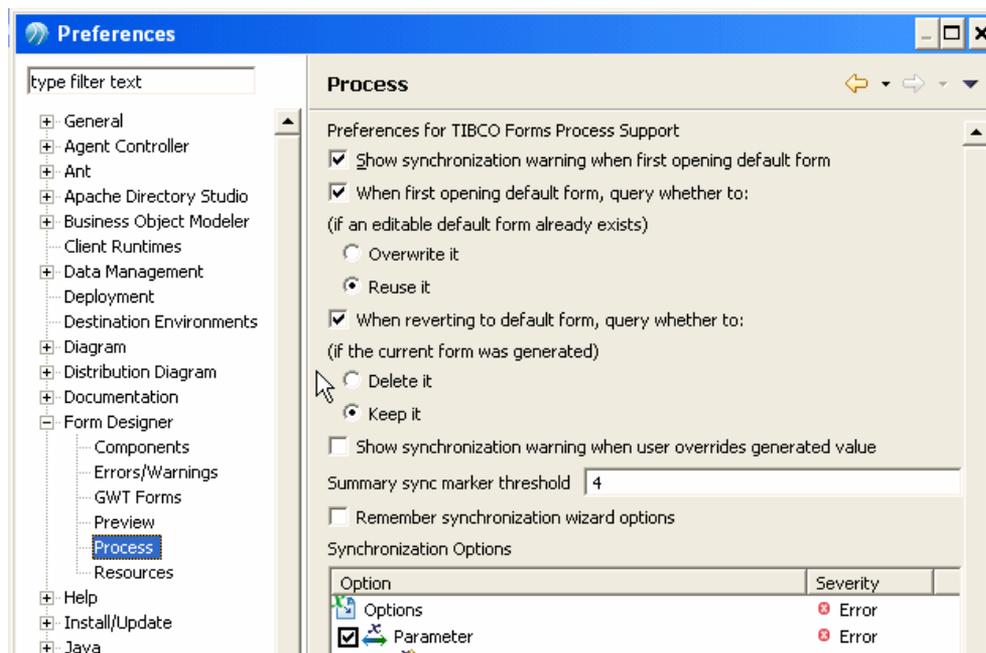
You can also click **Window > Preferences > Form Designer > Errors/Warnings > Process Synchronization** to view descriptions of the items that can be synchronized by the wizard, and view or edit the setting for the severity associated with an out-of-sync condition for each item.



The presence of an **Error** marker will prevent deployment of the form, while **Warning** and **Info** markers will not. If the marker's severity for an item is changed from **Error** to **Warning** or **Info**, the form will be deployable, but doing so is strongly discouraged, since the severity level of **Error** indicates a likelihood of severe problems with the form at runtime if the condition is not addressed. For this reason, the severity levels should normally be left unchanged.

There are additional preferences affecting synchronization that can be edited in the Process dialog (**Window > Preferences > Form Designer > Process**), shown in [Preferences for TIBCO Forms Process Support](#).

Preferences for TIBCO Forms Process Support



Most of the settings here are self-explanatory. The **Summary sync marker threshold** setting deserves mention, however. This is the number of sync markers on the form above which individual problem markers are not shown. By default, the threshold is 4. Change this setting by typing the desired number in the text field. If the number of sync markers on the form are above the **Summary sync marker threshold** value, only one summary marker reading "Form <form name> is out of sync," will be displayed.

If you check the **Remember synchronization wizard options** checkbox, the wizard will not start on page one in the future, **Synchronization Options**, but will take you straight to page two, **Synchronization Actions**. From page two, you can still reach page one by clicking the **Back** button.

Using Data Fields and Parameters with Process User Tasks

Data fields and process parameters represent data that can be used as input to, or output from, steps in a business process. When mapped to a user task, process parameters represent the inputs and outputs

of the user task. The parameters of a user task, in turn, can be mapped to controls on the form associated with that user task.

While XPDL defines two types of parameters: `FormalParameter` and `ActualParameter`, the term *parameter* is used in TIBCO Business Studio as follows:

Formal Parameters

Parameters created at the process level of a process package represent interfaces for *other* processes that want to provide input to, or receive output from, the process that contains these parameters. They can be used, in turn, as input to, or output from, any user task in the business process by being added to the user task's interface. Formal parameters are described in the *TIBCO Business Studio Modeling User's Guide*.

User Task Parameters

User task parameters are properties of a specific user task within a business process, and they are mapped to data fields or process parameters using the **Interface** tab of the user task's Properties View.

The Mode Property of User Task Parameters

The Mode property of a user task parameter refers to whether it is an **In**, **Out**, or **In/Out** parameter. The Mode is set differently depending on whether the parameter is derived from a process parameter or a data field:

- **Process Parameters**
Mode value set on the parameter's Properties view.
- **Process Relevant Data**
Mode value set on the user task's Interface Properties view.

Using Data Fields and Parameters

Incoming data can populate form controls with initial values or dynamically set runtime values for certain control properties, such as the text of a control's label, or whether it is initially visible on the form. Outgoing data are submitted when the user clicks the **Submit** button. These data can be used in various ways in subsequent phases of the business process, sent to an external process, or written to a persistent store.

Relationship Between User Task Parameters and Form Parameters

A user task and a form have their own separate models for the input and output parameters for the task or form. When a form is created from a user task, the form parameters are automatically set to match those in the user task.

The `Synchronize Form` operation on the user task will again update the form parameters to match those in the user task if there have been any changes since the form was originally generated.

Data Types for Data Fields and Process Parameters

Data fields and process parameters must either have one of the predefined primitive data types supported by XPDL or an existing complex type to be mapped to a form parameter.

Primitive Data Types

The following simple data types can be used for a data field or process parameter that will be mapped to a form parameter:

String

A set of alphanumeric characters, with a specified maximum length.

Decimal Number

A decimal number with a specified maximum length and specified number of decimal places.

Integer Number

An integer value, with a specified maximum length.

Boolean

True or false.

Date Time

A combination of date/time.

Time

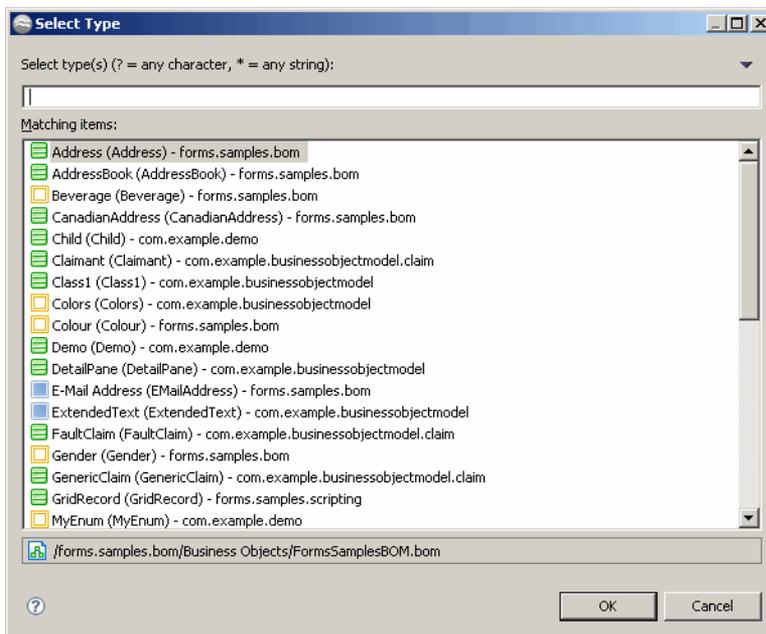
Time only

Date

Date only

Complex Data Types

Complex data types must first be created in a business object model to be available as types for data fields and parameters. All existing complex data types available to the business process appear in the Select Type dialog that opens when **External Reference** is chosen as the type for the parameter or field, and the browse button is clicked:



Icons for Data Types

In the case of data fields, an icon appears next to each data field in the Project Explorer to identify the data type of the field.

Data Types

Icon	Data Type
	Text
	Decimal Number

Icon	Data Type
	Integer Number
	Boolean
	Date Time
	Time
	Date
	External Reference

Data Field Levels

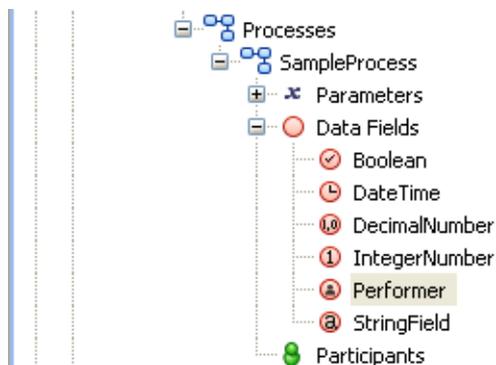
A data field can be created at either of two different levels of the project hierarchy in the Project Explorer, the *process level* or the *package level*. The level determines the scope of the data field, as explained below.

Data Fields at the Process Level

Data fields created at the process level appear under a business process in the Project Explorer. The fields are available to any form associated with a user task within that process, but not to forms in other business processes in the package.

Data fields created at this level appear beneath the business process in the Package Explorer.

Data Fields at the Process Level

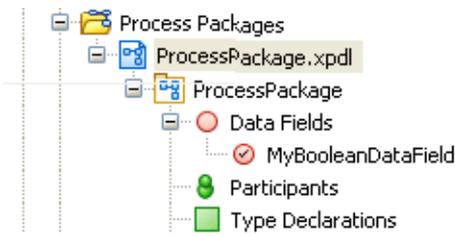


Data Fields at the Process Package Level

Data fields created at the package level appear under a process package in the Project Explorer, and are available to any form in any business process within the process package. In other respects, data fields at the package level are identical to those at the process level.

Data fields created at this level appear beneath the process package in the **Package Explorer**.

Data Fields at the Process Package Level



Using Presentation Channels to Display Tasks to Users

Tasks can be displayed to users in a variety of ways after deployment, depending on how you configure their Presentation Channels.

Example: Displaying Tasks to Users using Email

You can send a notification of a task to a user by email, and they can then click the hyperlink in the email to open the work item in a browser.

When you choose to deliver work by email (by pushing an email notification to a user for a work item), you need to set up your server to deliver work notifications by email. To do this you need to set up Push destinations using the Organization Browser. See the *TIBCO Organization Browser User's Guide* for more information.

A set of email attributes which you can edit allows you to tailor the delivery of the information. To do this select **Window > Preferences > Presentation Channels**, expand **Default Channel** and select **Openspace Email**. Then select the **Attributes** tab.

For more information, including how to create your own email template, see "Creating your Email template files" in *TIBCO Business Studio Modeling User's Guide*.

The email will be delivered using the presentation channel, **Openspace Email**.

Identifying an Appropriate Presentation Channel

The following channel types are provided as Default Channel types: Workspace Google Web Toolkit, Openspace Google Web Toolkit , Openspace Email.

Support for TIBCO General Interface (GI) channel types, which had been deprecated in favor of GWT-based alternatives, has now been discontinued at design time.



If you have existing applications that were designed to use a GI presentation channel, you can replace the channel by opening the project in TIBCO Business Studio. An error message is displayed, with a quick fix that replaces the GI channel with a GWT equivalent. You can then redeploy the application.

Alternatively see "Deployment of Applications that Use Unsupported GI Presentation Channels" in the *TIBCO ActiveMatrix BPM - BPM Administration* guide for information on deploying and running such applications.

You can also add the following channel types:

- **Openspace Mobile**. You can add this to the Default Channel (see [Adding a Channel Type to the Default Channel](#)). Creating an additional presentation channel is not advisable for the Openspace Mobile channel type.
- **Workspace Email**. This channel type has been deprecated and may be removed in a future release.

You cannot add Workspace Email in the default channel and a channel having Workspace Email in it cannot be set as the default. If you add Workspace Email in an additional presentation channel, a warning message is generated indicating that the channel type is deprecated. A quick fix is available that allows you to replace it with a supported channel type.

These behave as described below at runtime:

Channel Type	Behavior at runtime
Workspace Email	Work items are delivered to the user using an email with a link to a Workspace form.

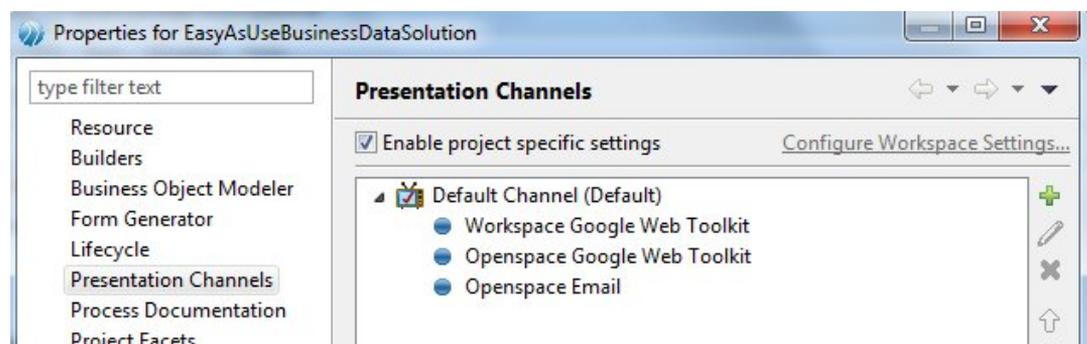
Channel Type	Behavior at runtime
Workspace Google Web Toolkit	Work items are displayed using a link when the user opens their worklist in Workspace.
Openspace Google Web Toolkit	Work items are displayed using a link when the user opens their worklist in Openspace.
Openspace Mobile	Work items are displayed using a link when the user opens their worklist in Openspace Mobile.
Openspace Email	Work items are delivered to the user using an email with a link to an Openspace form.

Viewing the Available Presentation Channels

Presentation channels are defined for the BPM project, which means that all work in the application created from this project will be presented in the specified channels.

Procedure

1. Select **Project > Properties > Presentation Channels** to see channels configured for the project.
2. Expand **Default Channel (Default)** to view the default presentation channels available.



3. If the project should be using shared workspace settings (that is, settings which apply to the entire workspace, so all projects inherit these settings), uncheck **Enable project specific settings** and click **Configure Workspace Settings...** to view or edit the current workspace channel configuration.

Adding a Channel Type to the Default Channel

Procedure

1. Double-click on Default Channel, or click the edit sign on the right of the Presentation Channels dialog. You see a list of the available Channel Types to add (the channel types which are already part of the Default Channel are ticked):
2. Add the Channel Type/s you require, and click **Finish**.



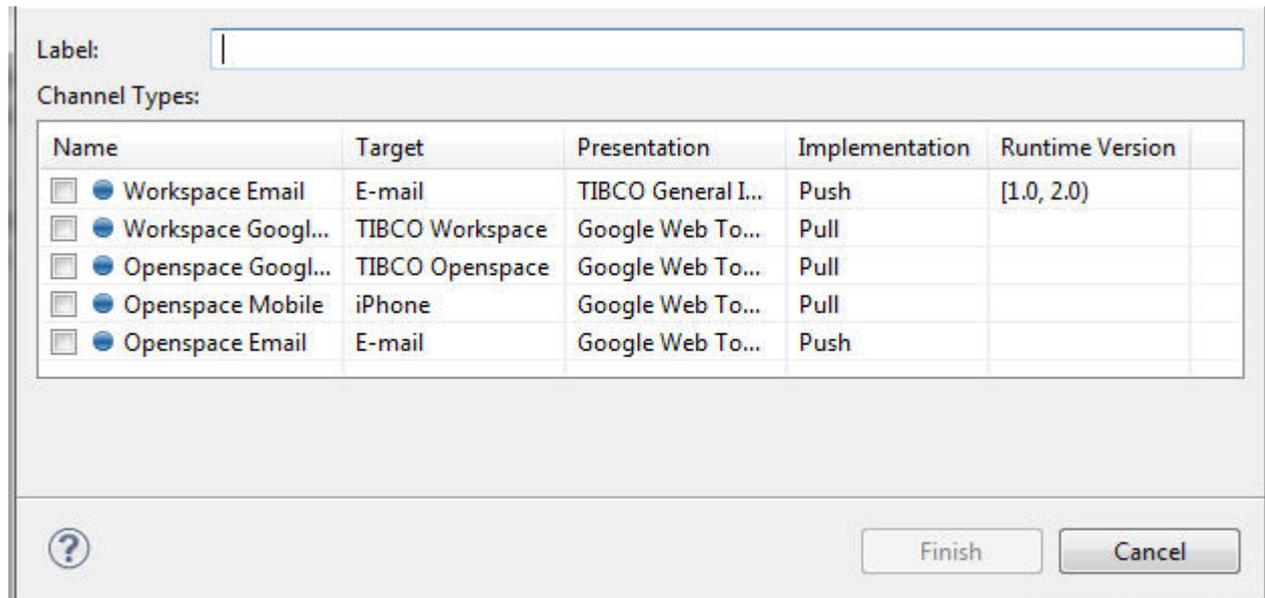
- You cannot have two channel types in the Default Channel with the same target.
- Workspace Email cannot be added in the default channel.

Adding a Presentation Channel

It is only meaningful to add one extra presentation channel, which you can then use to override the default presentation channel when you require.

Procedure

1. Click the plus sign on the right of the Presentation Channels dialog, and you see a list of the available channel types to add:



2. Choose a label for your channel, and select any channel types you want to add to that presentation channel from those available.



Do **not** create a new Presentation channel called "Default Channel".

3. Click **Finish**. The additional presentation channel you selected will be displayed.



You can press the **Restore Defaults** button from the Presentation Channels dialog at any time to go back to the original settings.

Editing Email Attributes

You can edit the attributes for an email channel type to tailor how emails are presented to the user at runtime when a process is deployed.

See the section in *TIBCO Business Studio Modeling User's Guide* for information on how to create your own email template.

Editing Email Attributes at Workspace Level

You can edit the attributes for the Openspace Email presentation channel for the workspace.

Procedure

1. In the Presentation Channels dialog, click **Openspace Email** to view its attributes. The attributes are as follows:

mailTemplateLocation	<p>This attribute allows you to specify a custom email template that you can tailor to meet your own requirements. It contains a picker to allow you to select an alternative email template.</p> <p>See the section in <i>TIBCO Business Studio Modeling User's Guide</i> for information on how to create your own email template.</p>
mailSubject	A brief summary of the contents of the message.
mailPriority	An entry is required here. The default setting is 3. Settings can range from 5-1, with 1 being the highest priority and 3 being what is considered normal.
fromAddress	<p>Address where the message originated from, including an optional name, for example, Jane Smith jsmith@anycompany.com. This is a mandatory field.</p> <p>This needs to be a valid address, depending on your SMTP server settings. Refer to your SMTP server documentation for more information.</p> <p>Note that this does not support multibyte characters. The character repertoire is limited to ASCII as specified in the RFC 822 standard.</p>
ccAddress	<p>An additional, comma-delimited list of recipients. This is an optional field.</p> <p>Note that this does not support multibyte characters. The character repertoire is limited to ASCII as specified in RFC 822 (standard) .</p>
mailHeaders	<p>List of custom headers, of the format <i>Header Name:Header Value</i>, delimited with commas. This is an optional field.</p> <p>Note that this does not support multibyte characters. The character repertoire is limited to ASCII as specified in RFC 822 (standard) .</p>
attachments	Reserved for future use.
presentationChannelId	<p>Identifies which presentation channel the user will open the work item with when they click the HTML link in the pushed email message.</p> <p>openspaceGWTPull_DefaultChannel - opens the work item in Openspace. This is the default setting.</p> <p>This is a mandatory field. The value is filled in automatically when you create a new channel.</p>

2. You can insert the following "tokens" into appropriate email attributes:

- %%token.workItemUrl%% - The work item URL.
- %%token.workItemId%% - The id of the work item.
- %%token.entityName%% - The entity name who requires push notifications as defined in the Organizational Model.
- %%token.mailDate%% - The date and time the pushed mail message was sent.
- %%token.mailFrom%% - The name of the sender of the pushed mail message.
- %%token.mailSubject%% - The subject line for the pushed mail message.

- %%token.mailTo%% - The user/s who the pushed mail message is sent to.
- %%token.mailCc%% - The user/s who the pushed mail message is copied to.
- %%token.mailBcc%% - The user/s who the pushed mail message is blind-copied to.
- %%token.hostIPAddress%% - The IP address of the host.
- %%token.hostMachineName%% - The host machine name.
- %%token.baseurl%% - The base URL.

You can add a token by typing the string directly into an attribute value. For example, for **mailSubject** you could enter: PUSH DEMO - Pushing Work Item Id %%token.workItemId%%.

Then at runtime, the token will be replaced by the actual value:

PUSH DEMO - Pushing Work Item Id 1

3. Click **Apply**. The attribute changes are applied next time you deploy your project.



You can click the **Restore Defaults** button from the Presentation Channels dialog at any time to go back to the original settings.

The **Extended Attributes** tab is also available for custom use.

The **Properties** tab shows the properties set for the presentation channel. The example below shows the properties for **Openspace Email**.

Attributes	Extended Attributes	Properties
Id:	openspaceEmailPush_DefaultChannel	
Name:	Openspace Email	
Target:	E-mail	
Presentation:	Google Web Toolkit	
Implementation:	Push	

Editing Email Attributes at Project Level

You can edit the attributes for the Openspace Email presentation channel for a specific project.

Procedure

1. From Project Explorer, right-click on the project and select **Properties > Presentation Channels**. Check the checkbox **Enable project specific settings**.



Alternatively select **Window > Preferences > Presentation Channels** and click on **Configure Project Specific Settings...** Select the project you want to configure and press **OK**.

2. Expand **Default Channel (Default)** to view the presentation channels available.
3. Check the **Enable project specific settings** checkbox.
4. Edit the presentation channels available. You can add using the plus sign, when you will be shown a list of available channel types to add. You can also delete channel types other than the default ones.

You can edit attributes as described in [Editing Email Attributes](#).

5. Click **Apply**. The attribute changes are applied next time you deploy your project.



You can click the **Restore Defaults** button from the Presentation Channels dialog at any time to go back to the original settings.

The **Extended Attributes** tab is also available for custom use.

The **Properties** tab shows the properties set for this presentation channel.

Sending an Email Message from a Process

You can send email messages from a process by configuring the service task. These messages could be internal notifications giving information about the progress of the process, or external messages such as automatic order confirmations sent to customers.

To set up your server to deliver email you need to do the following:

- An SMTP resource instance must exist in the BPM runtime before you can deploy an application that uses a service task to send an email. This resource instance defines the connection information used by BPM to contact the SMTP mail server.

To define a resource instance, you must use TIBCO Administrator to:

- create a resource template. A resource template specifies configuration properties for resource instances.
- create a resource instance based on the resource template. A resource instance represents a resource shared between applications.
- install the resource instance on a host. The resource instance is then available to applications running on that node.
- There are several different types of resource instance. An SMTP resource instance must be used to provide a connection to an SMTP mail server.
- You use TIBCO Administrator to define an SMTP resource instance.
- You will bind to the resource instance when you deploy the process.

See the *How to Send an Email from a Process* tutorial for more information.

Configuring Service Tasks to Send Email Messages from a Process

A service task that is configured as an email task:

- Defines the email addresses which the message is to be sent from, and to which replies are to be sent, as well as the address of the recipient of the message. Addresses can be specified explicitly or taken from available data fields; you can include fields in the message and the contents of the fields will be used at runtime.
- Defines the body of the text. This can also be typed in explicitly or taken from available data fields.
- Defines any files or field contents that are to be sent as attachments to the email message.

As well as configuring the service task, you need to:

- Create a system participant (of type email),
- Assign that participant to the email service task.
- At deployment, bind the participant to the appropriate SMTP resource instance (which must already exist).

See the *How to Send an Email from a Process* tutorial for more information.

Defining an E-Mail Service Type from a Service Task

Procedure

1. Select the service task. On the **General** tab of the Properties view for the service task, select the **E-Mail** option from the **Service Type** drop-down list.

The screenshot shows a configuration window for an email message. It has the following fields:

- Service Type:** A dropdown menu currently showing "E-Mail".
- To:** A text field containing the placeholder "%ManagerID%".
- Subject:** A text field containing the placeholder "Order %OrderID%".
- Body:** A large, empty text area for entering the message content.
- More Details...:** A link at the bottom of the window.

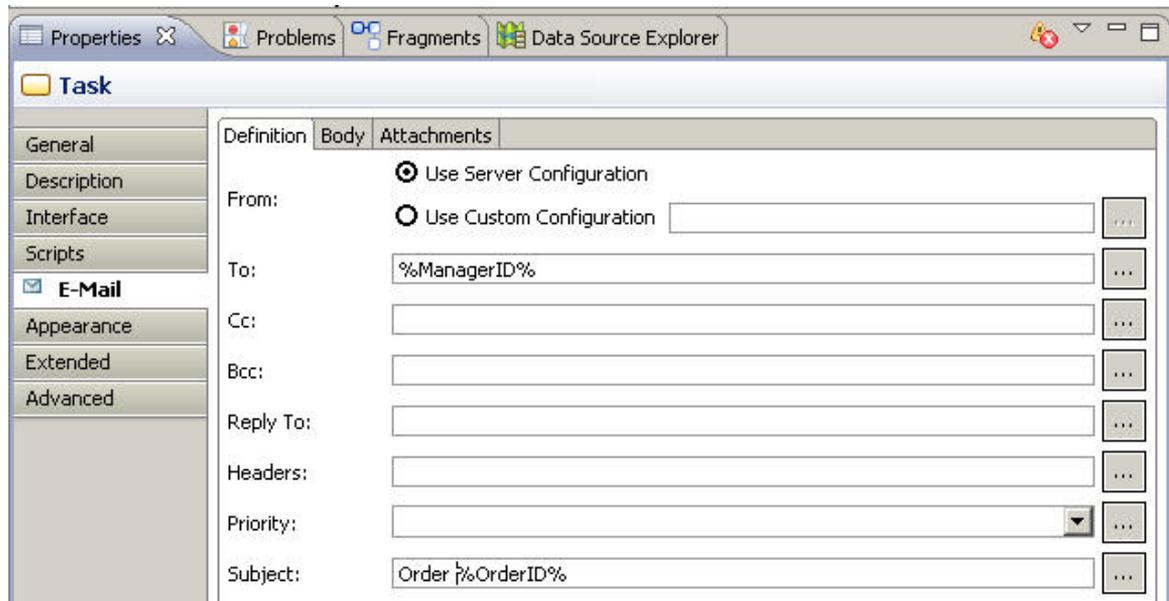
2. Enter an email address for the recipient in the **To:** field, a subject, and the body text for the message. You can enter actual email addresses and plain text, or you can specify data fields, delimited by percent signs. For example, if you specify:

%ManagerID%

in the **To** field, this will be replaced by the contents of the **ManagerID** data field. (See [Setting up Dynamic Data Inputs to an Email Message](#) for more information.)

This is the minimum configuration necessary to send an email message. For further options, click **More Details** (which will take you to the E-Mail tab) or the **E-Mail** tab and continue to specify further parameters.

3. On the **E-Mail** tab, you can specify further parameters for the **Definition** of the email message.



All of the parameters on the E-Mail tab can be specified using a data field or a parameter.

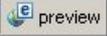


Click the  button to display the Select Data Field or Formal Parameter dialog and choose the required data field or parameter. You can also mix text with data field values, as shown in the **Subject** field of the example illustration.

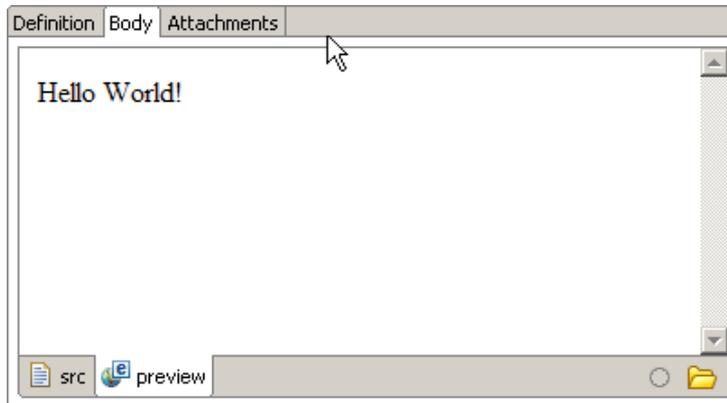
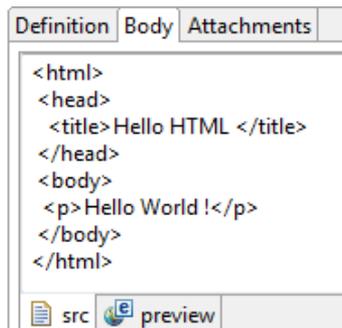
- **From:** Use **Custom Configuration** to choose a data field or parameter which will specify a different **From:** address for this email.
 - **To:** Specify the recipient of the email, either as an explicit email address or by selecting a data field or parameter. This is a mandatory field and you will receive an error if it is not present.
 - **Cc:** Specify any recipients to whom you want to send a copy of the email either as explicit email addresses or by selecting a data field or parameter. Their email address is visible to other recipients of the email.
 - **Bcc:** Specify any recipients to whom you want to send a blind copy of the email either as explicit email addresses or by selecting a data field or parameter. Their email address is not visible to other recipients of the email.
 - **Reply To:** Use this parameter to specify a different email address to which recipients of a message can reply. Alternatively, select a data field or parameter.
 - **Headers:** Message headers provide a list of technical details, such as who sent the mail message, the software used to compose it and the email servers it passed through to get to the recipient. Use this parameter to specify additional information in the header of the email.
 - **Priority:** Select a priority from the drop-down list (**Low**, **Normal**, or **High**) or select a data field or parameter.
 - **Subject:** Type the Subject line for the email message or select a data field or parameter.
4. Click **Body** to specify the main text of the message. Click in the main part of the field to enter text. You can enter either plain text or HTML. If you enter HTML, then when you use the preview option below, you will see what the output of the HTML will be. You can also specify the contents of data fields or files to be inserted.

At the bottom of the text area are the following buttons:

- Click on the  button to view the source of the body text of the email.

- Click on the  button to display a preview of how the body text will look to the recipient.
- Click on the  button to open the Select Data Field or Formal Parameter dialog, which you can use to select a field and specify that the contents of the field should be inserted in the email. This button is only available when you are viewing the source of the body text, not when you have selected a preview.
- Click on the  button to open the Open dialog, which you can use to select a file and specify that the contents of the file should be inserted in the email.

You can alternate between viewing the source of the body text and previewing how it will look to the recipient, as in the example of HTML for 'Hello World' below.



5. Click **Attachments** to specify a document to be attached to the message:
 - **Field Contents:**
At run time, the resulting e-mail message produced by this task has an attachment named field.txt, where field is the name of the attached data field.
 - **Files:**
Use this option to browse the file system and attach one or more files to the email message.



To be successful, file attachments must use a consistent directory structure and file name at design time and runtime. So for example, if you are attaching a file `C:\temp\readme.txt` at design time, then that same file, in the same directory, needs to be available to the runtime.

This also means that in a distributed environment, this needs to be available on any host running a BPM Server node. It would be good practice to have a mapped drive available to all hosts where the attachment is available.

Setting up Dynamic Data Inputs to an Email Message

Data fields available to the process can be used to dynamically set parts of the email at runtime, using either `%FieldName%` notation or the Select Data Field or Formal Parameter dialog.

See [Defining an E-Mail Service Type from a Service Task](#).

However, a process formal parameter, or a package-level or process-level data field that is defined as an external reference to an attribute of a class defined in a business object model (that is, as structured data), cannot be used in this way.

Procedure

1. Create an activity-level data field on the **Data Fields** tab of the email service task. The data field:
 - must be defined as a simple type that matches the type of the attribute you want to reference.
 - must not have the same name as an existing formal parameter, or package-level or process-level data field.
2. On the **Scripts** tab, define an **Initiate Script** (of type **JavaScript**) to assign the value defined in the desired attribute to the data field.

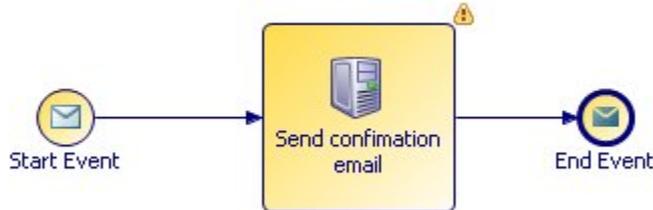
The attribute must have been populated earlier in the process - for example:

- by passing the data into the process from an external source, using a message start event, a receive task or a catch message intermediate event. See [Exposing a Web Service](#).
- by collecting the data on a form. See "Associating Process Data with Events and Tasks" in *TIBCO Business Studio Modeling Guide*.
- in a script, using a dynamically created factory method to create a new data object from the attribute's parent class. See [Dynamically Created Factory Methods](#).

3. On the **General** or **E-Mail** tab, enter the local data field name at the appropriate place, using either `%FieldName%` notation or the Select Data Field or Formal Parameter dialog (as described in [Defining an E-Mail Service Type from a Service Task](#)).

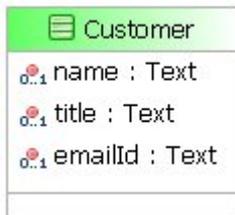
Example of Setting up Dynamic Data Inputs to an Email Message

A project contains a process that uses an email service task to send a delivery confirmation email to a customer.

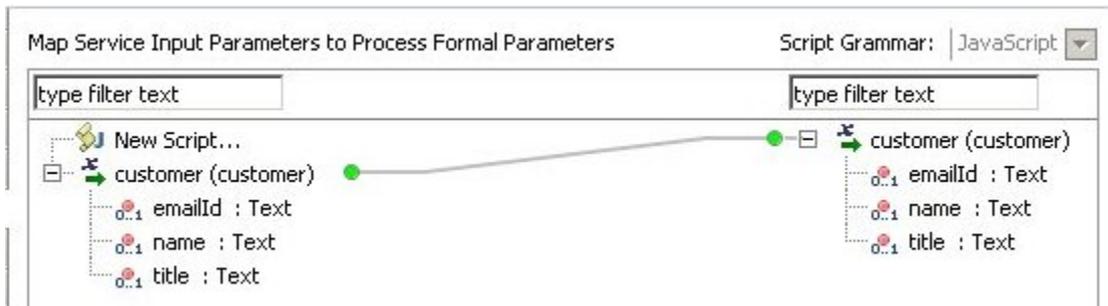


The process is triggered by an external application that sends in the customer's email address, name and title.

The **Start Event** collects this data via a formal parameter, **customer**, which is defined as an external reference to a business object model that defines a **Customer** class.



The **Input to Process** tab of the **Start Event** defines the mapping of the received data to the **customer** formal parameter.



The **Send confirmation email** service task will use the email address, name and title to dynamically build the email message. However, the service task cannot use the **customer** formal parameter to obtain this data.

Instead, to access the **Customer** attributes **name**, **title** and **emailId** on the service task, you must perform the following steps:

Procedure

1. Define the following activity-level data fields on the **Data Fields** tab of the **Send confirmation email** service task.

	Label	Name	Read Only	Type	Length	Decimal...	Array	External Reference	Type Declaration
General	@ locName	locName	<input type="checkbox"/>	Text	50	NA	<input type="checkbox"/>	NA	NA
Description	@ locEmailId	locEmailId	<input type="checkbox"/>	Text	50	NA	<input type="checkbox"/>	NA	NA
Interface	@ locTitle	locTitle	<input type="checkbox"/>	Text	50	NA	<input type="checkbox"/>	NA	NA

- On the **Scripts** tab, define the following **Initiate Script** to assign the values defined in the **Customer** object to these data fields.

Task Scripts

Initiate Script | Complete Script | Timeout Script | Cancel Script

Script executed by the process engine on initiation of the activity.

Script Defined As: JavaScript

Describe Initiated Script:

```
locName = customer.name;
locEmailId = customer.emailId;
locTitle = customer.title;
```

- On the **General** tab, enter the field names as shown below.

Service Type:	E-Mail
To:	%locEmailId%
Subject:	Order delivery
	Dear %locTitle% %locName%,
	Your order has now shipped.
	Thank you for shopping with us!

Result

At runtime, the email message:

- will be sent to the email address defined in the **emailId** attribute of **Customer**.
- will contain the **title** and **name** defined by those **Customer** attributes.

Calling a Database From a Process

This section explains how to use a service task to establish a Java Database Connectivity (JDBC) connection to an external database and execute either a stored procedure or an ad-hoc SQL query.

You can use a service task to establish a JDBC connection to an external database and execute a database operation - either a stored procedure, or a SQL query (a SELECT, INSERT, UPDATE or DELETE statement).



See the BPM Installation guide for details of supported database types.

Step	Task description	For more information, see...
1	(Optional) Define a database connection profile. Do this if you want to be able to introspect the database at design-time to help you define a SQL query, or automatically generate a business object model class to hold the return value from your database call.	Defining and Using a Database Connection Profile
2	Configure a system participant to identify the target database.	Using a System Participant to Identify the Target Database
3	Configure a service task to i) call the target database and ii) execute either a SQL query or a stored procedure.	Configuring a Service Task to Call a Database
4	Map the input and output parameters defined in your database call to corresponding data fields or parameters in the process.	Mapping Data Between the Process and the Database
5	When you deploy the project, bind the system participant to a resource instance that identifies the target run-time database.	

Defining and Using a Database Connection Profile

A database connection profile defines a connection to a design-time target database for the database service call.

Using a database connection profile is entirely optional. Doing so can make it easier to define the database service task by enabling you to introspect the database to:

- use the SQL Query Editor to interactively build up and test a SQL query.
- automatically populate the database parameters on the **Database** tab.
- automatically generate a business object model class to hold the return value from your database call.



The design-time target database referenced by a database connection profile does not need to be the same as the runtime target database - you only select the runtime target database when you deploy your project. Indeed, security and/or access considerations may mean that you cannot use a database connection profile to connect to your runtime target database.

However, if you do use a separate design-time target database - for example, one set up explicitly for development and/or testing purposes, it must use the same schema as the runtime target database

Creating a Database Connection Profile

Procedure

1. Click the Data Source Explorer view.
2. Right-click **Database Connections** and select **New**. The **New Connection Profile** wizard is displayed.
3. Select the type of database connection you want to create - either **Oracle, SQL > Server** or **DB2 for Linux, UNIX and Windows**.
4. Enter a **Name** and, optionally, a **Description** (optional), then click **Next**. The **Specify a Driver and Connection Details** page is displayed.
5. Select an appropriate driver from the **Drivers** drop-down list. If one does not exist, you must define a new driver. To do this:
 - a) Download and install the appropriate JDBC drivers for the type of database you want to use.
 - b) Click . The New Driver Definition dialog is displayed.
 - c) On the **Name/Type** tab, select one of the available driver definition templates. This populates the **Driver name** and **Driver type** fields.
 - d) Click the **Jar List** tab.
 - e) Select any **.jar** files that are listed and click **Remove JAR/Zip** to delete them.
 - f) Click **Add JAR/ZIP**. A file selection dialog is displayed.
 - g) Browse to the location of your JDBC drivers. Select the appropriate **.jar** files and click **Open**. The selected **.jar** files are displayed on the **Jar List** tab.
 - h) Click **OK** to return to the **New Connection Profile** wizard.
6. The **Properties** section displays default connection details for the selected driver. Edit these as required with the appropriate information for the database you want to connect to. See [Mapping the Result Set](#).
7. Click **Test Connection** to verify that you can connect to the database using the details you entered.
8. Optionally, select **Save password** to avoid having to enter the database password each time you connect.
9. Click **Finish**.
10. The new connection profile is added to the list of **Databases** in the Data Source Explorer view.

Creating and Using a Local Copy of the Database Connection Profile

You can use a database connection profile to either connect directly to the database, or to create and use a local copy of the database schema.

Procedure

1. Expand the list of **Databases** in the Data Source Explorer view.

2. Right-click the database connection profile you want to copy and select **Connect** (if you are not already connected).
3. Right-click the database connection profile and select **Save Offline**.



If you choose to use a local copy of the databases schema, you must manually keep the local copy synchronized with the master database. If you believe the master schema has changed, use the same procedure to recreate your local copy.

Working Online or Offline To Connect to the Database or to the Local Copy

Procedure

1. Expand the list of **Databases** in the Data Source Explorer view.
2. Right-click the database connection profile you want to use and select:
 - **Connect** if you want to connect to and use the database.
 - **Work Offline**, if you want to use your local copy of the database schema.

Using a System Participant to Identify the Target Database

A system participant is a logical identifier for a connection to an external system - in this case, a database. A database service task must use a system participant that identifies the database that it is to connect to.

A system participant can be mapped to different databases as required:

- At design-time, the system participant can (optionally) be mapped to a database connection profile, allowing you to introspect the design-time target database while you are defining your database call.
- During deployment, the system participant must be mapped to a JDBC resource instance that defines the connection to the runtime target database. (See for more information.)

If you are using a database connection profile, TIBCO Business Studio can automatically create and configure the system participant for you when you configure the service task. See [Configuring a Service Task to Call a Database](#).

Otherwise, you should perform the following tasks manually before you configure the service task.

Creating a System Participant and Mapping it to a Target Database

Procedure

1. Create a system participant. (See "Creating Participants" in the *TIBCO Business Studio Modeling User's Guide* for more information about how to do this.)
2. In Project Explorer, select the system participant.
3. On the **General** tab of the Properties view, expand **Shared Resource** and select **JDBC**.
4. Use one of the following fields to identify the database connection represented by this system participant:
 - **Instance Name:** Enter the name of a JDBC resource instance that identifies a connection to a runtime target database. (The connection will only be actually defined when you deploy the project.)

- **Jdbc Profile Name:** Enter the name of a database connection profile that defines a connection to a design-time target database. (You must do this if you want to be able to introspect the design-time target database while you are defining your database call.)

Assigning the System Participant to the Database Service Task

Procedure

1. Select the service task that you want to use to call the database.
2. Click the **General** tab of the **Properties** view.
3. Click the picker to the right of the **Participants** field. The Select Participants dialog is displayed.
4. Select the appropriate system participant, then click **Add** to add it to the **Selection** list.
5. Click **OK**. The selected participant is displayed in the **Participants** field.

Result



Alternatively, you can drag and drop a system participant from Project Explorer onto the database service task. If the task already has participants, a context menu is displayed - select **Set Task Participant(s)** to replace the existing participant list with the new participant.

If a process contains multiple database service tasks that each connect to the *same* database, they can all use the same system participant.

Configuring a Service Task to Call a Database

Procedure

1. Select the service task that you want to use to call the database.
2. On the **General** tab of the **Properties** view, set **Service Type** to **Database**.
3. If you have already defined the system participant that you want to use, drag it from Project Explorer and drop it in the **Participants** field. See [Using a System Participant to Identify the Target Database](#).
4. Select the **Operation** that you want to execute on the database, either:
 - **SQL**, to execute a SQL query.
 - **Stored Procedure**, to execute a stored procedure.
5. Define your SQL query or select the stored procedure that you want to execute on the database. See [Creating a SQL Query](#) or [Selecting a Stored Procedure](#).
6. Define the input and/or output parameters required for the query or stored procedure. See [Mapping Data Between the Process and the Database](#).

Creating a SQL Query

You can create your SQL query in two ways, by manually entering a query or accessing SQL Query Builder (if you have defined a database connection profile, you can use the Eclipse Data Tools Platform

(DTP) SQL Query Builder to interactively define and, optionally, test the SQL query against the design-time target database).

Use of Multiple Statements

If you are using Oracle PL/SQL to define a query you can use multiple INSERT, DELETE and UPDATE statements in a clause. For example:



```
BEGIN
delete from BPMUSER.ORDERDETAILS;
delete from BPMUSER.CUSTOMERDETAILS;
delete from BPMUSER.MUSICCATALOGUE;
END;
```

The use of multiple SELECT statements in this way is not supported.

Procedure

1. Select the service task that you want to use to call the database.
2. On the **General** tab of the **Properties** view, type the required SQL code directly in the **SQL** area in the right-hand pane.
3. Select the service task that you want to use to call the database.
4. On the **General** tab of the **Properties** view, click the picker in the right-hand pane.
 - If a system participant is already assigned to the database service task, and the system participant is mapped to a database connection profile, SQL Query Builder opens, using that database connection.
 - If the assigned system participant is not mapped to a database connection profile, or if no system participant is assigned to the database service task, the Connection Profile Selection dialog is displayed.

Select the database connection profile that you want to use, then click **OK**. SQL Query Builder opens, using the selected database connection.

A system participant is also automatically created or updated and assigned to the database service task, as shown in the following table.

If...	Then...
No system participant was assigned to the database service task.	<ol style="list-style-type: none"> 1. A system participant is created at the process level, with the same name as the selected database connection profile. 2. The participant's Shared Resource property values are automatically set: <ul style="list-style-type: none"> • Name is set to the same name as the selected database connection profile. • Type is set to Database. 3. The system participant is assigned to the database service task.
A system participant was already assigned to the database service task, but was not mapped to a database connection profile	The Name value of the system participant's Shared Resource property is changed to the same name as the selected database connection profile.

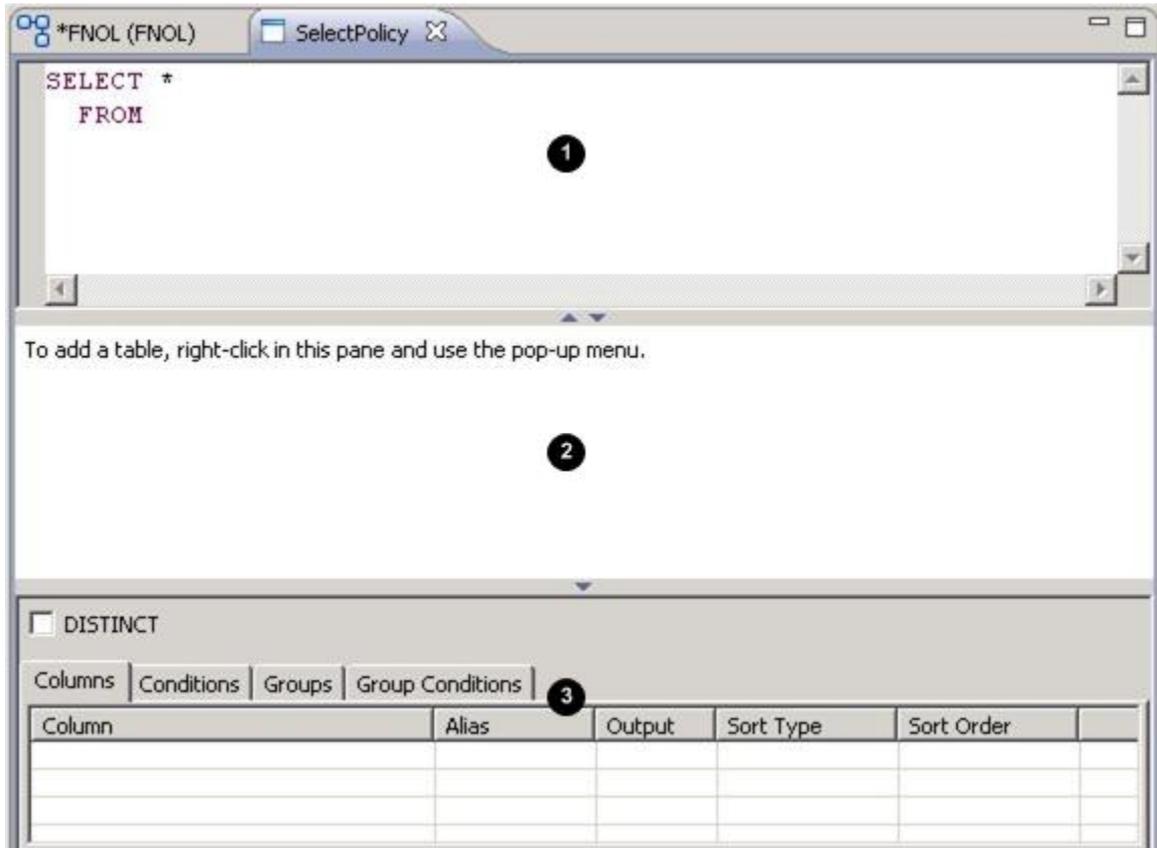
Using SQL Query Builder

SQL Query Builder provides a graphical interface that provides access to your database schema and objects so that you can quickly create or edit SQL statements without actually typing any SQL code.



This section provides only a brief overview of the Query Builder. See the *Data Tools Platform Guide* for more detailed information about its features and how to use it. You can find this guide in the *Supplemental Eclipse Help* section of the TIBCO Business Studio Help system.

The Query Builder contains three panes:



- **The Outline Pane**

(1) The Outline pane contains the SQL statement. You can either edit this directly in this pane or use the Table and Detail panes to automatically build it up.

- **The Table Pane**

(2) The Table pane displays the tables and columns to use in the SQL statement. To add a table, right-click anywhere in the Table pane and select **Add Table**.

- **The Detail Pane**

(3) The Detail pane contains information appropriate to the SQL statement and tables in the first two panes. For example, you can use it to define conditions in the statement, or to define the sort order and type of selected table columns.

Changing the SQL Statement

The default statement is `SELECT * FROM`.

Procedure

1. Right-click anywhere in the Outline pane and select **Change Statement Type**. The Change SQL Statement Type dialog is displayed.
2. Select the statement you want - either `SELECT`, `INSERT`, `UPDATE` or `DELETE`.
3. Click **OK**.

Result

The current contents of all three panes are cleared and the appropriate SQL syntax for the new statement is displayed in the Outline pane.

Building a Query - A Simple Example

The following example briefly demonstrates how to build a simple query against a database that contains employee information. The query selects an employee's first and last names and their job title, based on their employee number (which will be passed into the query from the service task).

Service Type: Database

Operation: SQL Stored Procedure

SQL: `SELECT FIRSTNAME, LASTNAME, JOBTITLE
FROM CLASSICMODELS.EMPLOYEES
WHERE EMPLOYEENUMBER = ?`

Procedure

1. Right-click anywhere in the Table pane and select **Add Table**. The Add Table dialog is displayed.
2. Browse and select a table - in this case, **EMPLOYEES**, then click **OK**. The table is displayed in the Table pane, showing the columns contained in the table.

EMPLOYEES	
<input type="checkbox"/>	EMPLOYEENUMBER
<input type="checkbox"/>	LASTNAME
<input type="checkbox"/>	FIRSTNAME
<input type="checkbox"/>	EXTENSION
<input type="checkbox"/>	EMAIL
<input type="checkbox"/>	OFFICECODE
<input type="checkbox"/>	REPORTSTO
<input type="checkbox"/>	JOBTITLE

3. Click the fields that you want the query to return in the order that you want them returned - in this case, **FIRSTNAME**, **LASTNAME** and **JOBTITLE**.

Note that the Outline pane now displays:

```
SELECT FIRSTNAME, LASTNAME, JOBTITLE
FROM CLASSICMODELS.EMPLOYEES
```

4. Add a condition to input an employee number to the query:
 - a) Click **Condition** in the Details pane.
 - b) Click the first row under **Column** and select **EMPLOYEEENUNBER** from the drop-down list.
 - c) Enter = as the **Operator** (or select it from the drop-down list).
 - d) Enter ? as the Value. This indicates that the value will be passed to the query from the database service task.

<input type="checkbox"/> DISTINCT				
Columns	Conditions	Groups	Group Conditions	
Column	Operator	Value	AND/OR	
EMPLOYEEENUNBER	=	?		

5. Enter **CTRL + S** to save the query.

Testing the SQL Statement

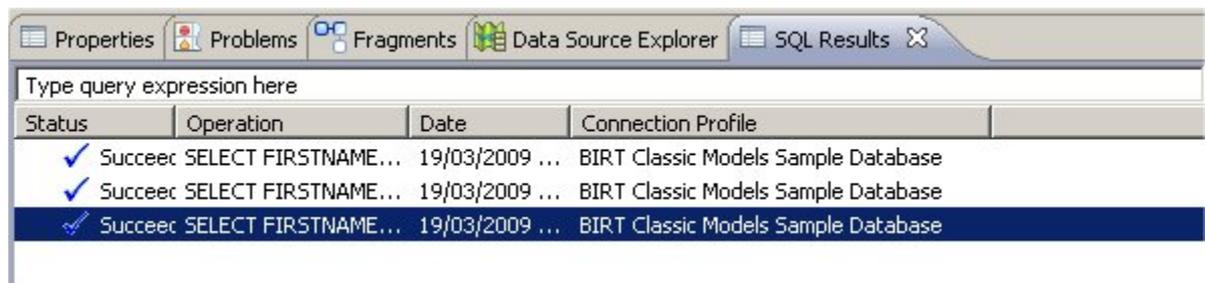
Procedure

1. Right-click anywhere in the Outline pane and select **Run SQL**.
2. If your query defines any input variables (conditions defined in the '?'), the Host Variable Values dialog is displayed, in which you can enter values for each variable.

Result

The **SQL Results** view is displayed. This contains two panes:

- The left-hand pane shows the status of each SQL statement that you have run in this session.



Status	Operation	Date	Connection Profile
✓	Succesec SELECT FIRSTNAME...	19/03/2009 ...	BIRT Classic Models Sample Database
✓	Succesec SELECT FIRSTNAME...	19/03/2009 ...	BIRT Classic Models Sample Database
✓	Succesec SELECT FIRSTNAME...	19/03/2009 ...	BIRT Classic Models Sample Database

- The right-hand pane shows the status and results (in separate tabs) of the query that is currently selected in the left-hand pane.



Status	Result1		
	FIRSTNAME	LASTNAME	JOBTITLE
1	Diane	Murphy	President



If you want the right-hand pane to display just the results, on a single tab, click 

Selecting a Stored Procedure

You can select a stored procedure in two ways: you can manually enter the stored procedure name, or if you have defined a database connection profile, you can select the stored procedure from the design-time target database.

Manually Entering a Stored Procedure Name

Stored procedures names can be manually entered.

Procedure

1. Select the service task that you want to use to call the database.
2. On the **General** tab of the **Properties** view, type the stored procedure name directly in the **Name** field in the right-hand pane, using the following format:


```
[owner].stored_procedure_name
```

 where *owner* is the name of the database user that owns the *stored_procedure_name* stored procedure.

Selecting a Stored Procedure From the Database

You can select a stored procedure from the database.

Procedure

1. Select the service task that you want to use to call the database.
2. On the **General** tab of the **Properties** view, click the picker next to the **Name** field in the right-hand pane.
 - If a system participant is already assigned to the database service task, and that system participant is mapped to a database connection profile:

TIBCO Business Studio opens the database connection and displays the Pick Stored Procedure dialog.

Expand the database through the **Catalogs** and **Schemas** until you find the stored procedure you want, select it and click **OK**. The selected stored procedure is displayed in the **Name** field.
 - If a system participant is already assigned to the database service task but is not mapped to a database connection profile, or if no system participant is assigned to the database service task:

The Connection Profile Selection dialog is displayed.

 1. Select the database connection profile that you want to use, then click **OK**. TIBCO Business Studio opens that database connection and displays the Pick Stored Procedure dialog.
 2. Expand the database through the **Catalogs** and **Schemas** until you find the stored procedure you want, select it and click **OK**. The selected stored procedure is displayed in the **Name** field.

A system participant is also automatically created or updated and assigned to the database service task.

Updating a Stored Procedure Used in a Database Task Activity

The Database Task Activity holds a cache of metadata for stored procedures for performance reasons. It checks for any changes in the database activity that uses the stored procedure (for example, changes in

the number of input and output parameters, type changes) before using the one from cache. If there is a change, the initial cache entry is invalidated and the metadata for the stored procedure is retrieved.

If you decide to update the stored procedure, you must do so in the following order.



If it is not done in this order, the cache entry will reflect that of the previous stored procedure.

Procedure

1. Update the stored procedure in the database.
2. Update and deploy the process using the stored procedure.

Mapping Data Between the Process and the Database

Each input and output parameter defined in the SQL query or stored procedure must be mapped to a corresponding data field or parameter in the process.

Depending on the type of the database parameter, the process data field or parameter can be:

- a simple data field or parameter
- an existing business object model object
- (for values returned to the process from the database) a new business object model object. See [Automatically Creating a Business Object Model to Store Returned Data](#).

Mapping Data Parameters for each parameter required by the SQL query or stored procedure

Mapping Data Parameters

You must map parameters for each parameter required by the SQL query or stored procedure

Procedure

1. Select the service task that you are using to call the database, then click the **Database** tab of the **Properties** view.
2. If you have selected the stored procedure from the database, the **Parameter** and **Type** fields are automatically populated from the stored procedure. Otherwise, click the plus sign to add a new mapping.
3. In the **Parameter** field, enter a suitable name to identify the database parameter that this mapping refers to.



Parameters must be listed in the same order that they appear in the query or are used in the stored procedure. You can use the **Move Up** and **Move Down** buttons to alter the order of listed mappings.

4. In the **Type** field, select:
 - **IN**, if the mapping defines an input parameter from the process to the database.
 - **OUT**, if this mapping defines an output parameter from the database to the process.



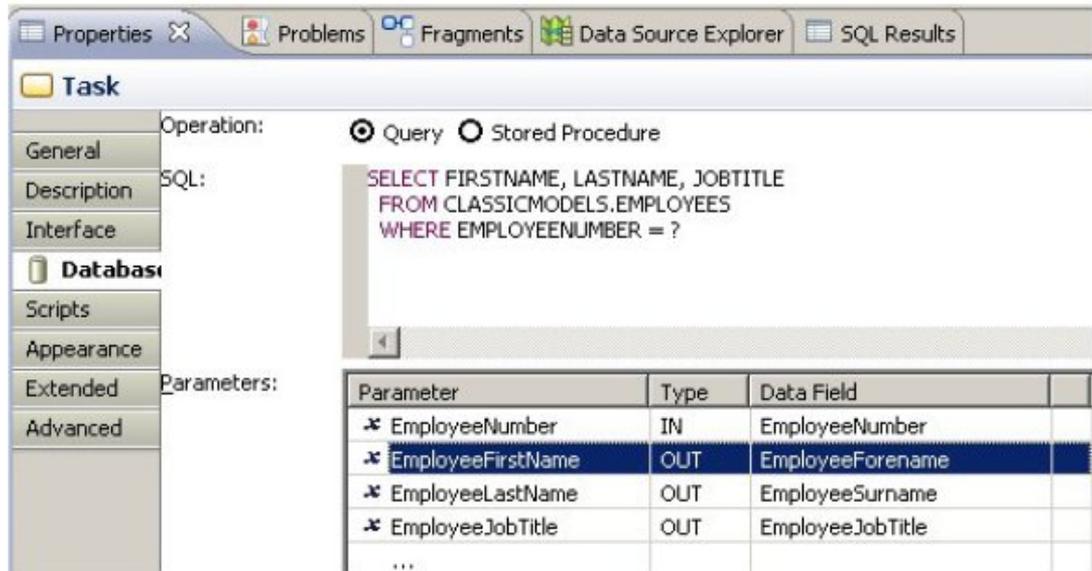
If you are using a stored procedure, the **Parameter** name used for a mapping must be the same as the name of the parameter declared in the stored procedure.

- **INOUT**, if this mapping defines an input/output parameter to/from the database.
5. In the **Data Field** field, click the picker. The Select Data Field or Formal Parameter dialog is displayed.
 6. Select the process data field or formal parameter that you want to map to this database parameter and click **OK**.



- The data type of the data field or parameter must match the data type of the database parameter it is being mapped to. TIBCO Business Studio does not validate this.
- You cannot use a data field or parameter that is defined as an external reference to a business object model for a mapping of type **IN**. If you select such a data field or parameter, a validation error is displayed against the database service task.

The following screenshot shows example mappings for a query that takes a single input parameter and returns three output parameters.



Mapping an Externally Referenced Class Attribute to a Database Parameter

A process formal parameter, or a package-level or process-level data field, cannot be mapped to a database parameter if the parameter or data field is defined as an external reference to an attribute of a class defined in a business object model (that is, as structured data).

Procedure

1. Create an activity-level data field on the **Data Fields** tab of the database service task. The data field:
 - must be defined as a simple type that matches the type of the attribute you want to reference.
 - must not have the same name as an existing formal parameter, or package-level or process-level data field.
2. On the **Scripts** tab, define an **Initiate Script** (of type **JavaScript**) to assign the value defined in the desired attribute to the data field.



The attribute must have been populated earlier in the process - for example, by:

- passing the data into the process from an external source, using a message start event, a receive task or a catch message intermediate event. See [Exposing a Web Service](#).
 - collecting the data on a form. See "Associating Process Data with Events and Tasks" in *TIBCO Business Studio Modeling User's Guide*.
 - in a script, using a dynamically created factory method to create a new data object from the attribute's parent class. See [Dynamically Created Factory Methods](#).
3. On the **Database** tab, map the local data field name to the database parameter (as described in [Mapping Data Parameters for each parameter required by the SQL query or stored procedure](#)).

Automatically Creating a Business Object Model to Store Returned Data

You can automatically create a business object model to store data returned by the SQL query or stored procedure.

Procedure

1. On the **Database** tab, select a database **Parameter** defined as **OUT** or **INOUT**:
2. In the **Data Field** field, click . The Select Data Field or Formal Parameter dialog is displayed.
3. Select **Create new return parameter**, then click **OK**. A data field called **ResultSet** and a corresponding business object model are automatically created. For example:

Parameters:

Parameter	Type	Data Field	
EmployeeNumber	IN	EmployeeNumber	
EmployeeDetails	OUT	ResultSet	
...			



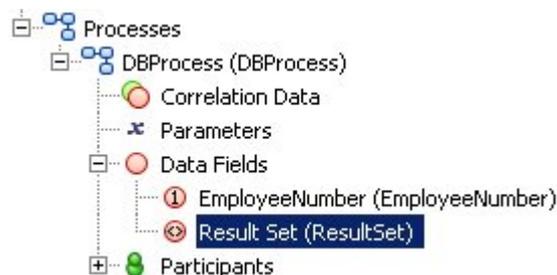
If this is not the first data field created in this way, a unique number is appended to the name - **ResultSet1**, **ResultSet2** and so on.

You can open the business object model directly by clicking .

Result

A **ResultSet** data field and a **ResultSet** business object model are created as follows:

- The **ResultSet** data field is automatically created in the process' **Data Fields** folder.



The data field is defined as an **External Reference** to a business object model. The business object model is also automatically created, with a name formed by concatenating the database service task's name with the data field's name - in this example, **DBCall13ResultSet**.)

The data field is defined as an **External Reference** to a business object model. The business object model is also automatically created, with a name formed by concatenating the database service task's name with the data field's name - in this example, **DBCall13ResultSet**.)

Data Field

General Label: Result Set

Description Name: ResultSet

Extended

Advanced

Type...

Read Only:

Basic Type

Declared Type

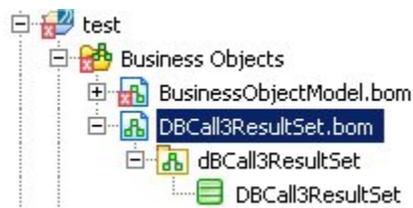
External Reference

Set a reference to a type defined in the domain model:

Reference: DBCall3ResultSet

Array

- The **ResultSet** business object model is created in the project's **Business Objects** folder.



The business object model contains a single class (with the same name as the business object model), which is automatically populated with attributes whose names and types match the data returned by the database.

TIBCO Business Studio can only populate the business object model attributes if you are using a database connection profile to provide a connection to a design-time target database and you are defining an ad-hoc SQL query.



If you do not have a database connection, or if you do but you are defining a stored procedure, the business object model class will be empty and you will need to manually add the required attributes and ensure that their names and types match those being returned by the database.

Mapping the Result Set

TIBCO Business Studio automatically populates the **Database** tab with the return parameters from the stored procedure.



- Despite being similar in appearance, the return parameters from the stored procedure are *not* created as parameters of the process. For this reason, they are visible only on the **Database** tab, and not in the Project Explorer.
- You should also check any stored procedure parameters created on the **Database** tab to see that they have been created correctly.

Each input and output parameter defined in the stored procedure must be mapped to a corresponding data field or parameter in the process.

The process data field or parameter can be:

- a basic type data field or parameter
- an existing business object model object
- (for values returned to the process from the stored procedure) a new business object model object.)



You must ensure that the data type of your selected data field or formal parameter matches the data type of the database parameter it is being mapped to. TIBCO Business Studio does not validate this.

You map parameters using the **Database** tab of the **Properties** view.

JDBC Driver Connection Details

The following sections define the connection information required when defining a new JDBC driver for a database connection profile.

For complete information, refer to the documentation accompanying a specific driver.

SQL Server

Field	Description
Database	The name of the database that you want to connect to.
Host	The IP address or host name where the database is located.
Port number	The port used to access the database.
Use integrated authentication	Whether the database uses Windows authentication or SQL Server authentication. If this checkbox is selected the User name , Password and Save password fields are disabled.
User name	The SQL Server login used by the connection.
Password	The password for the SQL Server login used by the connection.
Save password	Whether the password is saved. If not, the password must be entered on every connection attempt.
Connection URL	The connection URL. This is built up automatically from the connection information entered in the other fields.

Oracle

Field	Description
SID	The SID of the database that you want to connect to.
Host	The IP address or host name where the database is located.
Port number	The port used to access the database.
User name	The Oracle user name used by the connection.
Password	The password for the Oracle user name used by the connection.
Save password	Whether the password is saved. If not, the password must be entered on every connection attempt.
Connection URL	The connection URL. This is built up automatically from the connection information entered in the other fields.
Catalog	The name of the catalog containing the database that you want to connect to.

DB2

Field	Description
Database	The name of the database that you want to connect to.
Host	The IP address or host name where the database is located.
Port number	The port used to access the database.
Use client authentication	Whether the database uses client authentication or server authentication. If this checkbox is selected the User name , Password and Save password fields are disabled.
User name	The DB2 user name used by the connection.
Password	The password for the DB2 user name used by the connection.
Save password	Whether the password is saved. If not, the password must be entered on every connection attempt.
Connection URL	The connection URL. This is built up automatically from the connection information entered in the other fields.

Using Web Services

This section provides general information about using web services with a process.

Subsequent topics provide more specific information about how to call or expose a web service from a process - see:

- [Calling a Web Service](#).
- [Exposing a Web Service](#).

BPM application architecture is based on SOA principles: applications are composed of services that interact by exchanging messages.

Using BPM, you can call web services from a process and expose a process as a web service.

Web Service Definition Language (WSDL) Documents

Web services are described in documents expressed in WSDL. When interacting with a web service, a process will adopt one of two roles - supplier or consumer.

A *service provider* publishes a WSDL document that describes the services it offers.

A *service consumer* uses the published WSDL document to determine the services offered by the supplier and the messages required to access those services.

Abstract and Concrete WSDL Documents

WSDL documents are described as either *abstract* or *concrete*:

- An *abstract WSDL* document describes what the web service does, but not how it does it or how to contact it. An abstract WSDL document defines:
 - the operations provided by the web service.
 - the input, output and fault messages used by each operation to communicate with the web service, and their format.
- A *concrete WSDL* document adds the information about how the web service communicates and where you can reach it. A concrete WSDL document contains the abstract WSDL definitions, and also defines:
 - the communication protocols and data encodings used by the web service.
 - the port address that must be used to contact the web service.

WSDL Document Requirements

If a process is to be deployed to the BPM runtime, any WSDL document used by that process must adhere to the requirements in the following sections. This applies whether the process is acting as a service consumer or as a service provider, and whether the WSDL is manually created, imported, or generated from a business object model.

See [Web Service Definition Language \(WSDL\) Documents](#) for more information about the content and structure of WSDL documents.

WSDL Version

TIBCO Business Studio and BPM support WSDL Version 1.1.

WSDL Version 2.0 is not supported.

Message Exchange Patterns

TIBCO Business Studio and BPM support the Message Exchange Patterns (MEP) shown in the table below. A web service operation must use one of these Message Exchange Patterns.

Supported WSDL Operation Message Exchange Patterns

MEP	Description	Required WSDL Operation Message sequence
One-way (In-only)	A client (service consumer) sends a message to the web service (service provider) but expects no response.	Input
Request-response (In-Out)	A client (service consumer) sends a request message to the web service (service provider). The web service (service provider) returns a response message to the client (service consumer). The web service may optionally return a fault message in the event of an error.	Input Output [Fault]

Message Parameter Mappings

WSDL operation messages can be mapped to the following types of process data parameter:

- Simple data types
- Arrays of simple data types
- Structured data types (including structured data types that contain further structured data types). These can only be used with **JavaScript** script grammar. The use of **XPath** script grammar with structured data types is not supported.
- Arrays of structured data types



An array can be mapped to an array, but an array element cannot be mapped to an array element. This applies whether the array contains simple data types or structured data types.

Data Transport Mechanism

The only supported data transport mechanisms are:

- Simple Object Access Protocol (SOAP) requests over Hypertext Transfer Protocol (HTTP) - (SOAP/HTTP).
- Simple Object Access Protocol (SOAP) requests over Java Message Service (JMS) - (SOAP/JMS).
- Virtualized service bindings.

SOAP versions

SOAP 1.1 and SOAP 1.2 bindings are supported when calling or exposing a service.

SOAP Binding Style

The only supported SOAP bindings are:

- Document/literal
- RPC/literal

Calling an ActiveMatrix BPM invocation fails due to a mismatch of binding and operation style if you use a WSDL from an ActiveMatrix BPM project (instead of one imported from ActiveMatrix Administrator), and then configure it using SOAP/HTTP binding.



Resolution:

- For SOAP Binding import the correct WSDL from ActiveMatrix Administrator.
- Check the configuration of the ActiveMatrix service for a mismatch of port types.

WSDL Documents and Schema Files

Schema type definitions can be embedded, included or imported in a WSDL document.

XSD Constructs

A number of XSD constructs are not supported by the BPM runtime. A process that will be deployed to the BPM runtime cannot use a WSDL that contains or references one of these constructs. An attempt to import a WSDL containing one of these constructs into TIBCO Business Studio may fail or display validation errors, depending on the particular construct used and the context in which it is used.

WSDL documents and XSD schema files, when used in TIBCO Business Studio, are mapped to and from business object models. For example:

- You can create a business object model and then export it as a WSDL document.
- When you import a WSDL document into the **Service Descriptors** folder, a corresponding business object model is automatically created in the **Generated Business Objects** folder.



For more detailed information about how WSDL documents and XSD schema files are mapped to and from business object models, and how particular XSD and WSDL constructs are represented in a business object model, see the *TIBCO Business Studio Modeling User's Guide*.

Unsupported Elements

- xsd:key
- xsd:keyref
- xsd:list
- xsd:redefine
- xsd:unique

Unsupported Built-in Datatypes

- xsd:NOTATION

Using Service Registries

If you plan to add a WSDL document to your project from a service registry, you can create a new service registry entry either before importing the WSDL document, or as part of the import process.

When you deploy a process that exposes a service operation to the BPM runtime, it is automatically published to the BPM runtime service registry. If you want to call the process from another process, you can therefore access its WSDL using this registry.



This section describes how to add a UDDI registry before importing the WSDL document.

To create a registry do the following steps.

Procedure

1. Select **New > Other** .
2. Expand **Services**, select **Service Registry** from the list of wizards and click **Next**.
3. Enter the details of the Web Service registry:
 - **Name** - enter the name that you want to be displayed for the registry in the Service Explorer.
 - **Query Manager (Inquiry) URL** - the URL used to retrieve information about the services and businesses of the registry.
 - **Lifecycle Manager (Publish) URL** - the URL used for publishing services and businesses to the registry.
4. Click **Finish**.
To view a registry do the following steps:
5. Select **Window > Show View > Other** .
6. Expand **Web Service Registries** and select **Registries**.
7. The Registries view opens and you should see any UDDI Registries that you have added.
To create a registry search do the following steps:
8. Click the Add Search button () or right-click the Registry and select **Add Search**
9. Select the type of search you want to perform (either for a business or for a service) and click **Next**.
10. Enter the service search criteria:
 - **Name**
This is the name you want displayed in the Registries view for your search.
 - **Search Criteria**
You can use a percent sign (%) as a wildcard to specify search criteria. For example, specifying **c%** would return all businesses or services that start with the character **c**.
11. Click **Finish**.
If the search is successful, the results are displayed in the Registries view. If the search is not successful, a message is displayed and you should check the error log for more details.
When you expand the search in the Registries view, the results are displayed.
Search results are preserved for subsequent browsing, but may be refreshed.
To change the properties of a registry or search do the following steps.
12. Do one of the following:
 - Double-click the Registry or Search.
 - Right-click the Registry or Search and select **Properties**.
 - Select the Registry or Search and select **File > Properties**.
13. From the resulting Properties dialog, change the settings as necessary, then:
 - Click **Apply** to effect any changes you have made.
 - Click **OK** to exit the dialog.
 - Click **Cancel** to exit the dialog without applying your changes.

Importing a WSDL Document Into a Project

Procedure

1. Right-click the **Service Descriptors** folder into which you want to import the WSDL document and select **Import > Service Import Wizard**.
2. Select one of the following import methods:
 - **Import from a File** Use this method to browse the file system for the WSDL document.
 - **Import from a URL** Use this method to specify a URL that resolves to the location of the WSDL document.
 - **Import from a UDDI Registry** Use this method to select a WSDL document from a UDDI registry.



The **Descriptor for XML over JMS** option is not supported by the BPM runtime. Do not use it.

3. Click **Next**. If you chose:
 - **Import from a File**, browse to specify the **Location** of the WSDL document.
 - **Import from a URL**, enter the URL for the WSDL document.
 - **Import from a UDDI Registry**, the dialog lists any existing UDDI registries that you have added (see Creating a Registry on page 5). Expand a registry and select a WSDL.
- 

To add a new registry, right-click in the blank area of the dialog and select **Add Registry**. You can also add registry searches in this dialog using the right-click menus.
4. Click **Next**.
 5. Browse to select the **Project > Location** (the folder in your project where you want to store the WSDL document), and if necessary change the name of the WSDL document. Select the **Overwrite existing resources** checkbox if you want to replace any existing WSDL documents with the same name.
 6. Click **Finish**.

Calling a Web Service

This section describes some general points that you need to be aware of when implementing a call to a web service from a process, and how to call a web service from a process.

[Using Web Services](#) provides general information about using web services with processes.

See also the following tutorials:

- *How to Call an External Web Service From a Process*
- *How to Call a Secured External Web Service From a Process*
- *How to Call a Virtualized Service from a Process*
- *Using Credential Mapping to Associate a Specific Identity with a Process Instance*

A process can invoke web service operations provided by other processes or applications. In this case, the process acts as the service consumer in the conversation.

Service Types

A process can call two types of service:

- an *internal service*, provided by an application hosted in the BPM runtime. The application being called can itself be another BPM process that is exposing one or more service operations.
- an *external (web) service* provided by an external application (that is, an application that is not hosted in the BPM runtime).

Service Bindings and WSDLs

A process can call a service whose endpoint is exposed on the following types of binding:

- a *virtualization binding*. A virtualization binding can be provided only by an internal service (one hosted in the BPM runtime).

You must use an abstract WSDL to call a service exposed on a virtualization binding.

- a *SOAP binding*. A SOAP binding can be provided either by an external service or by an internal service.

You must use a concrete WSDL to call a service exposed on a SOAP binding.



If you need to enforce a security policy when calling an internal service, you must use a concrete WSDL to call the service on its SOAP binding. See [Configuring Security on an Outgoing Service Call](#) .)

Service Development - Contract First or Contract Last

TIBCO Business Studio allows you to use either a contract first or contract last approach to developing the service call, depending on which suits your requirements:

- *Contract first* (or top-down): You obtain the WSDL that defines the service contract from the service provider, then configure the process to send and receive the appropriate data.
- *Contract last* (or bottom-up): You first define the process data that you want to send and receive, then generate a WSDL that defines the service contract. The service provider must then implement the service to send and request that data.

How to Call a Web Service

The procedure you need to use to call a web service depends upon two things:

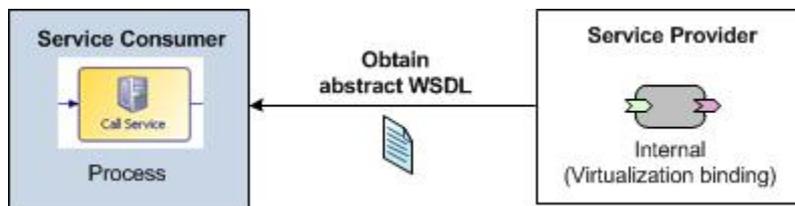
- whether you are using a contract first or contract last approach to developing the service call. (See [Service Development - Contract First or Contract Last](#).)
- whether you need to call the service on a virtualization binding or on a SOAP binding. (See [Service Bindings and WSDLs](#).)

Use the following table to choose the correct procedure to follow.

Service Development	Service Binding	Procedure
Contract first	Virtualization	See Calling a Service on a Virtualization Binding (Contract First) .
	SOAP	See Calling a Service on a SOAP Binding (Contract First) .
Contract last	Virtualization	See Calling a Service on a Virtualization Binding (Contract Last) .
	SOAP	See Calling a Service on a SOAP Binding (Contract Last) .

Calling a Service on a Virtualization Binding (Contract First)

Use the following procedure when you want to call an internal service that is exposed on a virtualization binding, and obtain the WSDL that defines the service contract from the service provider, then configure the process to send and receive the appropriate data.



Step	Task description	For more information, see...
1.	Configure an activity to call the web service.	Configuring a Task or Event to Call a Web Service
2.	Select or import an abstract WSDL supplied by the service provider, then select the web service operation that you want to invoke.	Selecting the Web Service Operation to Invoke
3.	Define the process data that you want to send to and receive from the web service.	Defining Input and Output Data
4.	Map the input/output parameters required by the web service operation to corresponding data fields or parameters in the process.	Defining Input and Output Mappings
5.	(Optionally) Modify the process to catch and handle any fault messages returned by the web service.	Catching WSDL Fault Messages on a Request-Response Operation

Step	Task description	For more information, see...
6.	Deploy the project to the BPM runtime.	Deploying a Process That Calls a Web Service

Calling a Service on a SOAP Binding (Contract First)

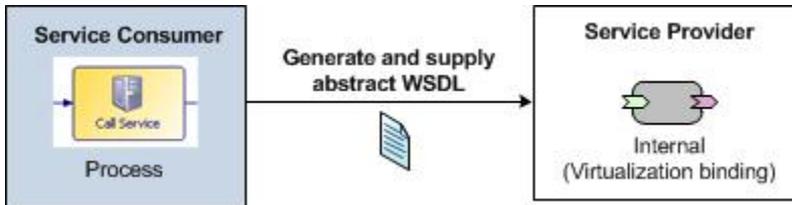
Use the following procedure when you want to call a service (internal or external) that is exposed on a SOAP binding, and obtain the WSDL that defines the service contract from the service provider, then configure the process to send and receive the appropriate data.



Step	Task description	For more information, see...
1.	Configure an activity to call the web service.	Configuring a Task or Event to Call a Web Service
2.	Select or import a concrete WSDL supplied by the service provider, then select the web service operation that you want to invoke.	Selecting the Web Service Operation to Invoke
3.	(If required) Configure the system participant to apply security policies to the outgoing service call.	Using a System Participant to Identify the Web Service Endpoint
4.	Define the process data that you want to send to and receive from the web service.	Defining Input and Output Data
5.	Map the input/output parameters required by the web service operation to corresponding data fields or parameters in the process.	Defining Input and Output Mappings
6.	(Optionally) Modify the process to catch and handle any fault messages returned by the web service.	Catching WSDL Fault Messages on a Request-Response Operation
7.	Deploy the project to the BPM runtime.	Deploying a Process That Calls a Web Service

Calling a Service on a Virtualization Binding (Contract Last)

Use the following procedure when you want to call an internal service that is exposed on a virtualization binding, and define the process data that you want to send and receive, then generate a WSDL that defines the service contract.



Step	Task description	For more information, see...
1.	Define the process data that you want to send to and receive from the web service.	Defining Input and Output Data
2.	Configure an activity to call the web service.	Configuring a Task or Event to Call a Web Service
3.	Generate an abstract WSDL, then select the web service operation that you want to invoke.	Selecting the Web Service Operation to Invoke
4.	Map the input/output parameters required by the web service operation to corresponding data fields or parameters in the process.	Defining Input and Output Mappings
5.	(Optionally) Modify the process to catch and handle any fault messages returned by the web service.	Catching WSDL Fault Messages on a Request-Response Operation
6.	Deploy the project to the BPM runtime.	Deploying a Process That Calls a Web Service
7.	Make the generated abstract WSDL available to the service provider.	

Calling a Service on a SOAP Binding (Contract Last)

Use the following procedure when you want to call a service (internal or external) that is exposed on a SOAP binding, and define the process data that you want to send and receive, then generate a WSDL that defines the service contract.



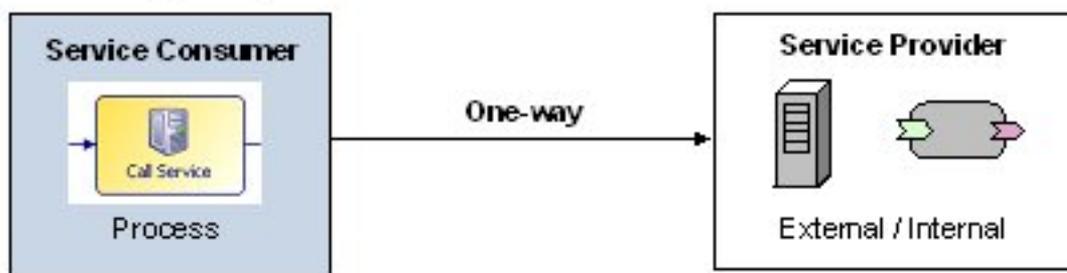
Step	Task description	For more information, see...
1.	Define the process data that you want to send to and receive from the web service.	Defining Input and Output Data
2.	Configure an activity to call the web service.	Configuring a Task or Event to Call a Web Service
3.	Generate a concrete WSDL, then select the web service operation that you want to invoke.	Selecting the Web Service Operation to Invoke
4.	(If required) Configure the system participant to apply security policies to the outgoing service call.	Using a System Participant to Identify the Web Service Endpoint
5.	Map the input/output parameters required by the web service operation to corresponding data fields or parameters in the process.	Defining Input and Output Mappings
6.	(Optionally) Modify the process to catch and handle any fault messages returned by the web service.	Catching WSDL Fault Messages on a Request-Response Operation
7.	Deploy the project to the BPM runtime.	Deploying a Process That Calls a Web Service
8.	Make the generated concrete WSDL available to the service provider.	

Configuring a Task or Event to Call a Web Service

A process can invoke two types of operation - **one-way** and **request-response**.

Invoking a One-Way Operation on a web service

A process can invoke a one-way operation to send a message to the web service, without receiving a response.



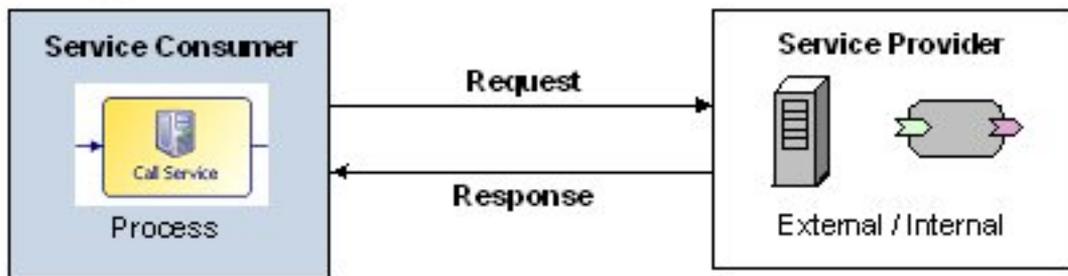
Procedure

1. Select the task or event that you intend to use to send a message to the web service. This must be one of the following types:
 - a send task
 - a throw intermediate message event

- a message end event
2. On the **General** tab of the **Properties** view:
 - a) Click **Send One Way Request**.
 - b) Set **Service Type** (for a send task) or **Implementation** (for a message event) to **Web Service**.

Invoking a Request-Response Operation on a web service

A process can invoke a **request-response** operation to send a request message to the web service and receive a response message back from it. (The response may be a fault message.)



Procedure

1. Select the service task that you intend to use to send a message to and receive a response back from the web service.
2. On the **General** tab of the **Properties** view, set **Service Type** to **Web Service**.

Selecting the Web Service Operation to Invoke

Procedure

1. Select the task or event that you will use to call the web service. On the **General** tab of the Properties view, the following buttons are shown against the **Operation** field.

2. Click one of these buttons, depending on what you want to do. The following table describes each available option.

Button	Usage
Select	Choose an operation from a WSDL that already exists in your workspace. See Selecting an Operation From a WSDL That Exists in the Workspace .
Import WSDL	Import a WSDL from an external source (a file, URL or service registry), then choose an operation from the imported WSDL. See Importing a WSDL and Selecting an Operation from the WSDL .
Generate	Automatically create an abstract WSDL, an operation and the required data mappings from the data fields and types defined on the Interface tab. See Generating a WSDL and Creating an Operation from your Process Data .
Clear	Clear the current selections in the Operation and Endpoint resolution sections.

Selecting an Operation From a WSDL That Exists in the Workspace

Procedure

1. Select the task or event that you will use to call the web service.
2. On the **General** tab of the Properties view, click **Select**. The Operation Picker dialog is displayed:
 - The dialog lists every web service operation that is available in a WSDL file (in either a **Service Descriptors** or **Generated Services** folder) in any project in your workspace.
 - The status line in the dialog shows the project location and filename of the WSDL that contains the currently selected operation.
3. Select the operation that you want to invoke and click **OK**.

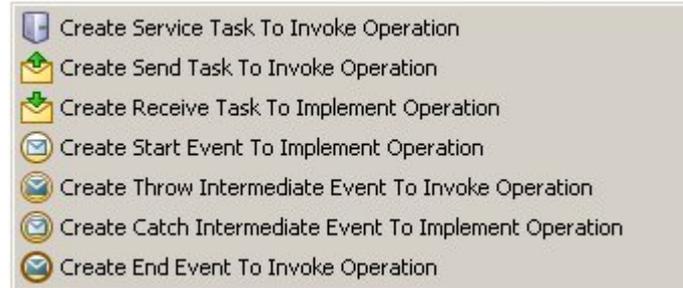
Result

TIBCO Business Studio now automatically performs the following tasks:

- It creates a system participant to identify the web service endpoint to be called. See [Using a System Participant to Identify the Web Service Endpoint](#).
- It populates the **Operation** and **Endpoint resolution** sections of the selected activity with the relevant service details from the WSDL. (See [Web Service Implementation Properties](#).)

As a shortcut, when you have the WSDL in your workspace you can create a task or event to call an operation using the following method:

- Expand the WSDL in Project Explorer and select the operation you want to invoke.
- Drag the operation to the point in your process flow where you want to invoke the web service operation. A context-sensitive menu is displayed, listing the tasks and events you can use to invoke the selected operation. For example:



- Select the appropriate option from the menu. The corresponding task or event is created and configured. The **Operation** and **Endpoint resolution** sections are populated with the relevant service details from the WSDL.

Importing a WSDL and Selecting an Operation from the WSDL

Procedure

1. Select the task or event that you will use to call the web service. On the **General** tab of the Properties view, click **Import WSDL**. The **WSDL Import Wizard** is displayed.
2. Select one of the following import methods:
 - **Import from a File** - to browse the file system for the WSDL document.
 - **Import from a URL** - to specify a URL that resolves to the location of the WSDL document.
 - **Import from a UDDI Registry** - to select a WSDL document from a UDDI registry.



The **Descriptor for XML over JMS** option is not supported by the BPM runtime. Do not use it.

3. Click **Next**. If you chose:
 - **Import from a File**, browse to specify the **Location** of the WSDL document.
 - **Import from a URL**, enter the URL for the WSDL document.
 - **Import from a UDDI Registry**, select the WSDL document from the list of registries and registry searches. (See [Using Service Registries](#).)
4. Click **Next**. The **Destination Selection** page is displayed.
5. Browse to select the **Project > Location** (the project folder where you want to store the WSDL document), and if necessary change the name of the WSDL document. Select the **Overwrite existing resources** checkbox if you want to replace any existing WSDL document with the same name.
6. Click **Next**. The **Operation Picker** page is displayed. This shows the WSDL files available in the selected destination project.
7. Select the operation you want to invoke and click **OK**.

Result

TIBCO Business Studio now automatically performs the following tasks:

- It creates a system participant to identify the web service endpoint. See [Using a System Participant to Identify the Web Service Endpoint](#).
- It populates the **Operation** and **Endpoint resolution** sections of the selected activity with the relevant service details from the WSDL. (See [Web Service Implementation Properties](#).)

Generating a WSDL and Creating an Operation from your Process Data



Before you generate a WSDL, use the Interface tab to define the data fields (and types) that you want to use to exchange data with the web service - see [Defining Input and Output Data](#).

Procedure

1. Select the task or event that you will use to initiate the call to the web service. (This must be either a service task, or a throw intermediate message event, end message event or send task that is configured as a one-way operation.)
2. On the **General** tab of the Properties view, click **Generate**. The Create WSDL for Activity dialog is displayed:
 - a) Select the project folder where you want to store the generated WSDL file. The default option is **Service Descriptors**. (If your process contains multiple service tasks that use generated WSDLs, each task uses its own WSDL.)
 - b) Enter a name for the WSDL file. (The default name is *ProcessName-TaskName*.)
 - c) Click **Next**.
 - d) If desired, edit the **Target namespace**.
 - e) Select the WSDL **Binding Style** that you want to use - either **Document Literal** (the default option) or **RPC Literal**.
 - f) Select the **WSDL Type** that you want to generate - either **Abstract** (the default option) or **Concrete**.
 - g) If you selected **Concrete**, change the **SOAP Address** to the URI that will be used to contact the service at runtime, in the format *protocol://host:port/path*.



You can use any values for the *protocol*, *host* and *port* components of the address, as these values will be ignored at runtime.

At runtime, the *protocol*, *host* and *port* components of the address will be supplied from the configuration of the HTTP Client resource template that is referenced by the system participant's [HTTP Client Instance](#) property.

- h) Click **Finish**.

Result

TIBCO Business Studio now automatically performs the following tasks:

- It creates the WSDL file.
- It creates a system participant to identify the web service endpoint. See [Using a System Participant to Identify the Web Service Endpoint](#).
- It populates the **Operation** and **Endpoint resolution** sections of the selected activity with the relevant service details from the WSDL. (See [Web Service Implementation Properties](#).)



The **Operation Name** and **Port Type** are named, respectively, after the name of the service task and the process.

- It creates input and output mappings based on the data defined on the **Interface** tab of the selected activity. See [Defining Input and Output Mappings](#).

Updating a Generated WSDL File

If you change the data on the **Interface** tab after you have generated the WSDL file, you can click **Generate** again to update the WSDL file at any time.

This will also automatically regenerate the input and output mappings. (However, if you have made any manual alterations to these mappings, these will be lost and you will need to redo them.)

Using a System Participant to Identify the Web Service Endpoint

A system participant is a logical identifier for a connection to an external system - in this case, a web service endpoint. An endpoint defines the URL that will be used to contact the web service.

A task or event that calls a web service must use a system participant that identifies the endpoint of the web service that is to be invoked. This information is used at runtime to map the call to the web service.

When you select or import a concrete WSDL operation binding () or select, import or generate an abstract WSDL operation () a system participant is automatically created and assigned to the calling task or event's **Endpoint Name**. (The system participant's name is taken from the portType of the chosen operation.)

See [System Participant Shared Resource Properties](#) for more information about the configuration of this system participant.

Configuring Security on an Outgoing Service Call

If you need to apply a security policy on the outgoing service call, you do so by assigning a policy to the system participant that identifies the service endpoint.

You may need to do this, for example:

- to invoke a secured external web service. See the *How to Call a Secured External Web Service From a Process* tutorial for more information.
- to enforce credential mapping to ensure that a process instance always runs using fixed credentials. See the *Using Credential Mapping to Associate a Specific Identity with a Process Instance* tutorial for more information.

The security policy will then be applied to the outgoing message sent by the task or event, allowing it to be authenticated by the called service.



The service must be invoked using a SOAP binding (with a concrete WSDL). You cannot apply a security policy if you are calling the service on its virtualization binding (using an abstract WSDL).

To assign a security policy to the system participant:

Procedure

1. In Project Explorer, select the system participant that identifies the service endpoint.
2. On the **General** tab of the Properties view, expand **Shared Resource**. The endpoint's configuration details are displayed.

▼ Shared Resource...

E-mail
 Jdbc
 Web Service Consumer
 Web Service Provider

Binding Details: Virtualization SOAP over HTTP SOAP over JMS
 Binding Details:
 HTTP Client Instance:
 Security Configuration:
 Policy Type:

3. In the **Policy Type** field, select the type of security policy required to invoke the service from the drop-down menu - one of:

- **Username Token, X509 Token or SAML Token**, to authenticate the outgoing SOAP request using a Web Services Security (WSS) token of the indicated type.
- **Custom Policy**, to apply a custom security policy to the outgoing SOAP request and, if required, to the incoming SOAP response.



You must use a **Custom Policy** if the SOAP response message returned by the service contains a security header. The **Username Token, X509 Token or SAML Token** policies do not handle an incoming SOAP response that contains a security header.

See [SOAP over JMS Binding Details \(Provider\)](#) or [SOAP Over JMS Binding Details \(Consumer\)](#) for more information about these policy types.

4. If you selected **Username Token, X509 Token or SAML Token**, a **Governance App. Name** field is displayed. Enter the name of the identity provider application from which the BPM runtime will obtain the authentication information needed to contact the service.
5. If you selected **Custom**, a **Custom Policy Set** field is displayed:
 - a) Click the Browse button. The Select Policy Set dialog is displayed, listing all external policy sets that are available in the current workspace.



The external policy set file that defines the policy to be used must be available in the same workspace. (It does not have to be in the same project.)

If the required policy set file is not already available, click **Cancel**, import the file to the workspace and try again.

- b) Select the policy set that the BPM runtime will apply to the outgoing SOAP request (and, if appropriate, to the incoming SOAP response).
- c) Click **OK**.

Defining Input and Output Data

You use the **Interface** tab to define the subset of data fields and/or formal parameters defined in the process that are available to the task or event being used to invoke the web service operation.

The selected fields will appear on the **Input to Service, Output from Service or Output from Process** tabs, where they can be mapped to corresponding service input/output parameters - see [Defining Input and Output Mappings](#).

If the task or event is using an existing WSDL (either selected or imported), the fields/parameters selected on the **Interface** tab define the fields that are available for mapping on the **Input to Service** and **Output from Service** mapping tabs.



If the task or event is using a WSDL generated from the process, the fields/parameters selected on the **Interface** tab act as a filter to control the parts that are created in the WSDL operation.

By default, all fields/parameters defined in the process are available.

To delete all the fields/parameters from the list of those available, use the checkbox **No interface association required**.

Click  to add new fields. The Select Data Field or Formal Parameter dialog is displayed, listing the available data fields and formal parameters that are defined in the process and/or package.

Procedure

1. Select the data fields or parameters you need, click **Add**, then click **OK**. The selected fields are displayed on the **Interface** tab.
2. For each field, click in the **Mode** cell and select the appropriate value from the drop-down list:

- **In** - defines a field whose value will be sent to the web service.
- **Out** - defines a field that will be used to store a value returned from the web service.
- **In/Out** - defines a field whose value will be sent to the web service, and then updated with a value returned from the web service.

Defining Input and Output Mappings

When you are invoking a web service operation, you must map the appropriate input/output parameters provided by the web service operation to the appropriate parameters and/or data fields in the process.

On the Properties view for the relevant task or event, the **Input to Service**, **Output from Service** or **Output from Process** tab (as appropriate) provides a Mapper tool that allows you to easily perform the required mappings.



If you have generated a WSDL and operation (see [Generating a WSDL and Creating an Operation from your Process Data](#)), service input/output parameters and mappings are automatically created (based on the data defined on the Interface tab). You do not have to manually create them.

Creating a Mapping

The Mapper automatically populates the left-hand and right-hand sides of the tab with the appropriate data.

On the **Input to Service** or **Output from Process** tab:

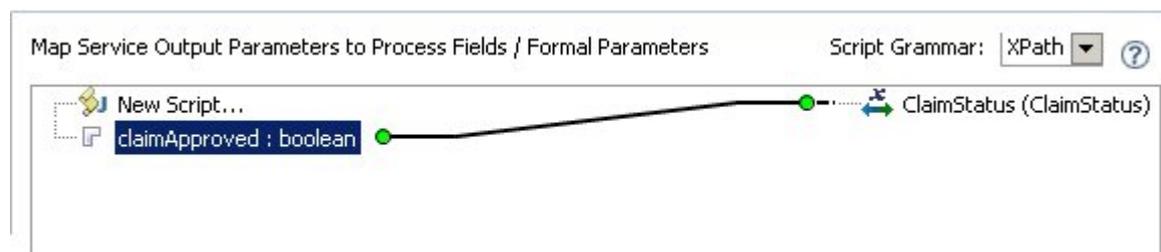
- The left-hand side displays the parameters and data fields available to this task or event, as defined on the **Interface** tab.
- The right-hand side displays the service input or output parameters defined for the selected operation (in the WSDL document) that is being called.

On the **Output from Service** tab (which is only displayed for a service task invoking a request-response operation):

- The left-hand side displays the service input or output parameters defined for the selected operation (in the WSDL document) that is being called.
- The right-hand side displays the parameters and data fields available to this service task, as defined on the **Interface** tab

To perform a mapping, simply drag and drop the parameter or data field that you want to map from one side onto the appropriate service input or output parameter on the other side. A mapping is created between the two entities.

The following example shows the **Output From Service** tab for a service task. The **claimApproved** parameter, which is the only output parameter defined for the selected operation in the associated WSDL document, is mapped to the process parameter **ClaimStatus**. This field will therefore store the **claimApproved** value returned by the web service.



Points to Note About Mappings

TIBCO Business Studio validates the mappings as you create them, and displays error markers to indicate any problems.

For example:

- Every mandatory service parameter defined in the WSDL must be mapped to a process data field or formal parameter.
- All mappings must be from and to equivalent data types - you cannot, for example, map a data field defined as an integer to a service input parameter defined as a date.

See [Message Parameter Mappings](#) for more information about the different types of data mappings that are supported.

If the available process data fields do not provide the necessary data for a particular mapping, you can use the **Script Editor** create a script that manipulates the data fields, and map the script to the service parameter.

Using A Script to Define a Mapping

You can use the Script Editor to modify mappings from process data to service input/output parameters or fault messages.



For simple transfer of data between different process data, it may be easier to create scripts with Data Mapper instead of writing them in JavaScript. See [Data Mapping](#) and [Mapping Contents in Data Mapper](#).

Procedure

1. Select the **Script Grammar** that you wish to use, either:
 - **JavaScript**, to enter a JavaScript script.
 - **XPath**, to enter an XML Path Language expression.
2. Click **New Script** in the appropriate mapping tab.
3. Enter your script in the **Script Editor**.

Scripts are validated as you enter them. Validation markers are displayed if any errors occur.

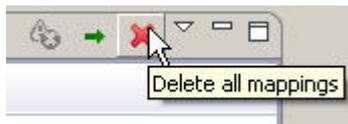
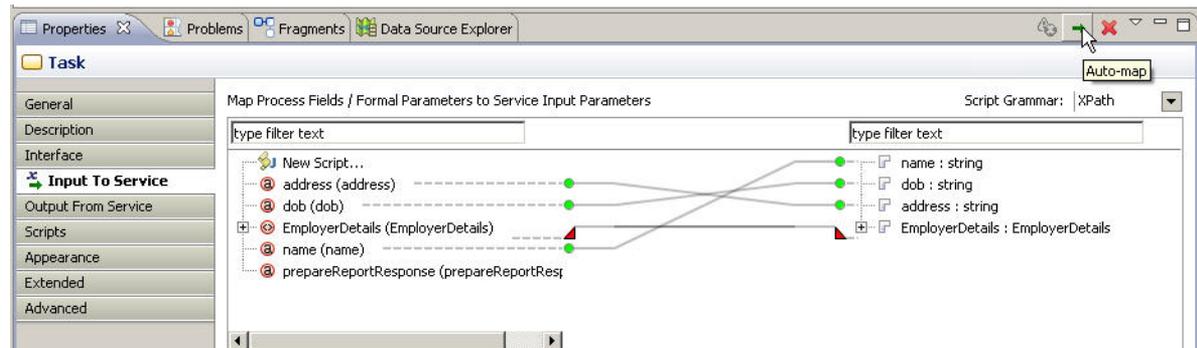
You can also use content assist in the script Editor by pressing **Ctrl + Space**.
4. When you have finished, click outside of the **Script Editor**. A script called **Script_n** is displayed in the tab. (You can rename it by right-clicking the script name, and selecting **Rename Script**.)
5. Map the script by dragging it from the left side of the mapper to the appropriate entity on the right-hand side.

Automapping

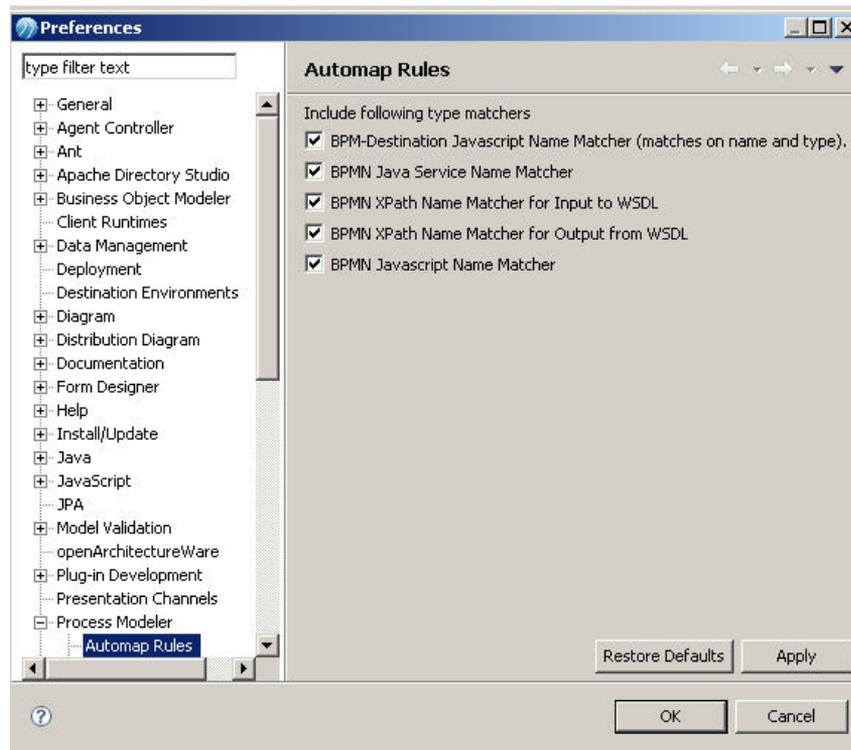
Automapping is a feature that attempts to map parameters based on their name and type. When you select automapping, mappings between items are performed automatically.

There is a button available above the mapping on the **Input To Service**, **Output From Service**, **Input To Process**, **Output From Process**, **Map To Sub-Process**, **Map From Sub-Process**, **Map From Error** and **Map From Signal** tabs to allow you to select automapping.

To delete all the mappings, click the **Delete** button. This is irrelevant if they were drawn using the auto-map tool or in some other cases.



Automapping rules can be tailored so that you can choose whether you need automapping from different providers by selecting **Window > Preferences > Process Modeler > Automap Rules**.



The providers are grammar specific and are responsible for matching the source and target of the mapping ends:

- There is one Java Service Name Matcher Provider.
- For XPath, there are two providers: one for name matching in the input to the WSDL and other for name matching on the output to the WSDL.
- For Javascript, there are two providers: one default BPMN provider which does name matching and the other from the BPM destination which does name and type matching.

You can deselect a rule, and then when you use Automapping, it will not be used.

Catching WSDL Fault Messages on a Request-Response Operation

If you are using a service task to invoke a request-response operation, you can use catch intermediate error events to catch any fault messages that are returned by the web service.

You cannot catch fault messages:

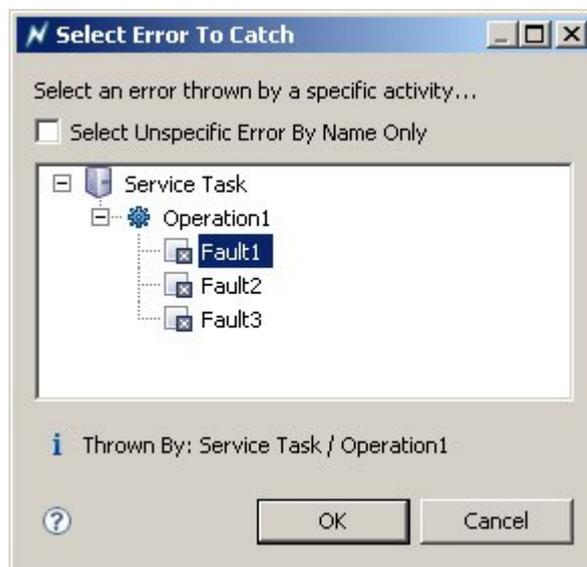
- if you are using a service task with an automatically generated WSDL (see [Generating a WSDL and Creating an Operation from your Process Data](#)), unless you have manually added a fault message to the WSDL after you have generated it.
- when you invoke a one-way operation using an end task, a throw intermediate message event or a message end event. Fault messages are not supported by this MEP (see [Message Exchange Patterns](#)).



Using a Catch Intermediate Error Event to Catch a Fault Message

Procedure

1. Add a catch intermediate error event to the boundary of the service task that is invoking the web service operation.
2. Select the catch intermediate event.
3. On the **General** tab of the **Properties** view:
 - a) Enter a suitable **Label** for the event.
 - b) Click **Select Error** (to the right of the **Catch Error Code** field). The Select Error to Catch dialog is displayed. This lists the fault messages which can be returned by the operation being invoked. For example:

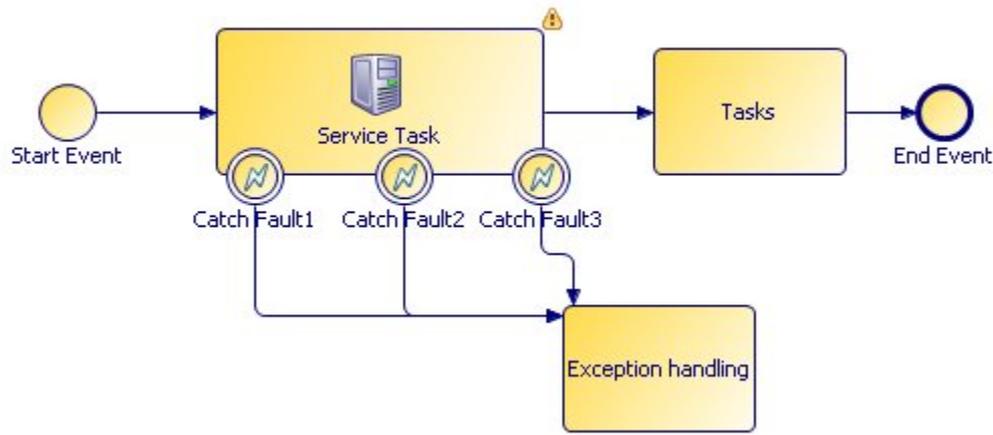


- c) Select the fault message you want to catch, then click **OK**. The **Catch Error Code** and **Thrown By** fields contain the details of the fault message.
4. On the **Map From Error** tab:
 - The left-hand side displays the fault message parameters defined for the selected fault message, which will contain the data returned about the error.
 - The right-hand side displays the formal parameters and data fields available to this service task, as defined on the **Interface** tab.

Drag and drop the fault message parameter(s) onto the appropriate formal parameters or data fields.

Catching Errors Example

The following process illustrates one way to catch the errors.



Service Task invokes the request-response operation provided by the process described in [An Example](#). That operation includes three separate fault messages, **Fault1**, **Fault2** and **Fault3**.

Three catch intermediate error events - **Catch Fault1**, **Catch Fault2** and **Catch Fault3** - are added to **Service Task** to catch each fault.

Each event:

- is configured to catch the appropriate error - **Fault1**, **Fault2** or **Fault3** - in the **Catch Error Code** field.
- maps the data from the fault message parameter to appropriate process data fields on the **Output Fault From Process** tab.

The process then provides whatever exception handling logic is required to process these errors.



An alternative approach would be to use a single catch intermediate error event, set to **Catch All** instead of catching each specific error. See "Throw and Catch Error Events" in the *TIBCO Business Studio Modeling User's Guide* for more information about how to throw and catch errors.

Handling SOAP/JMS Message Timeouts on a Request-Response Operation

If you are using a service task to invoke a request-response operation over SOAP/JMS, you can configure JMS message timeouts to prevent the service task from hanging if the called web service does not respond in a timely fashion. These timeouts can also prevent duplicate messages from being submitted if the receiver was not able to receive and process the initial message(s) - for example, when communication to a mainframe system is intermittent.

You can configure SOAP/JMS message timeout behavior by using the following properties of the service task's endpoint:

- **Request-Response Timeout** (default 6 seconds): defines the time period after which a web service call will time out if a response message has not been received from the web service. If this occurs, a **Timeout exception** error is thrown by the web service task.
- **Request Expiration Timeout** (default 3 seconds): defines the time period within which the called web service must pull the *request* message from the JMS message queue. If this timeout expires, the

JMS server will be instructed to purge the request message from the JMS message queue. (This ensures that if the web service call is retried, the JMS message queue does not contain duplicate copies of the same request message for the web service to consume.) The web service call itself does not timeout when this timer expires, so no **Timeout exception** error is thrown.



If you migrate a pre-4.0 version of a process to this version, default values are *not* automatically set for these properties. You must set them manually.

Procedure

1. In Project Explorer, select the **Participant** used by the web service task to invoke the SOAP/JMS web service operation.
2. On the **General** tab of the Properties view, in the **Message Configuration** section:
 - a) Set the **Request-Response Timeout** to an appropriate value (in seconds) for the web service that you are calling.
3. Optionally:
 - a) Add a catch intermediate error event to the boundary of the web service task and configure it to catch the **Timeout exception** error thrown if the **Request-Response Timeout** expires.

You can delete this property or set it to 0, so that the web service call will wait indefinitely until the response message is received from the called web service. This is not recommended, because if the called web service is slow to respond or does not respond at all, the process instance will be stuck. A validation warning is displayed if you do this.

- b) Set the **Request Expiration Timeout** to an appropriate value (in seconds) for the web service that you are calling. This value must be less than the value of the **Request-Response Timeout**.

- a) Add a catch intermediate error event to the boundary of the web service task and configure it to catch the **Timeout exception** error thrown if the **Request-Response Timeout** expires.

See [Using a Catch Intermediate Error Event to Catch a Fault Message](#).

- b) Add appropriate error handling to your process flow to deal with the timeout error.

If you do not use a catch intermediate error event, the web service task will fail with **TimeoutException** error. The Process Instance will halt or fail based on process implementation. Using the Force special action on halted process instance(s) from Openspace Process Views, you can progress halted processes. However, you cannot progress process instances that are in a failed state. For more information about progressing halted process instances, see "Progressing Halted Processes" in *TIBCO ActiveMatrix® BPM Openspace User's Guide*.

Deploying a Process That Calls a Web Service

When you deploy a project that calls a web service, the method you must use depends on the type of web service call you are making.

Calling a Service on a Virtualization Binding

If your process calls the service on a virtualization binding, the promoted reference in the composite application (representing the service call) must be wired to the appropriate service virtualization binding of the application that provides the service.

You can perform this wiring by using the **Wiring Configuration** page of the **DAA Deployment Wizard** wizard. For more information see [Using Pageflow Processes and Business Services](#).

Alternatively, you can export the project to a Distributed Application Archive (DAA), then use the Administrator interface in the BPM runtime to perform the wiring and deploy the application. See the Administrator interface documentation for your BPM runtime environment for more information.

Calling a Service on a SOAP Binding

If your process calls the service on a SOAP binding, when you deploy the project, you must bind the system participant to the appropriate resource instance(s) in the BPM runtime:

- The system participant logically identifies the web service to be called.
- Resource instances define the actual runtime connection to the target web service.

You can perform this binding using the **Property Configuration** page of the **DAA Deployment Wizard** wizard. For more information see the following references:

- [SOAP over JMS Binding Details \(Provider\)](#)



A binding is made for each system participant, so if multiple tasks or events use the same system participant to call the same web service, they will all use the same service endpoint.

- [SOAP Over JMS Binding Details \(Consumer\)](#)

Alternatively, you can export the project to a Distributed Application Archive (DAA), then use the Administrator interface in the BPM runtime to perform the binding and deploy the application. See the Administrator interface documentation for your BPM runtime environment for more information.

Exposing a Web Service

This section describes some general considerations to be aware of when exposing a web service.

It follows on from [Using Web Services](#), which provides general information about using web services with processes.

See also the *How to Expose a Web Service From a Process* tutorial.

A process can expose a web service operation that other processes or applications can invoke. In this case, the process acts as the service provider in the conversation.

Service Bindings and WSDLs

A process can expose a service using the following types of binding on the service endpoint:

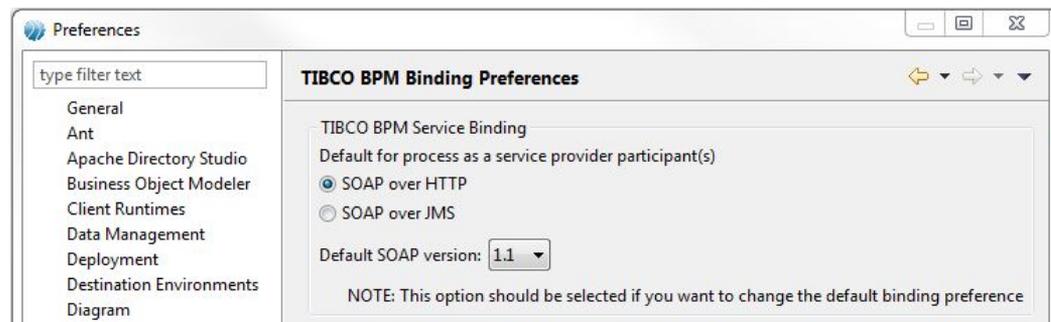
- a *virtualization binding*. A virtualization binding can only be accessed by an internal client application (one hosted in the BPM runtime). A virtualization binding is exposed whether you use an abstract WSDL or a concrete one.
- a *SOAP binding*. A SOAP binding can be accessed either by an external client (an application that is not hosted in the BPM runtime) or, if required, by an internal client. You must use a concrete WSDL to expose a service on a SOAP binding.



SOAP binding can be exposed over HTTP or JMS.

- You can use the TIBCO BPM Binding Preferences page (**Window > Preferences > TIBCO BPM Binding Preferences**) to set the default binding preferences for a service as a process provider participant. The default is SOAP over HTTP using SOAP version 1.1.

This applies to a generated WSDL, and to an imported abstract WSDL. If you import a concrete WSDL then this will be ignored and the binding will be the one defined in the WSDL.



Service Development - Contract First or Contract Last

TIBCO Business Studio allows you to use either a contract first or contract last approach when exposing a service, depending on which suits your requirements:

- *Contract first* (or top-down): You first obtain the WSDL that defines the service contract from the service consumer, then configure the process to receive and return the appropriate data.
- *Contract last* (or bottom-up): You first define the data that the process will receive and return, then generate a WSDL that defines the service contract. The client application must then conform to that contract when it calls the service.

Application Upgrade

When a process exposes a web service operation, a WSDL defines the service interface to that operation. The WSDL can be either imported or automatically generated by TIBCO Business Studio.

Once the application containing the process has been deployed to the BPM runtime, it can only be subsequently upgraded if its service interface (defined by the WSDLs used to expose its services) has not changed. (See "Upgrading a Deployed Application" in the *TIBCO ActiveMatrix BPM Deployment Guide*.)

If a process uses a **generated** WSDL, making changes to the project may result in changes to the generated WSDL, meaning that the application cannot be upgraded. See [Making Changes to the Service Interface](#) for more information about interface changes and about best practice in making and managing them.

Exposing the Web Service operation as a REST Service

You can expose a business process as a REST service for use via the REST API. See [Exposing the Web Service Operation as a REST Service](#) for more information.

Exposing a Service

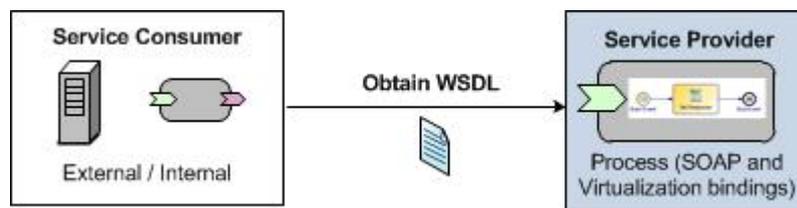
The procedure you need to use to expose a web service depends on the approach you are taking to develop the service call: contract first or contract last.

See [Exposing a Service \(Contract First\)](#)

See [Exposing a Service \(Contract Last\)](#)

Exposing a Service (Contract First)

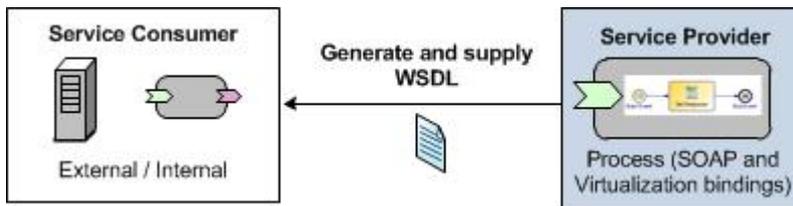
Using contract first to expose a service involves obtaining the WSDL that defines the service contract from the service consumer, then configuring the process to receive and send the appropriate data.



Step	Task description	For more information, see...
1.	Configure an activity to expose the web service.	Configuring a Task or Event to Expose a Web Service
2.	Select or import an abstract or concrete WSDL supplied by the service consumer, then select the web service operation that you want to expose.	Selecting an Alternative Web Service Operation to Expose
3.	Define the process data that you want to receive from and return to the service consumer.	Defining Input and Output Data
4.	Map the input/output parameters required by the web service operation to corresponding data fields or parameters in the process.	Defining Input and Output Mappings
5.	Deploy the project to the BPM runtime.	Deploying a Process That Exposes a Web Service

Exposing a Service (Contract Last)

Using contract last to expose a service involves defining the process data that you want to receive and send, then generating a WSDL that defines the service contract.



Step	Task description	For more information, see...
1.	Define the process data that you want to receive from and return to the service consumer.	Defining Input and Output Data
2.	Configure an activity to expose the web service.	Configuring a Task or Event to Expose a Web Service
3.	Generate an abstract WSDL, then select the web service operation that you want to expose.	Using the Default Generated Web Service Operation
4.	Map the input/output parameters required by the web service operation to corresponding data fields or parameters in the process.	Defining Input and Output Mappings
5.	(If required) Review the configuration of the SOAP binding and endpoint.	System Participant Shared Resource Properties
6.	Deploy the project to the BPM runtime.	Deploying a Process That Exposes a Web Service
7.	(If required) Generate the concrete WSDL that will be used to expose the service on its SOAP binding.	Administrator interface documentation for your BPM runtime environment
8.	Make the appropriate generated WSDL (abstract or concrete) available to the service consumer.	Calling a Service on a Virtualization Binding (Contract First) or Calling a Service on a SOAP Binding (Contract First)

Configuring a Task or Event to Expose a Web Service

A process can expose two types of operation: **one-way** and **request-response**.

Exposing a One-Way Operation

A process can expose a **one-way** operation to receive a message from a client application. For example, it may do this to allow a client application to start an instance of the process, to inject a piece of data or to trigger an event.

Procedure

1. Add or select the task or event that you intend to use to expose the web service operation. This must be one of the following types:
 - a receive task
 - a message start event
 - a catch message intermediate event
2. On the **General** tab of the **Properties** view, **Service Type** (for a receive task) or **Implementation** (for a message event) is automatically set to **Web Service**.

Result

TIBCO Business Studio automatically generates a default service.



A task or event that is to be exposed as a one-way web service operation cannot have a Response activity (such as a message end event) associated with it. Also, the **Mode** of all its formal parameters must be **In**.

Exposing a Request-Response Operation

A process can expose a **request-response** operation to receive a message from, then return a response to, a client application.

- a message start event, receive task or catch message intermediate event, which defines the **request** (Input) message. **This task or event must be paired with:**
- a downstream send task, throw intermediate message event or message end event, which defines the **response** (Output) message, and is configured as the reply to the **request**.



The calling application or process pauses its activity when it sends the request and waits for the response from the process before continuing. Because of this, you should only use a request-response operation when there are no arbitrary length tasks between the tasks or events being used to expose the request and response messages. See [Arbitrary Length Tasks and Request-Response Operations](#) .

Procedure

1. Add or select the task or event that you intend to use to expose the **request** part of the web service operation. This must be one of the following types:
 - a receive task
 - a message start event
 - a catch message intermediate event

On the **General** tab of the **Properties** view, **Service Type** (for a receive task) or **Implementation** (for a message event) is automatically set to **Web Service**.
2. Select the task or event that you intend to use to expose the **response** part of the web service operation. This must be one of the following types:
 - a send task

- a throw intermediate message event
 - a message end event
3. On the **General** tab of the **Properties** view:
- a) Click **Reply to Upstream Incoming Request**.
 - b) In the **Request Activity** field, enter the name of the task or event that you selected earlier to expose the web service operation. Content Assist is available to help you complete this field.

Using the Default Generated Web Service Operation

When you configure a task or event to expose either a one-way operation or the request part of a request-response web service operation, TIBCO Business Studio automatically creates a default web service operation for the task or event.

See [Configuring a Task or Event to Expose a Web Service](#).

TIBCO Business Studio automatically creates a default web service operation for the task or event by performing the following actions:

- It creates a WSDL file in the project's **Generated Services** folder. The WSDL filename is *PackageName.wsdl*. The WSDL file contains:
 - a portType with the same name as the process.
 - an operation with the same name as the task or event that is exposing the operation.
 - input and output messages and parameters that match the formal parameters and their modes defined in the process.



The **Generated Services** folder and WSDL file are not created until you save the project. A problem marker, stating that "The web service operation has not been generated", is displayed against the task or event until you do this.

- It populates the **Operation** and **Endpoint resolution** sections with the relevant service details from the WSDL. (See [Web Service Implementation Properties](#) for more information about these fields.)

Implementation:

Operation:

Reply Immediately With Process Id

Port Type:

Operation Name: (Default Generated Service)

Port Name:

Service Name:

Transport:

Endpoint Resolution:

WSDL: Use local: Use remote:

Location:

Endpoint Name:

Security Profile:

On a Message Start Event, the **Reply Immediately With Process Id** checkbox can be used to configure a business process to respond to an incoming web-service message request immediately with the id of the new process instance. When this is configured an **Output Process Id** mapping

property tab is shown where you can map a single process-id value back into the output message.

The purpose of doing this is to get an immediate reply without some of the delay while waiting for process-engine threads to become available to process the start request. The process-id can be used to identify the process from the calling application later (for instance, by using it for correlation in a downstream incoming message activity).

The **Port Type** and **Operation Name** are named, respectively, after the name of the process and the task/event name being used to expose the web service operation. The "(Default Generated Service)" label against the **Operation Name** also indicates that the default web service operation is being used.

- It creates a service input parameter for each formal parameter (of type In or In/Out) associated with the task/event (implicitly or explicitly via the **Interface** tab), and maps each formal parameter to its corresponding service input parameter.
- It creates a system participant (at the package level) and assigns that system participant to the task or event's **Endpoint Name**. The system participant defines the endpoint provided by the web service. Its:
 - **Name** is *ProjectPackageName_ProcessName*. This is shown against the task or event's **Endpoint Name**.
 - **Label** is *ProcessName*. This is shown against the participant in Project Explorer.
 - Transport (HTTP or JMS) will be as defined in the BPM Bindings Preference dialog. See Service Bindings and WSDLs on page 146.
- If the task or event is exposing the request part of a request-response web service operation, TIBCO Business Studio also updates the corresponding task that provides the response part. It creates a service output parameter for each formal parameter (of type Out or In/Out) associated with the task/event (implicitly or explicitly via the **Interface** tab), and maps each formal parameter to its corresponding service output parameter.

If the default web service operation meets your requirements you can use it without further modification. If not, you can modify either:

- the system participant used (see [Using a System Participant to Define the Endpoint Provided by the Web Service](#)) and/or
- the input and/or output data used (see [Defining Input and Output Data](#)).

Alternatively, you can use a web service operation from a different WSDL instead - see [Selecting an Alternative Web Service Operation to Expose](#).

Updating the Default Web Service Operation

TIBCO Business Studio automatically updates the default WSDL file (and associated settings) whenever you make a change that affects it (provided you have the **Project > Build Automatically** flag set).

For example, if you change the formal parameters and/or their modes on the **Interface** tab, TIBCO Business Studio automatically updates the WSDL file and the associated data mappings.

Exposing Multiple Default Web Service Operations

Generated default services are maintained in a single WSDL file:

- If you expose further service operations from the same process, new operations are added to the portType representing that process.
- If you add additional processes to the package, and those processes expose service operations, new portTypes and operations are added to the WSDL file.

Additional operations also use the same system participant.

Selecting an Alternative Web Service Operation to Expose

If you do not want to use the default web service operation , you can select a different one to expose. See [Using the Default Generated Web Service Operation](#).

Procedure

1. Select the task or event that you will use to expose the web service. On the **General** tab of the Properties view, the following buttons are shown against the **Operation** field.

The screenshot shows the 'General' tab of the Properties view. The 'Implementation' dropdown is set to 'Web Service'. The 'Operation' field has four buttons: 'Select', 'Clear', 'Import WSDL', and 'Set Default'. Below this, there is a checkbox for 'Reply Immediately With Process Id' which is unchecked. The 'Port Type' is 'AdjustClaim', 'Operation Name' is 'CatchMessageEvent' (with '(Default Generated Service)' in parentheses), 'Port Name' is empty, 'Service Name' is empty, and 'Transport' is 'Service Virtualisation'. The 'Endpoint Resolution' section has 'WSDL' set to 'Use remote:', 'Location' is 'This is taken from the Alias URL at Run-time', 'Endpoint Name' is 'AdjustClaim' (with '...' and 'Clear' buttons), and 'Security Profile' is empty.

2. Click one of these buttons, depending on what you want to do. The following table describes each available option.

Button	Usage
Select	Choose an operation from a WSDL that already exists in your workspace. See Selecting an Operation From a WSDL That Exists in the Workspace .
Import WSDL	Import a WSDL from an external source (a file, URL or service registry), then choose an operation from the imported WSDL. See Importing a WSDL and Selecting an Operation .
Set Default	Reset the task or event to use the default web service operation. See Using the Default Generated Web Service Operation . <i>This option is not available if the default WSDL is already selected.</i>
Clear	Clear the current selections in the Operation and Endpoint resolution sections.



When you configure a task or event to expose the response part of a request-response web service operation, it will automatically use the web service operation configured for the request activity. The **Select**, **Clear**, **Import WSDL** and **Set Default** buttons are not available, and you cannot select a different web service operation.

Selecting an Operation From a WSDL That Exists in the Workspace

You can select an operation from a WSDL that already exists in your workspace:

Procedure

1. Click **Select**. The Operation Picker dialog is displayed:
 - The dialog lists every web service operation that is available in a WSDL file (in either a Service Descriptors or Generated Services folder) in any project in your workspace.
 - The status line in the dialog shows the project location and filename of the WSDL that contains the currently selected operation.
2. Select the operation that you want to expose and click **OK**.

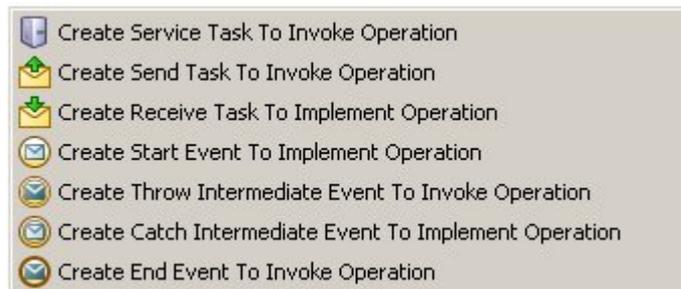
Result

TIBCO Business Studio now automatically performs the following tasks:

- It creates a system participant to identify the web service endpoint to be provided. See [Using a System Participant to Define the Endpoint Provided by the Web Service](#).
- It populates the **Operation** and **Endpoint resolution** sections of the selected activity with the relevant service details from the WSDL. (See [Web Service Implementation Properties](#) for more information about these fields.)

As a shortcut, when you have the WSDL in your workspace you can create a task or event to expose an operation using the following method:

- Expand the WSDL in Project Explorer and select the operation you want to expose.
- Drag the operation to the point in your process flow where you want to expose the web service operation. A context-sensitive menu is displayed, listing the tasks and events you can use to expose the selected operation. For example:



- Select the appropriate option from the menu. The corresponding task or event is created and configured. The **Operation** and **Endpoint resolution** sections are populated with the relevant service details from the WSDL



Importing a WSDL and Selecting an Operation

You can import a WSDL to your workspace and then select an operation from that WSDL.

Procedure

1. Click **Import WSDL**. The **WSDL Import Wizard** is displayed.
 2. Select one of the following import methods:
 - **Import from a File** - to browse the file system for the WSDL document.
 - **Import from a URL** - to specify a URL that resolves to the location of the WSDL document.
 - **Import from a UDDI Registry** - to select a WSDL document from a UDDI registry.
-  | The **Descriptor for XML over JMS** option is not supported by BPM. Do not use it.
3. Click **Next**. If you chose:
 - **Import from a File**, browse to specify the **Location** of the WSDL document.
 - **Import from a URL**, enter the URL for the WSDL document.
 - **Import from a UDDI Registry**, select the WSDL document from the list of registries and registry searches. (See [Using Service Registries](#).)
 4. Click **Next**. The **Destination Selection** page is displayed.
 5. Browse to select the **Project > Location** (the project folder where you want to store the WSDL document), and if necessary change the name of the WSDL document. Select the **Overwrite existing resources** checkbox if you want to replace any existing WSDL document with the same name.
 6. Click **Next**. The **Operation Picker** page is displayed. This shows the WSDL files available in the selected destination project.
 7. Select the operation you want to expose and click **OK**.

Result

TIBCO Business Studio now automatically performs the following tasks:

- It creates a system participant to identify the web service endpoint to be provided. See [Using a System Participant to Define the Endpoint Provided by the Web Service](#).
- It populates the **Operation** and **Endpoint resolution** sections of the selected activity with the relevant service details from the WSDL. (See [Web Service Implementation Properties](#) for more information about these fields.)

Using a System Participant to Define the Endpoint Provided by the Web Service

A system participant is a logical identifier for a connection to an external system - in this case, a web service endpoint. An endpoint defines the URL that will be used to contact the web service.

A task or event that exposes a web service operation must use a system participant that defines the endpoint of the web service that is to be provided. This information is used at runtime to map the call from the client application to the web service operation provided by the process.

When you select or import a concrete WSDL operation binding () or an abstract WSDL operation () or if you use the default generated web service operation, a system participant is automatically created and assigned to the exposing task or event's **Endpoint Name**. (The system participant's name is taken from the portType of the chosen operation.)



When you configure a task or event to expose the response part of a request-response web service operation, it will automatically use the system participant configured for the request activity. You cannot select a different system participant.

If you want to view the properties of the system participant:

Procedure

1. In Project Explorer, select the system participant.
2. On the **General** tab of the Properties view, expand **Shared Resource** and select **Web Service Provider**.
3. A column is displayed showing the binding types that will be used to expose the service:
 - **Virtualization**. There are no further properties for this binding type.
 - **SOAP over HTTP**. Select this binding type to view the binding details - see [SOAP over HTTP Binding Details \(Provider\)](#) for more information
 - **SOAP over JMS**. Select this binding type to view the binding details - see [SOAP over JMS Binding Details \(Provider\)](#) for more information



The binding types used to expose the service are fixed.

Setting a Common Context Root for Web Service Endpoint URIs

You can define a common context root that will be used as a prefix to the URI generated for any system participant. The URI is displayed in the **Endpoint URI Path** field for a SOAP over HTTP binding - see [SOAP over HTTP Binding Details \(Provider\)](#).

You can define a common context root for the entire workspace and/or on a per-project basis. By default, no common context root is defined.



If a common context root is not used, each application will, when deployed, run as a separate web application, increasing the application's usage of system resource and memory. On small systems, with few applications deployed, this is unlikely to be an issue. However, on larger systems, if significant numbers of applications are deployed, this could potentially impact system performance.

Whether or not to use a common context root, and what that root should be, should therefore be an architectural decision based on how application services are to be grouped and presented within the overall SOA environment.

You can define a context root on a workspace and/or on a per-project basis. If you set:

- a workspace-level context root, it will be automatically applied to the endpoint URI whenever a system participant is generated in the workspace.
- a project-level context root, it will be automatically applied to the endpoint URI whenever a system participant is generated in the project. (A project-level context root overrides any workspace-level root that exists.)

To set a workspace-level context root:

- Click **Window > Preferences** . The Preferences dialog is displayed.
- Click **User Profile**.
- In the **Endpoint URI** field, enter the string that you want to use as the context root - for example, **EasyAs** or **/EasyAs/BPM**.

To set a project-level context root:

- Right-click the project in Project Explorer, then choose **Properties**. The Properties dialog is displayed.
- Click **User Profile**.
- Click **Enable project specific settings**.
- In the **Endpoint URI** field, enter the string that you want to use as the context root - for example, **EasyAs** or **/EasyAs/BPM**.

Once a context root has been defined for a project (either directly or at Workspace level), any existing system participant in that project whose URI does not begin with that context root displays the following warning on the Problems tab:



Process Manager 1.x : System participant shared resource endpoint uri does not start with the same prefix as configured in the preference page

You can either ignore this warning or manually edit the system participant's **Endpoint Uri Path** field to include the context root.

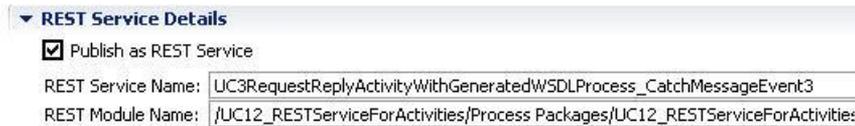
Exposing the Web Service Operation as a REST Service

You can publish a process as a REST (**R**epresentational **S**tate **T**ransfer) service. This is achieved by creating a catch-message (start or intermediate) event or receive-task and then configuring it to be published as a REST service. A client application can then use the BPM REST API to invoke the published REST service.

See "Developing a Client Application Using the REST API" in the *TIBCO ActiveMatrix BPM - BPM Developer's Guide* for more information about how to use the BPM REST service.

Procedure

1. Open your business process in TIBCO Business Studio. You can set REST Service Details on one of the following incoming web service request activities:
 - Start None Event
 - Start Message Event
 - Intermediate Catch Message Event
 - Receive Task
2. When the implementation type for the activity is set to web service, you can select the checkbox "Publish as REST Service" from the General tab of the Properties view. This exposes the REST interface for the business process, and the service and module name fields are populated with the values required for invoking the REST service at runtime.



You also see the following information on any input and output parameters that are defined for that activity:

Input Parameters:

Parameter Name	Parameter Type
x p1	string_50 (http://www.tibco.com/bs3.0/_4PE2EFjcEeKzdM-tKc7NZA)
x p3	User (http://example.com/uc12_restserviceforactivities)

Output Parameters:

Parameter Name	Parameter Type
x p2	string_50 (http://www.tibco.com/bs3.0/_4PE2EFjcEeKzdM-tKc7NZA)
x p3	User (http://example.com/uc12_restserviceforactivities)

The following table describes the input parameters:

Parameter Name	Name of parameter to use: <ul style="list-style-type: none"> For a user defined WSDL this is the original message partname prefixed with "In" (to avoid confusion between same named input/output parts that are of different types). For an auto-generated WSDL, the name matches the business process parameter name.
Parameter Type:	The XSD type of the element and its namespace in the format: <i>TypeName (namespace)</i>

The following table describes the output parameters. Note that the output parameters are only shown for request-reply services when there are output parameters associated with the activity or the activity has 'Reply Immediately With Process Id' option selected.

Parameter Name	Name of parameter to use: <ul style="list-style-type: none"> For a user defined WSDL this is the original message partname prefixed with "Out" (to avoid confusion between same named input/output parts that are of different types). For an auto-generated WSDL, the name matches the business process parameter name..
Parameter Type:	The XSD type of the element and its namespace in the format: <i>TypeName (namespace)</i>

Defining Input and Output Data

You use the **Interface** tab to define the subset of formal parameters defined in the process that are available to the task or event being used to expose the web service operation.

The selected fields will appear on the **Input to Process** or **Output from Process** tabs, where they can be mapped to corresponding service input/output parameters - see [Defining Input and Output Mappings](#).

If the task or event uses an existing WSDL (either selected or imported), the fields/parameters selected on the **Interface** tab define the fields that are available on the **Input to Process** or **Output from Process** tabs.



If the task or event uses the default generated WSDL, the fields/parameters selected on the **Interface** tab act as a filter to control the parts that are created in the WSDL operation. (Note that the generated WSDL is automatically synchronized with the process if changes are made.)

By default, all fields/parameters defined in the process are available.

To define the available fields:

Click  to add new fields. The Select Data Field or Formal Parameter dialog is displayed, listing the available formal parameters that are defined in the process.



If the task or event is configured as a **Reply to Upstream Incoming Request**, and the request activity uses the default web service operation, you cannot add, delete or edit any parameters on this tab. The input and output parameters are declared in the request activity instead. If you wish to edit them, you must do so there.

Procedure

1. Select the data fields or parameters you need, click **Add**, then click **OK**. The selected parameters are displayed on the **Interface** tab.
2. For each parameter, click in the **Mode** cell and select the appropriate value from the drop-down list:
 - **In** - defines a parameter whose value will be sent to the web service.
 - **Out** - defines a parameter that will be used to stored a value r.eturned from the web service.
 - **In/Out** - defines a parameter whose value will be sent to the web service, and then updated with a value returned from the web service.

Defining Input and Output Mappings

When you are exposing a web service operation, you must map the appropriate input/output parameters provided by the web service operation to the appropriate parameters and/or data fields in the process.

On the Properties view for the relevant task or event, the **Input to Process** or **Output from Process** tab (as appropriate) provides a Mapper tool that allows you to easily perform the required mappings.

You can choose to use automapping to map automatically. To do this, use the automap button above the mappings. See "Automapping" in [Calling a Web Service](#) for more information.



If the task or event uses the default web service operation, service input/output parameters and mappings are automatically created (based on the data defined on the **Interface** tab). You cannot modify them.

See [Using the Default Generated Web Service Operation](#) .

You can, however, still map service input parameters to correlation data fields.

Creating a Mapping

The Mapper automatically populates the left-hand and right-hand sides of the tab with the appropriate data.

On the **Input to Process** tab:

- The left-hand side displays the service input and output parameters defined for the selected operation (in the WSDL document) that is being exposed.

- The right-hand side displays the formal parameters and correlation data fields available to this task or event, as defined on the **Interface** tab

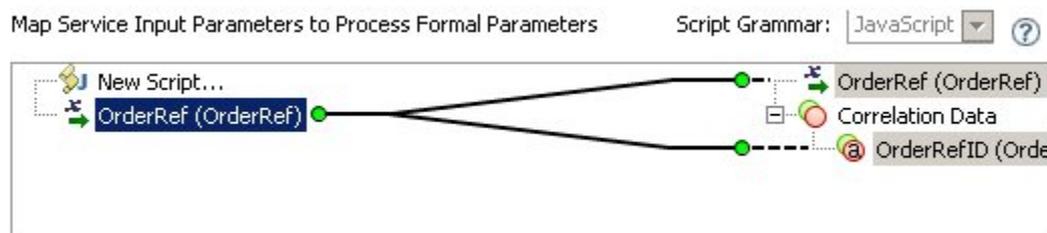
On the **Output from Process** tab:

- The left-hand side displays the formal parameters available to this task or event, as defined on the **Interface** tab.
- The right-hand side displays the service input or output parameters defined for the selected operation (in the WSDL document) that is being exposed.

To perform a mapping, simply drag and drop the parameter(s) or data field(s) that you want to map from one side onto the appropriate service input or output parameter on the other side. A mapping is created between the two entities.

The following example shows the **Input to Process** tab for a message start event that uses the default web service operation. The **OrderRef** parameter, which is the only input parameter defined for the selected operation in the associated WSDL document:

- has been automatically mapped to the **OrderRef** process parameter, which will be used to store the **OrderRef** value sent to the web service by the client application.
- has been manually mapped to the **OrderRefID** correlation data field, which will be used to identify the process instance that the incoming call relates to.



Points to Note About Mappings

TIBCO Business Studio validates the mappings as you create them, and displays error markers to indicate any problems.

For example:

- Every mandatory service parameter defined in the WSDL must be mapped to a process formal parameter.
- All mappings must be from and to equivalent data types - you cannot, for example, map a data field defined as an integer to a service input parameter defined as a date.

See [Message Parameter Mappings](#) for more information about the different types of data mappings that are supported.

If the available process data fields do not provide the necessary data for a particular mapping, you can use the **Script Editor** to create a script that manipulates the process parameters, and map the script to the service parameter.

Using A Script to Define a Mapping

You can use the Script Editor to modify mappings from process data to service input/output parameters or fault messages.



For simple transfer of data between different process data, it may be easier to create scripts with Data Mapper instead of writing them in JavaScript. See [Data Mapping](#) and [Mapping Contents in Data Mapper](#).

Procedure

1. Select the **Script Grammar** that you wish to use, either:

- **JavaScript**, to enter a JavaScript script.
- **XPath**, to enter an XML Path Language expression.

2. Click **New Script** in the appropriate mapping tab.

3. Enter your script in the **Script Editor**.



Scripts are validated as you enter them. Validation markers are displayed if any errors occur.

You can also use content assist in the script Editor by pressing **Ctrl+Space**.

4. When you have finished, click outside of the **Script Editor**. A script called **Script_n** is displayed in the tab. (You can rename it by right-clicking the script name, and selecting **Rename Script**.)

5. Map the script by dragging it from the left side of the mapper to the appropriate entity on the right-hand side.

Throwing WSDL Fault Messages on a Request-Response Operation

If you are using a pair of tasks/events to expose a request-response operation, you can use error end events to throw one or more fault messages as part of the operation.



You cannot throw fault messages when you expose a one-way operation. Fault messages are not supported by this MEP (see [Message Exchange Patterns](#)).

You should define whatever end error events are needed to throw the fault messages defined in the WSDL operation.

If you are using the automatically generated default web service operation, TIBCO Business Studio adds fault messages to the WSDL as required by the end error events you create. It will also automatically configure as much of the end event as possible when you create it, depending on the context.

Using an End Error Event to Throw a Fault Message

You can create an end error event to throw a fault message.

Procedure

1. Add an error end event at an appropriate point in the process.

2. Select the error event.

3. On the **General** tab of the **Properties** view:

- a) Enter a suitable **Label** for the error.
- b) Click **Throw Incoming Message Request Fault**.
- c) In the **Request Activity** field, use context assist to select the task or event that you want this error to be associated with. This should be the message start event, receive task or catch message intermediate event that defines the request part of the appropriate request-response operation.
- d) In the **Enter Fault Name** field, use content assist to select the appropriate fault message that you want to use to throw this error. (Content assist lists each fault message that is defined in the WSDL associated with the selected task or event.)

For example:

Label: StartError

Name: StartError

Result Type: Error

Throw Process / Sub-Process Error
 Throw Incoming Message Request Fault

Request Activity: Start Message Event

Enter Fault Name: StartError

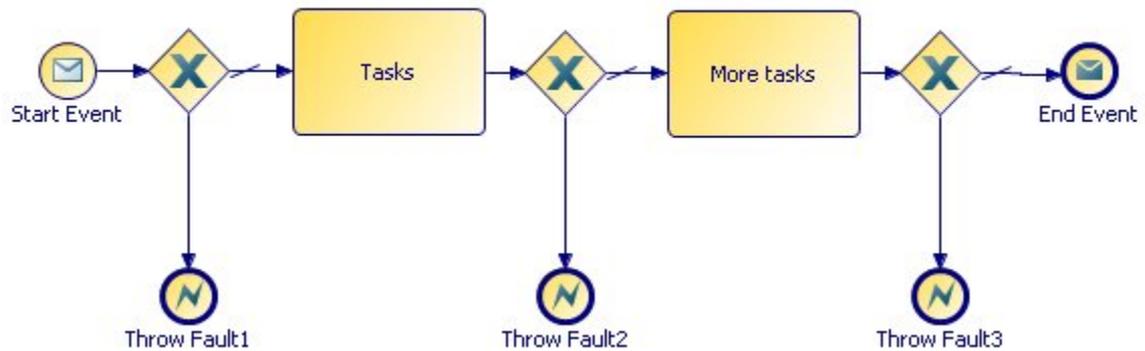
4. On the **Output Fault From Process** tab:

- The left-hand side displays the formal parameters available to this task or event, as defined on the **Interface** tab.
- The right-hand side displays the fault message parameters defined for the selected fault message (in the WSDL document).

Drag and drop the parameter(s) containing the relevant data about the error onto the appropriate fault message parameter(s).

Using an End Error Event Example

The following process illustrates one way to throw multiple fault messages for a single request-response operation.



Start Event and **End Event** are paired to provide a request-response operation, using the following WSDL operation.

Example		
Operation1		
input	INData2	string_50
	INdata1	string_50
output	OUTData1	string_50
	OUTData2	string_50
Fault1	OUTData2	string_50
Fault2	faultparam1	string_50
Fault3	fault1	string_50

The WSDL operation provides three separate fault messages, **Fault1**, **Fault2** and **Fault3**.

The process uses three end error events - **Throw Fault1**, **Throw Fault2** and **Throw Fault3** - to throw these fault messages at appropriate points in the process logic.

Each end error event:

- is associated with the **Start Event** activity in the **Request Activity** field.
- is configured to throw the appropriate error - **Fault1**, **Fault2** or **Fault3** - in the **Enter Fault Name** field.
- maps the appropriate process data for the error (using formal parameters) to the appropriate fault message parameter - **OUTData2**, **faultparam1** or **fault1** - on the **Output Fault From Process** tab.

When the operation is invoked by a client application, the process will either:

- return the output message provided by the **End Event**, if the process completes successfully.
- throw fault message **Fault1**, **Fault2** or **Fault3**, if one of these errors occurs.

It is the client application's responsibility to catch and handle the thrown fault messages. See [Catching WSDL Fault Messages on a Request-Response Operation](#) for more information about how to do this from a process.

Deploying a Process That Exposes a Web Service

When you deploy the project, you must bind the system participant to the appropriate HTTP Connector resource instance in the BPM runtime.

- The system participant defines the web service endpoint.
- The HTTP Connector resource instance is used by BPM to provide external client applications with a runtime connection to the web service.

You can perform this binding using the **Property Configuration** page of the **DAA Deployment Wizard** wizard. For more information see the following references:

- [SOAP over HTTP Binding Details \(Provider\)](#)
- [SOAP over JMS Binding Details \(Provider\)](#)

Alternatively, you can export the project to a Distributed Application Archive (DAA), then use the Administrator interface in the BPM runtime to perform the binding and deploy the application. See the Administrator interface documentation for your BPM runtime environment for more information.

A client application (which can be another process) hosted in the BPM runtime can now call the exposed web service operation on its virtualization binding or, if necessary, on its SOAP binding.

An external client application will need to access the exposed web service on its SOAP binding, using a concrete WSDL.

You can generate a concrete WSDL for the application from the Administrator interface in the BPM runtime.

Arbitrary Length Tasks and Request-Response Operations

When a process exposes a request-response operation, a client application pauses its activity when it sends the request and waits for the response from the process before continuing. If the result is not returned within a predefined time period, the request-response operation may timeout and fail.

An *arbitrary length task* is one that takes an indefinite period of time to complete, and so cannot be guaranteed to complete within this predefined time period:

- User tasks and manual tasks should always be regarded as arbitrary length tasks.
- Other tasks may or may not be regarded as arbitrary length tasks depending on the context in which they are used - for example, is a particular database call expected to complete in milliseconds, seconds or minutes (under normal circumstances)?

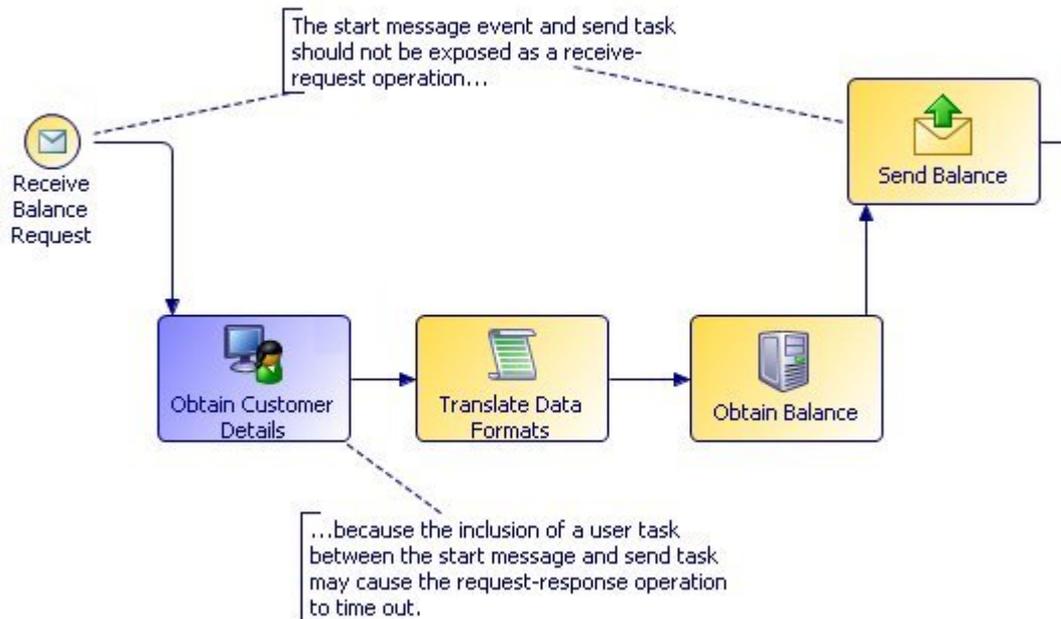


You should only use a request-response operation when there are no arbitrary length tasks between the tasks or events being used to expose the request and response messages.

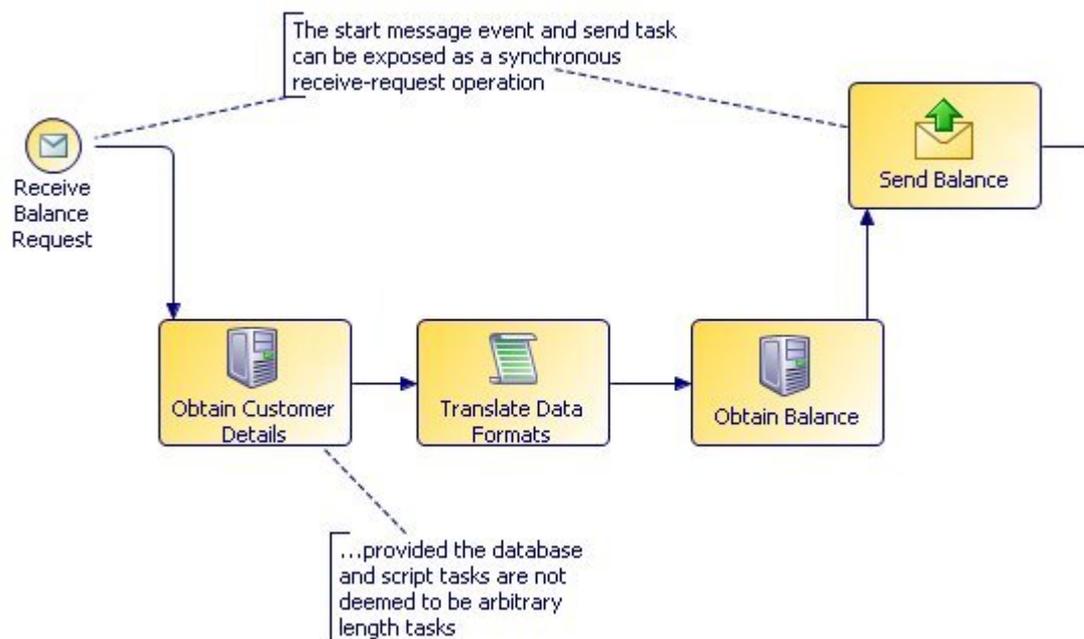
If arbitrary length tasks mean that you cannot use a request-response operation, you should use separate one-way operations instead - see [Handling a Process that Includes Arbitrary Length Tasks](#) for more information.

For example, the following screenshot shows a (partial) process which is intended to provide a balance enquiry web service. On receiving a balance request from an external application (or another process), the process obtains some further customer details, makes a database call to obtain the balance, and then returns the customer's balance to the caller.

In this case the web service should not be implemented using a request-response operation, because the inclusion of a user task may cause the operation to time out.



The next screenshot shows an alternative version of the process, where the additional customer details are obtained using a database task rather than a user task. In this case, whether the web service should be implemented using a request-response operation will be a design decision, based on the characteristics of the script and database calls involved, and whether or not they are regarded as arbitrary length tasks.



Handling a Process that Includes Arbitrary Length Tasks

The inclusion of arbitrary-length tasks can preclude the use of a request-response operation to expose a web service from a process.

See [Arbitrary Length Tasks and Request-Response Operations](#).

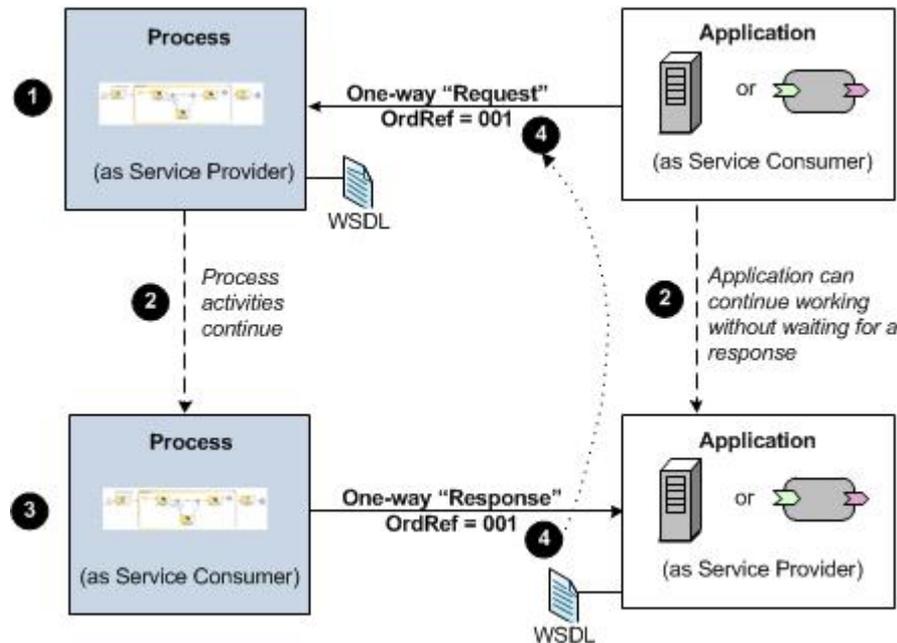
In this situation, the process (service provider) and the client application (service consumer) must instead co-operate to construct a conversation that uses two *one-way operations* - one to allow the process

to receive a message from the client application, one to allow the process to return a response to the client application.

The process and the client application swap their roles of service provider and service consumer for the "request" and "return" legs of the conversation.

Using two one-way operations to handle the problem of arbitrary length tasks illustrates this.

Using two one-way operations to handle the problem of arbitrary length tasks



(1) The process (acting as a service provider) exposes a **one-way** operation to receive a message from a client application (acting as a service consumer). The application can itself be another process.

(2) Both the process and the application continue with their work.

(3) When it is ready to send its response, the process (now acting as a service consumer) invokes a separate **one-way** operation to send a message to the application (now acting as the service provider).

(4) The application is responsible for connecting the separate one-way operations as a single conversation. If the application is another process, correlation data can be used to do this - see *Correlating Separate Request and Response Messages - an Example* on page 32.

Correlating Separate Request and Response Messages - an Example

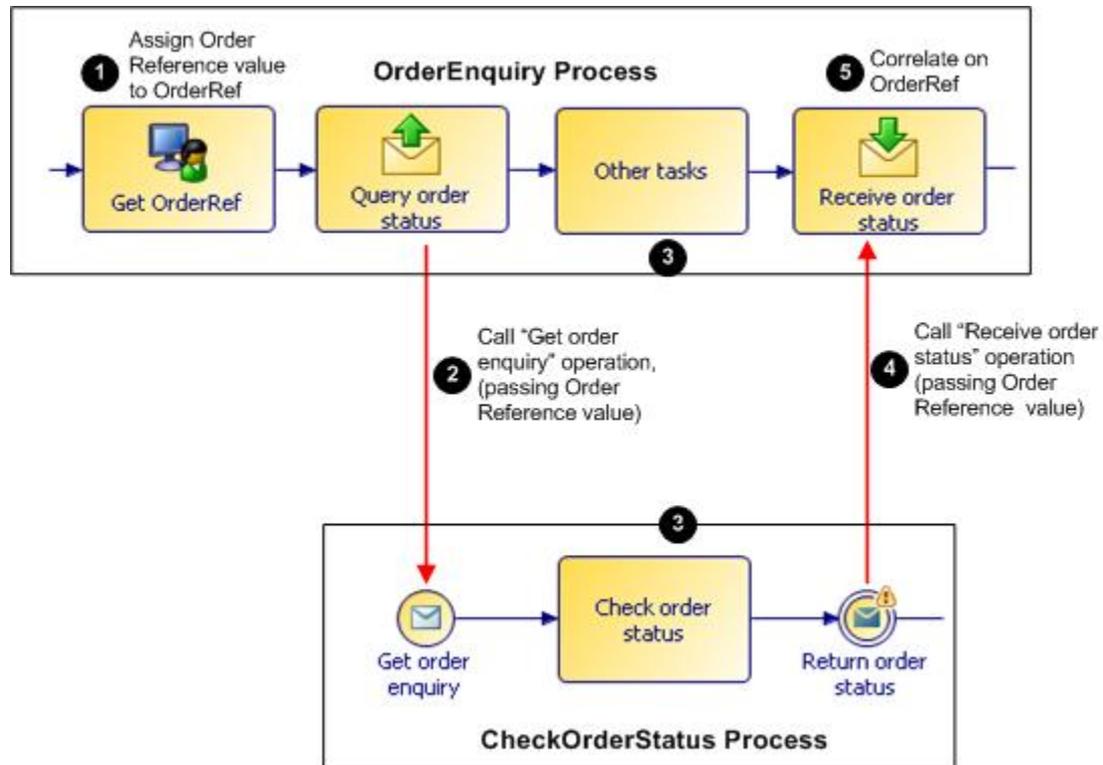
The **OrderEnquiry** process handles customer enquiries about orders, and uses a piece of data - the Order Reference number - which can be used to uniquely identify each process instance. As part of its work, the **OrderEnquiry** process calls a separate process, **CheckOrderStatus**, to check the status of an order.



The example assumes that the "Check order status" part of the **CheckOrderStatus** process contains arbitrary length tasks, which means that the "Get order enquiry" - "Return order status" pairing cannot be exposed as a request-response operation.

Using correlation data to connect separate one-way operations illustrates how the processes co-operate:

Using correlation data to connect separate one-way operations



(1) The **OrderEnquiry** process assigns a value for the queried Order Reference to the **OrderRef** correlation data field as part of the **Get Order Ref** task (which is a user task in this example).

(2) The **OrderEnquiry** process uses the **Query order status** send message task to invoke the **Get order enquiry** operation provided by the **CheckOrderStatus** process. The Order Reference value is required as a service input parameter by this operation.



The Order Reference value does not necessarily have to be defined explicitly as a service input parameter. It could simply be part of another data object that is passed as a service input parameter.

(3) Both processes continue with their work.

(4) When it has completed its "Check order status" work, the **CheckOrderStatus** process uses the **Return order status** event to invoke the **Receive order status** operation provided by the **OrderEnquiry** process. The Order Reference value is again required as a service input parameter by this operation.

(5) On the **Receive order status** task, the Order Reference value is mapped to

- The **OrderRef** correlation data field is defined as a correlation data item on the Interface tab, with **Mode** "Correlate".
- The Order Reference value is mapped to the **OrderRef** correlation data field.

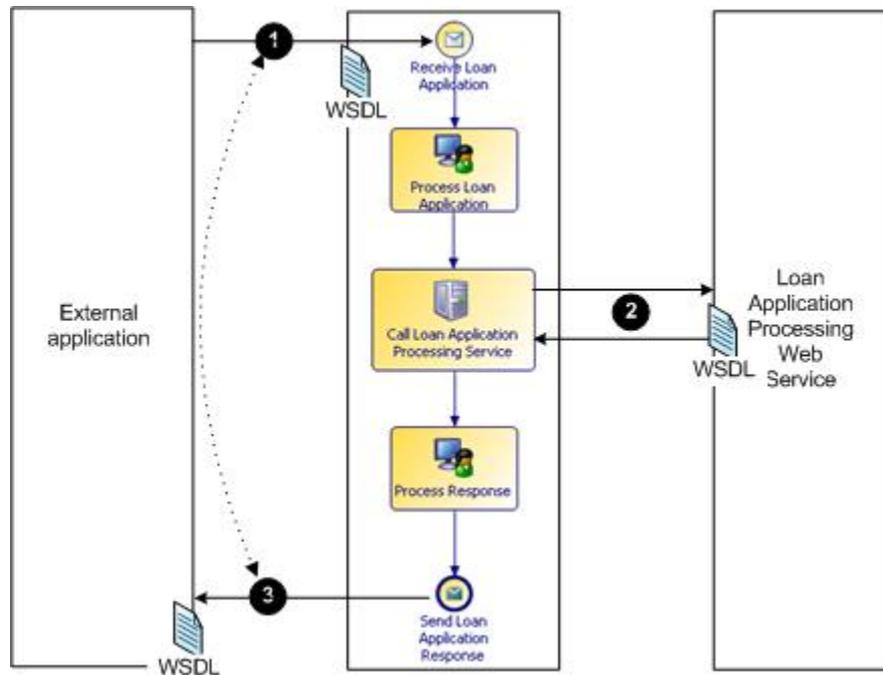
The **OrderEnquiry** process can then determine which process instance the incoming message relates to, based on the value of the **OrderRef** correlation data field.

Using a Process as a Service Provider and as a Service Consumer

A process is not limited to acting just as a service provider or service consumer. You can combine any of these operations within a process to achieve whatever results you need from the process logic. Thus, a process may be at one point a service consumer, at the next a service provider, and at a third may be both supplier and consumer.

[A Process Providing and Consuming Web Services](#) shows an example in which a process provides a loan application service to an external application, and makes use of an external web service to obtain the decision on the application.

A Process Providing and Consuming Web Services



The process does the following:

- receives the request from the external application (1).
- initiates a user task to perform some initial processing of the application.
- calls an external loan application processing web service to obtain a decision on whether the loan is approved or refused (2).
- initiates a user task to perform some processing of the response.
- sends the response back to the external application (3).

Note that:

- Because the user tasks between points 1 and 3 are arbitrary length tasks, the process uses separate one-way operations to provide the service to the external application.
- As part of the conversation formed by the one-way operations, the process uses a request-response operation to obtain the decision on the application from the loan application processing web service.
- The process acts as:
 1. a service provider when it receives the request from the external application,
 2. a service consumer when it obtains the decision from the loan application processing web service.
 3. a service consumer when it returns the response to the external application.

In each case, it is the service provider who provides the WSDL document that defines the messages that are exchanged in each operation.

Authenticating Access to an Exposed Service

At runtime, security policies are automatically enforced on the endpoint of an exposed service to ensure that access is restricted to authenticated users. Every call to the service must be made using the

identity of a user who is registered in the BPM organization model. A call that does not meet this requirement will be rejected.

The following table summarizes the authentication requirements, according to the type of client that is attempting to access the service.

Service is called by...	Authentication Requirements
External client application	<p>Every API call to the service must be authenticated. The following authentication methods are available:</p> <ul style="list-style-type: none"> • Direct authentication - This method requires the calling application to provide valid TIBCO ActiveMatrix BPM login credentials when calling a TIBCO ActiveMatrix BPM service. For more information, see "Direct Authentication" in the <i>TIBCO ActiveMatrix BPM Developer's Guide</i>. • Single sign-on - When using this method, a user who already has a login session with the client application does not need to provide their login credentials again when calling a TIBCO ActiveMatrix BPM service. For more information, see the <i>TIBCO ActiveMatrix BPM Single Sign-On</i> guide.
Another BPM application in the BPM runtime	None. The login credentials used to access the calling process are propagated automatically to the endpoint of the exposed service.
SOA application (for example, Mediation)	An appropriate security policy set and intent must be added to the calling SOA application, to ensure that the correct security context can be propagated to the endpoint of the exposed service. See Calling the Service from a SOA Application .

Calling the Service from a SOA Application

The security requirements of a particular scenario will determine:

- the policy sets and intents that need to be added to the calling SOA application to enforce authentication.
- how and where those policy sets and intents should be configured.
- whether the SOA application invokes the service exposed by the BPM process using a virtualization binding or a SOAP binding.

The following sections provide three high-level examples of how authentication can be enforced:

[Example 1 - Single Sign-on Using a Virtualization Binding](#)

[Example 2 - Single Sign-on Using a SOAP Binding](#)

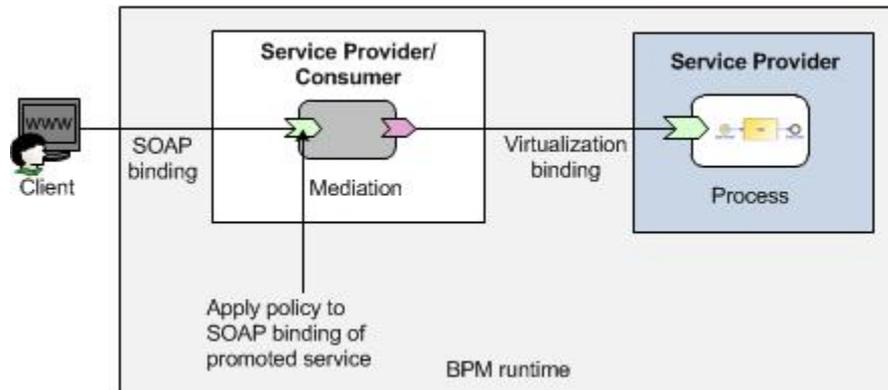
[Example 3 - Impersonation Using a SOAP Binding.](#)



For detailed information about how to configure an application to use intents, policy sets and policies, see the *Composite Development* guide.

Example 1 - Single Sign-on Using a Virtualization Binding

In this example, a client application supplies a user's credentials (username and password) when it calls the Mediation application.



The Mediation application::

- authenticates these credentials using a policy that is used by the BPM runtime.
- propagates these credentials to the service exposed by the BPM process across a virtualization binding.

To ensure that the supplied credentials are valid for both the Mediation application and the BPM process, you must force the Mediation application to authenticate using a specific policy that is used by the BPM runtime.

Procedure

1. Find the `WRMPolicySetsResource.policysets` file in the location where you installed TIBCO Business Studio (for example, `STUDIO_HOME\studio\3.n\samples`).
2. Import the `WRMPolicySetsResource.policysets` file to the project containing your Mediation application.
3. On the Mediation application, select the promoted service that external clients will use to access the Mediation application.
4. On the **Policies** tab of the Properties view:
 - a) Add the **WRMPolicySetsResource_authentication.usernameToken** policy set to the service.
 - b) Add the **Username Token Client Authentication** intent to the service.

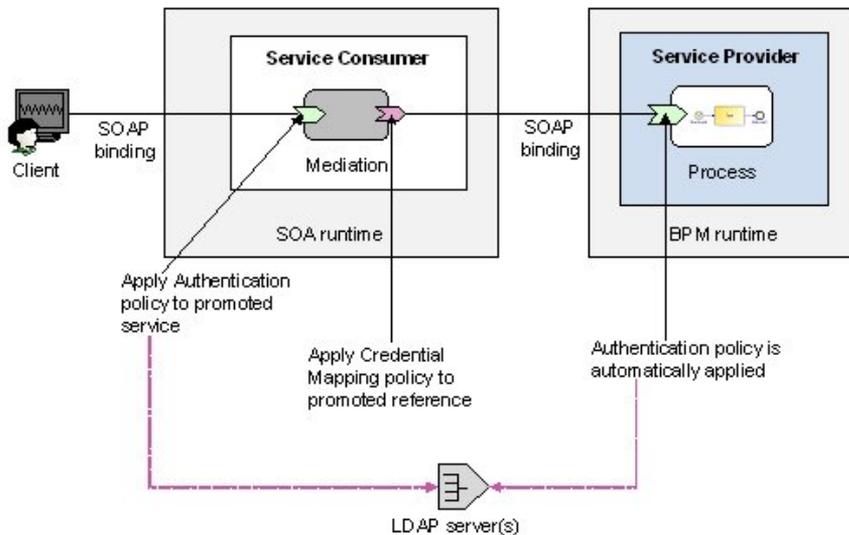
Result

A SOA application can only use a virtualization binding to invoke the service exposed by the BPM process if both applications are running on the same runtime node. (This is because the **WRMPolicySetsResource_authentication.usernameToken** policy set has a dependency on the BPM product application.)

If the SOA application is on a different node it must use a SOAP binding to invoke the service exposed by the BPM process - see the following examples.

Example 2 - Single Sign-on Using a SOAP Binding

In this example, a client application supplies a user's credentials (username and password) when it calls the Mediation application.



The Mediation application authenticates these credentials, which are automatically propagated to the service exposed by the BPM process across a SOAP binding. If the propagated credentials are also valid for the BPM process, access to the BPM service will be granted.

Note the following points about this scenario:

- The Mediation application must validate the user's login credentials against the same LDAP server(s) used by the BPM runtime.
- The credentials used to login to the Mediation application must belong to a valid BPM user. If they do not, authentication against the BPM runtime will fail (even if the credentials were successfully validated against the LDAP server by the Mediation application).

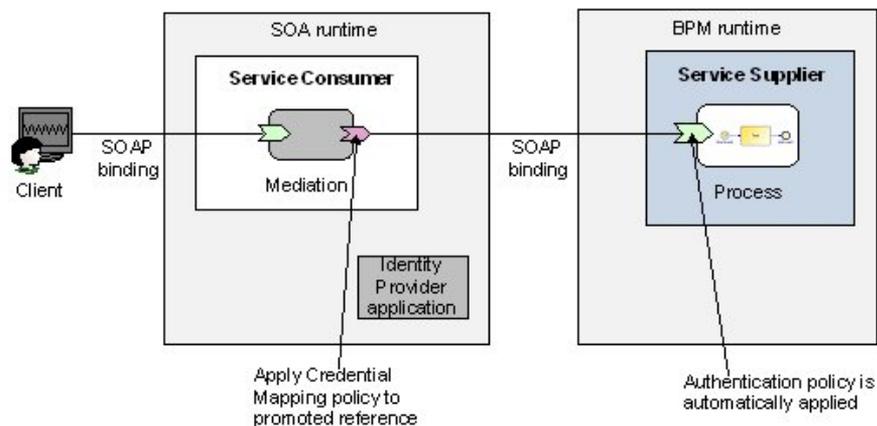
To enforce authentication in this way, configure the Mediation application as follows:

Procedure

1. Add an appropriate client authentication policy to the promoted service. The policy will be applied at runtime, forcing the Mediation application to authenticate the client's identity against the referenced LDAP server.
2. Add an appropriate credential mapping policy to the promoted reference. The policy will be applied at runtime, forcing the reference to propagate the client's previously authenticated identity to the endpoint of the exposed service. This identity will then be authenticated by the BPM runtime.

Example 3 - Impersonation Using a SOAP Binding

In this example, the Mediation application accesses the BPM service using a fixed (impersonated) identity, no matter what credentials are used to access the Mediation application itself. (The Mediation application may even allow anonymous access.)



An identity provider application must be created to provide the valid identity that will be used to access the BPM service.

To enforce authentication using impersonation:

Procedure

1. In the SOA runtime, create and configure an identity provider application that will be used to inject the appropriate credentials into the call to the BPM service.
2. Add an appropriate credential mapping policy to the promoted reference. The policy will be applied at runtime, forcing the reference to propagate the identity defined in the referenced keystore to the endpoint of the exposed service.

Calling a REST Service

Before making a call to the REST service from a process, ensure that you install any relevant runtime hotfixes. See the *TIBCO ActiveMatrix® BPM Release Notes*.



The REST Binding Type does not need to be explicitly installed; it is embedded in TIBCO ActiveMatrix Platform.

Defining the Interface to an External REST Service

To call a REST Service you need to create a REST Service Descriptor (RSD) file (located in a separate REST Service project). The REST method from the RSD file can then be referenced from REST Service invocation activities.

The purpose of the REST Service Descriptor (RSD) is to define the end-point URL relative resource path (optionally including query and header parameters), the available methods for the resource and the request-response data details for those methods.



You should not use URL encoding (for instance %20 for spaces) in the Context Path of the Resource path definition in the REST Service Descriptor. Also, you should not encode values in Path Parameters (that is, the value of data fields mapped to those parameters). This is because the path and parameter values are automatically encoded at runtime. If there is a use case where the path may require URL delimiter characters (such as /, @, ? etc) that need to be encoded then simply use a path parameter in the required location and then provide the required unencoded values in the data mapped to the parameter.

Procedure

1. Create a REST Services project (**New > Other > REST Services Project**).



REST Service Descriptor (RSD) projects do not need to be deployed.

2. A REST Service Descriptor is created automatically.

To create an additional REST Service Descriptor, select **New > New REST Service Descriptor**. The service descriptor can be used by multiple REST service activities.

3. Add the **Context Path** for the REST service, which will be used in the URL for the service at runtime. **Context Path** is a common path prefix that will be used for all resources' URLs in this service. For example, `/rest/api/2/`. The context path is applied to the beginning of all resource paths.

REST Service Descriptor

Resources

Specify resources and methods for this REST service.

JiraApi-Service

Properties Problems Fragments

JiraApi-Service

General

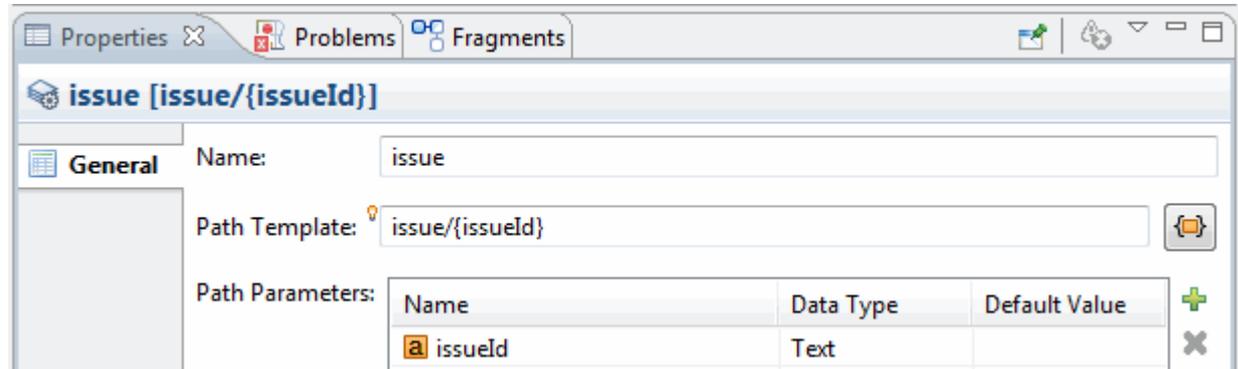
Name: JiraApi-Service

Context Path: /rest/api/2/

Media Type: Standard JSON

- Configure resources (including a Path Template and Path Parameters which will be used in the URL for the method). Path Template may contain path parameter references. Parameter reference is constructed using the name of the parameter enclosed between { and } characters.

Path parameters are variable parts of the path URL that are replaced with process data at runtime (by mapping process data to the parameters in the REST service invocation task).



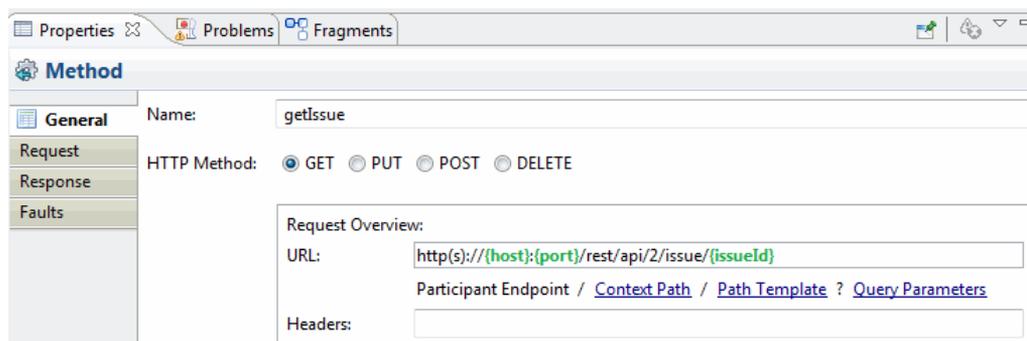
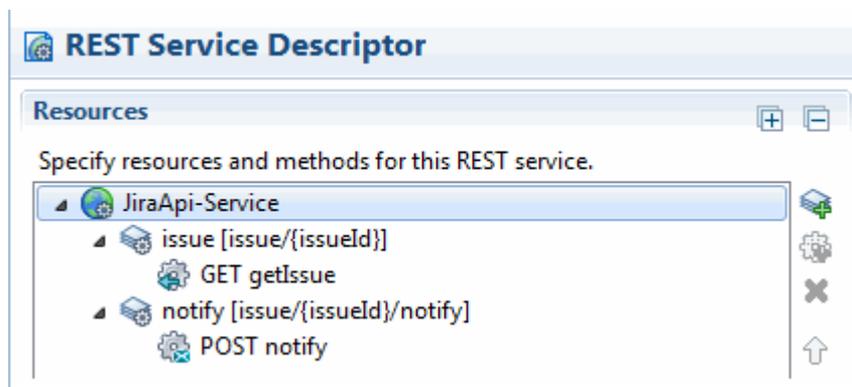
- The URL for the REST invocation method is built up from the following: <baseUrl> + context path + template path).

For example: `http://localhost:8080/rest/api/2/issue/{issueId}` (where <baseUrl> is `http://localhost:8080`, context path is `/rest/api/2`, template path is `/issue/{issueId}`).

The <baseUrl> is provided by the HTTP Client shared resource associated with the REST service task participant during project deployment.

- Add HTTP method/s for each resource, and select the HTTP Method type from GET, PUT, POST or DELETE.

The GET method is already added by default for each resource.



The **Request Overview** provides an idea of what the final URL for the REST invocation method will be.

7. Create request and response information using the method in the Request and Response tabs. On each method, you can define Query Parameters/Header Parameters and Payload (which must be a JSON schema type). See [Creating JSON Schemas](#).

The PUT and POST methods have the Request Payload specification.

- **Query Parameters:** for example, `http://localhost:8080/?param1={param1}` where `{param1}` is a query parameter added using the Request tab.

Name	Data Type	Mandatory	Default Value
param1	Text	<input type="checkbox"/>	

- **Payload Type:** Select from existing JSON objects.



If you cannot create the exact JSON object using the REST service task mapper (or the payload is not in JSON format), you can configure the REST method definition request payload type to be **Unprocessed Text**. In the REST Service task process create a text field whose value is the required JSON string and map that to the request payload.

- **Content Type:** needs to be set to that defined by the service.
- **Header Parameters:** some services require header parameters to be passed, also a service can return header parameters. These can be mapped from and to process data in an invocation task.

Method configuration - Payload tab:

- Payload Type: [issue](#) [Clear]
- Content Type (Accept): application/json
- Array Payload:

Use the **Faults** tab on the method in the REST Service project to define faults for specific error status codes.

If you declare an error you can also declare a JSON type for the information coming back and map it to local data.

Name	Status Code	Payload Type
Invalid Input	400	- not set -
Unable to send	403	- not set -



HTTP response status codes 300 and above are automatically treated as errors at runtime. You can catch any such error with a 'Catch All' task boundary error event or you can define faults for specific status codes that can be caught individually.

Creating JSON Schemas

You can create JSON schemas using this procedure.

Alternatively, you can create a JSON schema by deriving it from a JSON string sample (often made available by the documentation for the REST service). See [Creating JSON schema from a JSON sample](#).

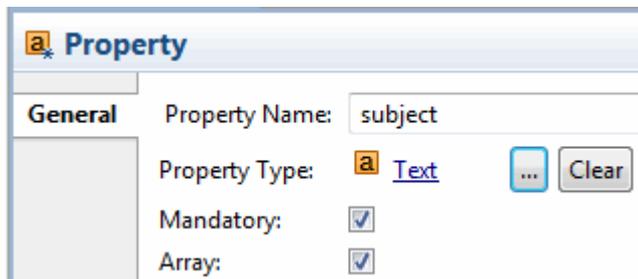
A JSON schema declares the types and properties of the complex types used for request / response payload data when invoking REST service methods.

Each JSON schema contains a **single** root complex type (as displayed on the left hand side tree of the editor). Child properties can be added to this type (on left hand side or right hand side). New child complex types can be added to the right hand side tree.

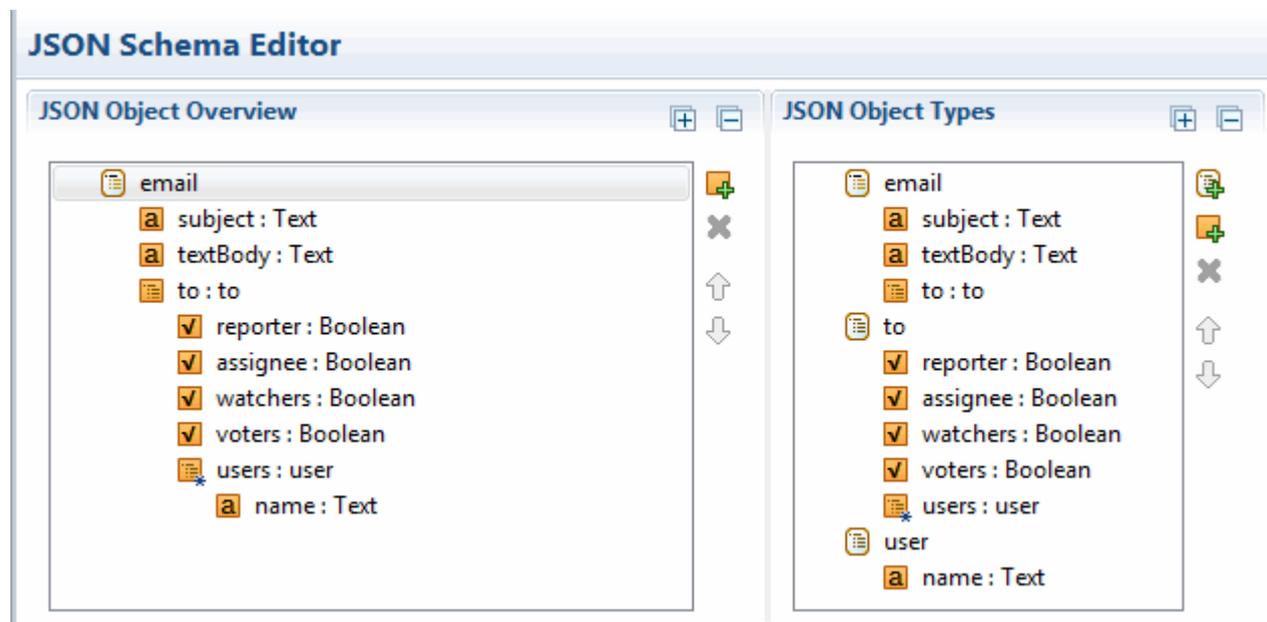
Procedure

1. Select **New > New JSON Schema** from a REST services project.
2. Add Properties to the root JSON object (in the left or right hand side tree control).
3. Create and use child types for properties of a complex type in right hand side tree.
4. On the General tab on the Properties view for a JSON Object Type, edit the property types (which are 'Text' by default). Alternatively you can drag-drop types onto properties in the main editor. You can also move properties around using drag drop or the controls provided.

For services that require or return arrays, check **Array**.



The result will look similar to the following example.





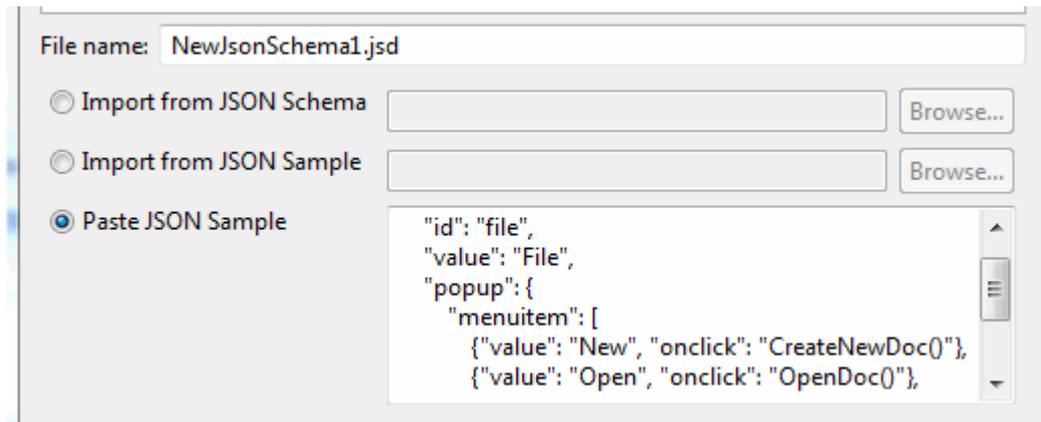
You can also create schemas by importing IETF JSON Schema files (as defined at <http://json-schema.org/>).

Creating JSON Schemas From a JSON Sample

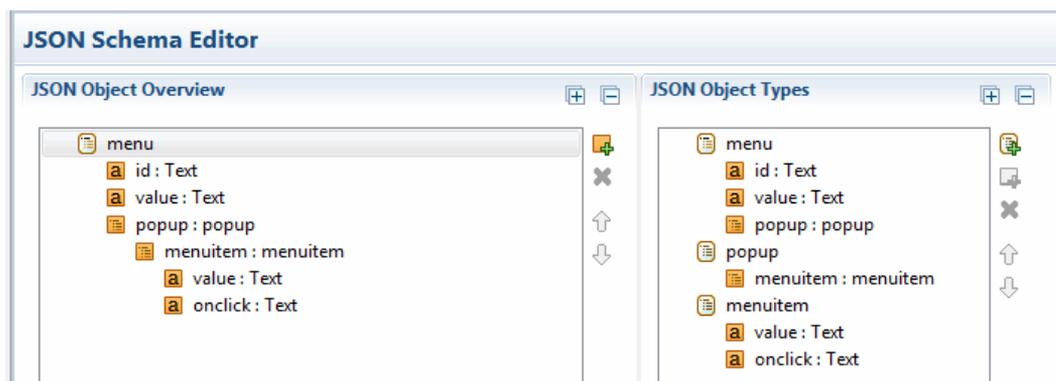
You can use the **Import JSON Schema** wizard to import and save various formats of JSON schema definitions.

Procedure

1. Select **Import > Import JSON Schema** from a REST services project.
2. You can choose to do one of the following:
 - **Import from JSON Schema:** you can import an existing JSON schema.
 - **Import from JSON Sample:** this allows you to import a sample of JSON data, and TIBCO Business Studio creates a schema that describes the data structure.
 - **Paste from JSON Sample:** this allows you to paste a sample of JSON text directly into the wizard and TIBCO Business Studio creates a schema that describes the data structure.



3. You can now view and edit the JSON schema you imported or pasted in the JSON schema editor.



If you import a sample that is an array, then you need to import it as a single instance type and specify that it is an array on a specific method call in the REST Service Descriptor file. See [Creating JSON Schemas](#).

Configuring the Process Project from Which you Want to Call the REST Service

You can configure a task (Service Task, Send Task, Throw Message Event, Message End Event) in a process to call a REST Service.

Procedure

1. Add a Service Task to your process, and configure it with a Service Type of **REST Service**. You can also configure the following to call a REST Service:
 - Send Task
 - Service Task
 - Message End Event
 - Send Task and Throw Message Event: you need to select **Send One Way Request** to activate the Implementation option "REST Service".
2. Select Operation: **Select** to select the REST service method. See [Defining the Interface to an External REST Service](#).

Service Type: REST Service

Operation:

Service: JiraApi-Service

Resource: notify

Method: [notify \(POST\)](#)

3. Define any data fields needed for the call.
4. Map process data to / from the method parameters (see [Defining Input and Output Mappings](#) for more information).

Defining Input and Output Mappings

When you are invoking a REST service operation, you must map the appropriate input/output parameters provided by the REST service operation to the appropriate parameters and/or data fields in the process.

On the Properties view for the relevant task or event, the **Input to Service** and **Output from Service** tabs provide a Mapper tool that allows you to easily perform the required mappings.

Data mappings are defined as follows:

- on the call request (the **Input To Service** tab), from process fields/parameters to path/query/header/payload parameters exposed by the selected Method.
- on the call response (the **Output From Service** tab), from the payload or header parameters exposed by the selected Method to process fields/parameters.

The request / response payload (optional) will be JSON (or 'Unprocessed Text' type). The path and query parameters are used to map into variable parts of the request URL from process data values. The header parameters allow mapping process data to/from request/response header.

Some services will not need a payload, but may still need mappings for headers, path and query parameters.

Creating a Mapping

The Mapper automatically populates the left-hand and right-hand sides of the tab with the appropriate data.

On the **Input to Service** tab:

- The left-hand side displays the parameters and data fields available to this task or event, as defined on the Interface tab.
- The right-hand side displays the path/query/header/payload parameters exposed by the called Method.

On the **Output from Service** tab (which is only displayed for a service task):

- The left-hand side displays the payload or header parameters exposed by the called Method.
- The right-hand side displays the parameters and data fields available to this service task, as defined on the Interface tab.

To perform a mapping, simply drag and drop the parameter or data field that you want to map from one side onto the appropriate Method parameter on the other side. A mapping is created between the two entities.

Map from Process Data to REST Service Request...

type filter text Show Only Mapped Content type filter text

Left pane (Process Data):

- New Script...
- subject
- emailBody (emailBody)
- issueId (issueId)
- users (users) []
- 0..1 name : Text
- doNotSend (doNotSend)
- Process
 - @ description
 - @ id
 - @ masterProcessId
 - @ moduleName
 - @ name
 - @ originator
 - @ originatorName
 - @ parentProcessId
 - ! priority
 - ! startTime

Right pane (REST Service Request):

- Path Parameters
 - issueId
- Query Parameters
- Header Parameters
- Payload
 - email
 - subject : Text
 - textBody : Text
 - to : to
 - assignee : Boolean
 - reporter : Boolean
 - voters : Boolean
 - watchers : Boolean
 - users : user []
 - name : Text

See [Data Mapping](#).

Configuring Security

If you need to apply a security policy on the outgoing REST service call, you do so by assigning a policy to the system participant that identifies the service endpoint.

In order to invoke a REST service, the invocation task/event requires a process package system participant. On deployment, this participant will be associated with an HTTP shared resource instance in the target ActiveMatrix BPM runtime. This shared resource instance (and therefore a single system participant) signifies the base endpoint URL to be used to invoke the REST service.



A REST Service only supports authentication policies of Basic Authentication or Custom Policy.

Procedure

1. In Project Explorer, select the system participant that identifies the service endpoint defined on the REST service call.

2. On the **General** tab of the Properties view, expand **Shared Resource**. The endpoint's configuration details are displayed.
3. In the **Policy Type** field, select the type of security policy required to invoke the service from the drop-down menu:
 - **None**
 - **Basic Authentication** enables you to require credentials, in the form of a username and password, to make a transaction.
 - **Custom Policy**, to apply a custom security policy to the outgoing REST request and, if required, to the incoming REST response.



You must use a **Custom Policy** if the REST response message returned by the service contains a security header. The **Basic Authentication** policy does not handle an incoming REST response that contains a security header.

4. If you selected **Basic Authentication**, a **Governance App. Name** field is displayed. Enter the name of the identity provider application from which the BPM runtime will obtain the authentication information needed to contact the service.
5. If you selected **Custom**, a **Custom Policy Set** field is displayed.
 - a) Click **Browse**. The Select Policy Set dialog is displayed, listing all external policy sets that are available in the current workspace.



Generally, you should copy the custom policy set file into the process package folder of the BPM project where the REST service is configured.



The external policy set file that defines the policy to be used must be available in the same workspace. (It does not have to be in the same project.)

If the required policy set file is not already available, click **Cancel**, import the file to a project in the workspace and try again. See [Custom Policy Set](#).

- b) Select the policy set that the BPM runtime will apply to the outgoing REST request (and, if appropriate, to the incoming REST response).
 - c) Click **OK**.

Custom Policy Set

The custom policy set defines the name of an external policy set that the BPM runtime will apply to the outgoing REST request (and if appropriate, to the incoming REST response).

The external policy set:

- must contain the security information required to construct the outgoing REST request and, if appropriate, to also handle the resultant incoming response.
- must be defined in an XML file (with the extension .policysets) that is available in the same workspace.



TIBCO Business Studio does not validate whether the external policy set is applicable to and correct for the target service. Using an incorrect policy type or a wrongly configured policy will result in an error, either during DAA configuration or at runtime.

For more information on how to set up the username/password identity provider see the tutorial *How to Call a Secured External Web Service From a Process*.

The BPM runtime supports a wide range of policies and policy sets that can be used to address different security requirements and scenarios. For more information about external policy sets and how to create them, see the following topics:

- "Policy Management", in *Composite Development*.

- "Security Resource Templates", in *SOA Administration*. (This guide is not included in the TIBCO Business Studio documentation set. You can access it either from the BPM runtime documentation set, or from the Help in the Administrator interface in the BPM runtime.)

Fault Handling and Propagation

Attach an error event to task boundary and select Catch-all or a Fault code specified for the method in the RSD.

REST and Authentication

You may have issues using authentication with REST at runtime if you do not install the relevant products correctly.



You must follow the installation instructions carefully, in particular you must stop the server during installation when instructed. If you do not, then when you are using REST, an invocation can fail with an authentication error.

TIBCO ActiveMatrix BPM

- You need to install the Server and then the Hotfix as described in the README.
- When you install the Hotfix, the Server must be stopped (make sure it is stopped, even if it is running in the background).

Configure Basic Authentication using the Governance App

- Manually generate the 'Keystore' containing the security credentials (username/password to call BPM REST API's).
- Once the keystore is generated, create the identity provider which uses the generated keystore to supply identity. (Directly provide the identity provider name as a Governance App name in the REST participant.)

For detailed steps, see the topic "Creating a Keystore Containing the Security Credentials to Run the Business Process" in the *Accessing External Data and Services* tutorial.

WSDL Change Considerations for Application Upgrade

This section explains the effects that changes made to a project's WSDL can have on the ability of that project to be upgraded, and suggests design considerations to minimize upgrade problems.

Application Upgrade

One consideration to make when changing a BPM project prior to attempting an application upgrade is the consistency of the application's 'service API' and the data used within those API's.

The service API consists of the web-services that the BPM application exposes to other applications (via processes with incoming message activities) and input/output data used by those services.

In order for the application to be upgradeable, first it must continue to provide any existing web-service operations unchanged (including the data classes referenced from the input/output data of each service).

Secondly, the application must ensure that any data referenced by service input/output data is consistent with data expected to be received by any existing process instances (that have not been migrated to the latest application version).

This is because there is only a single web-service end-point for a service exposed by a process for all versions of that process. Therefore incoming message activities in old-version process instances will be invoked using the data associated with the latest version of the WSDL service interface. The old-version process instances will use old-version BOM data models, and therefore would fail if the new-version data model had changed in an inconsistent way.

When process incoming message activities are configured to generate a WSDL, the WSDL service interface is derived from those activities and the set of data types referenced via formal-parameters associated with the activities. Therefore changes to those activities and data types will affect the application's web-service interface.

When process incoming message activities reference pre-defined WSDLs, the application's web-service interface is based on operations and schema data types defined in the WSDL.

Making changes to the project may result in changes to the generated WSDL, meaning that the application cannot be upgraded. See [Making Changes to the Service Interface](#) for more information about interface changes and about best practice in making and managing them.



You can add a new operation - a Receive Task/event handler - to an existing operation and would be able to upgrade from an existing application.

Reverting to the Original Version of an Upgraded Application

If you encounter a problem during deployment or operation of the upgraded application, you may subsequently want to revert to the original version of the application (by force undeploying the new version from ActiveMatrix Administrator).

However, a WSDL validation error will occur when you try to do this if changes made to the WSDL, though valid for upgrade, are detected as a change to the service interface when attempting to downgrade. (ActiveMatrix Administrator does not distinguish at this level between upgrade and downgrade, so this problem can occur even though the version of the WSDL being downgraded may never have been used, and even though the version being downgraded to will be perfectly compatible with the version of the application that will be active.)

For example, if you add a new operation this is valid when upgrading - but when downgrading you will be removing that operation. This will be detected as a change to the service interface and so is not permitted.

If this happens, you can force ActiveMatrix Administrator to skip WSDL validation and so allow you to revert this application to the original version. See "Troubleshooting > Applications > Unable to revert to

older version of an application" in *TIBCO ActiveMatrix BPM SOA Administration* for more information about how to do this.

Making Changes to the Service Interface

When a process exposes a web service operation, a WSDL defines the service interface to that operation. Once the application containing the process has been deployed to the BPM runtime, it can only be subsequently upgraded if its service interface (defined by the WSDLs used to expose its services) has not changed.

See "Upgrading a Deployed Application" in the *TIBCO ActiveMatrix BPM Deployment Guide* for information on how to upgrade an application.



The WSDL validation tool for upgrade only checks the data involved in an interface previously exposed via WSDL operations. It does not include data types referenced via process data that is not involved in the process API.

Process As A Service (PAAS) interfaces can be defined in one of two ways:

- Using a customer-supplied WSDL and mapping from the WSDL to the process. This is known as the Contract First approach.
- Generating a WSDL from the process definition. This is known as Contract Last approach.

For both approaches, see [Exposing a Service](#).

Best practice for SOA and software development states that Interface should be separated from Implementation. Given the current functionality of TIBCO Business Studio and of the ActiveMatrix BPM runtime, the Contract First approach better engenders the separation of Interface from Implementation. Currently the generated WSDL (the Contract Last approach) can more tightly bind the Interface and Implementation unless the process developer is thinking about how the WSDL is generated from the process.

Changes that Do Not Change the Service Interface

You can make any changes that do not affect the service interface without having any effect on application upgrade.

Such changes include:

- Adding a new operation - a Receive Task/event handler - to an existing operation.
- Adding or removing any tasks except incoming message activities.
- Changing the flow of logic within the process.
- Changing the place of incoming message activities within the process - provided no incoming message activities are added or removed, they can be moved around without affecting the interface.
- Changing the fields or layout of forms.
- Changing scripts.
- Making changes to the schema (BOM) that have no bearing on the WSDL.
- Changes in the way that an XSD type is expressed that do not change its meaning.
- The sequence of root (top-level) element definitions within an XSD.
- Changes in annotations.
- (In a concrete WSDL) Changing the SOAP version used by a binding.

This is a list of possible changes to the elements:

WSDL: Definitions

- Target namespace has to match (but not the prefix).
- PortTypes can appear in any order.
- You can add a new PortType. For generated WSDLs, a port type is synonymous with the source process.

WSDL: PortType

- Operations can appear in any order.
- You can add a new Operation for WSDLs generated from processes, Operation is synonymous with incoming-message-activity.

WSDL: Operation

- Input/Output for WSDLs generated from processes input/output/fault data is synonymous with the formal parameters associated with incoming message activities and throw error events.
 - Can appear in any order within the Operation.
 - Faults, if present, have to match and their corresponding messages have to be identical (can appear in any order).

WSDL: Message

- Target namespace has to match (but not the prefix).
- Parts have to be identical and in the same order.

XSD: Element

- Target namespace has to match (but not the prefix).

XSD: ComplexType

- ComplexType name has to match if directly referenced from a message part.
- Target namespace needs to match, but not the prefix.

XSD: SimpleType

- SimpleType name does not have to match.
- Target namespace and prefix does not have to match (provided that the underlying type and restrictions are identical).

What Changes the Interface Using the Contract Last Approach

If a process uses a generated WSDL, making any of the changes described in this topic will result in changes to the generated WSDL, meaning that the application cannot be upgraded.

Any Contract Last process that contains a task or a service that can expose a web service can be affected by changes in its interface. Projects containing the following tasks or events can be affected:



 <p>Start Event</p>	<h3>Start Message events</h3>
 <p>Catch Message Event</p>	<h3>Intermediate Catch Message events</h3>
 <p>Reply To: Start Event</p>	<h3>Error End events</h3> <p>Even though throw message events (intermediate and end) may be used as a reply to an incoming message activity, when it is used as such no configuration is made on the send task that affects the interface (it is wholly derived from the incoming message activity it is associated with).</p> <p>For throw error events, the associated parameters form the content of the fault message.</p>

If your Contract Last Process As A Service (PAAS) project contains any of these objects, it will include one or more generated WSDLs. It is only if your project has a generated WSDL (not an imported WSDL) that any changes that you make to the project are at risk of automatically altering the service interface. If so, the folder **Generated Services** will exist in Project Explorer, under the relevant project.



In this folder are one or more generated WSDLs - in this illustration, *ProcessPackage.wsdl*. You can open this WSDL in order to see its contents.

FrenchCustomerServices			
CatchMessageEvent			
input	MessageSource	FrenchCustomerServices_MessageSource	→
ReceiveTask			
input	MessageSource	FrenchCustomerServices_MessageSource	→
StartEvent			
input	MessageSource	FrenchCustomerServices_MessageSource	→

You can upgrade the application as long as the WSDL, or WSDLs, do not change.

The following process changes can trigger a change in a generated WSDL:

Changing Tasks or Events

If any of the incoming message activities or message events listed in [What Changes the Interface Using the Contract Last Approach?](#) change, the service interface changes. Changes that have that effect include:

- Removing such a task or event.
- Changing the name of a task or event.

Changing the label without changing the name is permitted because the label is for users' convenience and is not used by the code, it does not affect the interface).

These items and their names are used to automatically generate the operations within the WSDL and therefore will trigger a WSDL change if they are changed.

You can add other kinds of tasks or events to the process without the interface being affected, and will still be able to upgrade the application.

Best Practice

If you expect these items to change then use the Contract First approach to define the service, including the operations and map from the supplied WSDL to the process definition.

Changing the Name of the Process

If you change the name of an application or of a process, its service interface changes. The port type it represents will no longer be available.

Retaining Previous WSDL Versions

It is recommended that all previous deployed generated WSDL versions are source controlled along with the process artifacts, so that you can subsequently compare them with the latest WSDL version being deployed.

Changing Parameters

If you change *any* of the parameters used by a Message Start event, Message Error End event, Intermediate Catch event, or a Receive task, used to expose a web service operation, the change will change the WSDL and the interface that it defines. This applies if you:

- Add a parameter,
- Delete a parameter,
- Rename a parameter,
- Change the name of a type referenced by a parameter,
- Change the type of a parameter



When the BOM data type is used, then the referenced BOM data type and any other BOM data types that it references, directly or indirectly, must remain unchanged. This check is restricted to only this set of BOM types and other BOM types defined in the same BOM packages that are not referenced via incoming message activity parameters can be changed.

- Change the length of a parameter.

Best Practice

Changing an interface is changing a contract and therefore will prevent application upgrade. Due attention should be paid to ensuring that the interface design takes account of the foreseeable future design requirements of the interface. The Contract First approach will naturally drive towards taking this into account.

There are a number of data-related changes which one might normally want to make to an interface which are compatible with the existing deployed clients using this interface. However, this is not allowable for application upgrade because in that circumstance there is only a single end-point for the service but incoming requests may be directed towards existing process instances from older versions of the application that are developed with the old data models and service operations. Such changes would therefore be incompatible with the existing process instances that would be using the new interface.

These include (but are not necessarily limited to):

- Lengthening of a parameter, for example changing a `string(10)` to `string(20)`, `int` to `bigint`, and so on.
- Addition of optional parameters.

Depending upon your use case, you may be able to use Mediation to wrap the different interface implementations that need to be active at the same time.

Adding a new operation to existing port type that uses a Contract First approach: Recommended approach

The following procedure applies when it is necessary to add a new operation to existing port type that uses a Contract First approach.

Take all these actions in the context of the Best Practice guidelines given in [Changing Parameters](#).

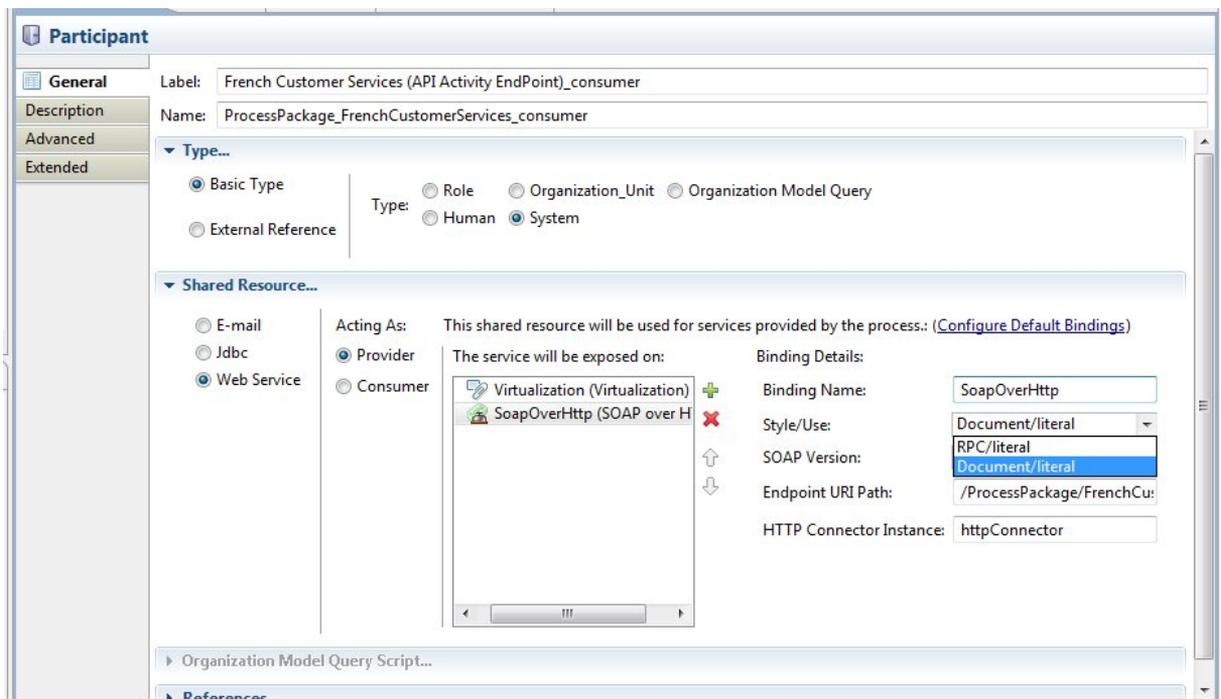
1. Add operation to the WSDL.
2. Add an incoming message activity to the process for a new operation (using the existing participant).

Changes That Apply to Both Contract First and Contract Last Approaches

The following changes can trigger an interface change whether you use the Contract First or the Contract Last approach.

Changing the Binding Style

The service interface changes if you change the binding **Style/Use** for the generated WSDL — for example, from **Document/literal** to **RPC/literal**. You can do this when defining the details of the endpoint system participant for the task or the event.



Best Practice

Changing the binding style is changing a contract and therefore is not something that should be undertaken lightly.

If you want to change the binding style for a PAAS interface while there are still deployed process templates that have executing process instances, then you will need to define a new endpoint.

Development vs. Production

When in development of a complete new application (before first production deployment) it is highly likely that a PAAS interface will change between iterative deployments. In this case it is recommended that previous application versions be completely undeployed and removed from the development system in order to prevent any failed deployments due to interface changes. This will require any in-progress work items and process instances to be cancelled first. In this case there are no application upgrade considerations and therefore any changes could be made to the application interfaces.

Using Scripts

This section describes how to use scripts.



The JavaScript and Data Mapper script grammars are only available when both the Solution Design capability is selected and the BPM destination environment is specified for the current project.

You can specify scripts in TIBCO Business Studio for use with BPM projects in several ways:

- As part of a script task. See [Implementing Script Tasks](#).
- As an action script on a task, particularly a user task. Action scripts can reference information about individual work items as well as using the parameters and fields defined for your process in TIBCO Business Studio. See [Scripts on Other Tasks](#).
- Attached to a conditional flow, for determining the flow of processing. See [Associating a Script with a Conditional Flow](#).
- Attached to a processing loop applied to a task, to determine how often that task should be performed. See [Associating a Script with a Loop](#).
- On an event. See [Timer Event Scripts](#) and also the event activity types in the table below.

Another form of scripting is the use of expressions to define a participant in a task. See [Using a Participant Expression to Define a Participant](#) for details of participant expressions.

Scripts may be entered in TIBCO Business Studio as:

- **Plain text.** When a business analyst creates a process, they may include a plain text description of the desired behavior of any scripts that form part of the process. These plain text entries are not intended to be executable.
- **JavaScript.** Executable scripts are written in JavaScript. If an analyst has entered a plain text description, it is then part of the solution designer's role, with the Solution Design capability switched on, to enter JavaScript in order to implement the script.

TIBCO Business Studio supports a subset of standard JavaScript:

- Those standard JavaScript facilities that are not supported in TIBCO Business Studio scripts are described in [JavaScript Exclusions](#).
- Some additional facilities are available. These are described in [Editing Scripts](#) and 'Script Functions' in the *Business Data Services Guide*.
- **Data Mapper.** For simple transfer of data between different process data, it may be easier to create scripts with Data Mapper instead of writing them in JavaScript. With Data Mapper, you can graphically map data across datafields and parameters to create complex BOM objects from a combination of process data fields and parameters. This script grammar is available for Script Tasks, Task scripts, Web Services, Call Sub-Processes, and Catch error events. For more information, see [Data Mapping](#) and [Mapping Contents in Data Mapper](#).

Different script types can be used as shown in the table below. For each activity type, there are script types that are:

- Supported.
- Available in the user interface but not supported.

Action scripts supported by different activity types

Activity Type	Script Types
User Task	Supported: Open, Close, Submit, Schedule, Reschedule, Initiate, Complete, Timeout, Cancel
Manual Task Service Task Send Task Receive Task Reference Task Call Sub-process Activity Embedded Sub-process Activity	Supported: Initiate, Complete, Timeout, Cancel
Script Task	Supported: Initiate, Complete, Timeout In user interface but not supported: Cancel
Pageflow User Task	Supported: Initiate, Complete In user interface but not supported: Open, Close, Submit, Schedule
Multi-instance Task - Multi-instance Loop	Supported: Loop Expression, Complex Exit Expression, Additional Instances Expression
Multi-instance Task - Standard Loop	Supported: Loop Condition
Start Event	Supported: Complete
End Event	Supported: Initiate
Intermediate Event	Supported: Initiate, Complete, Cancel

Implementing Script Tasks

You can create a script task in a process using the Script Task icon on the TIBCO Business Studio palette.

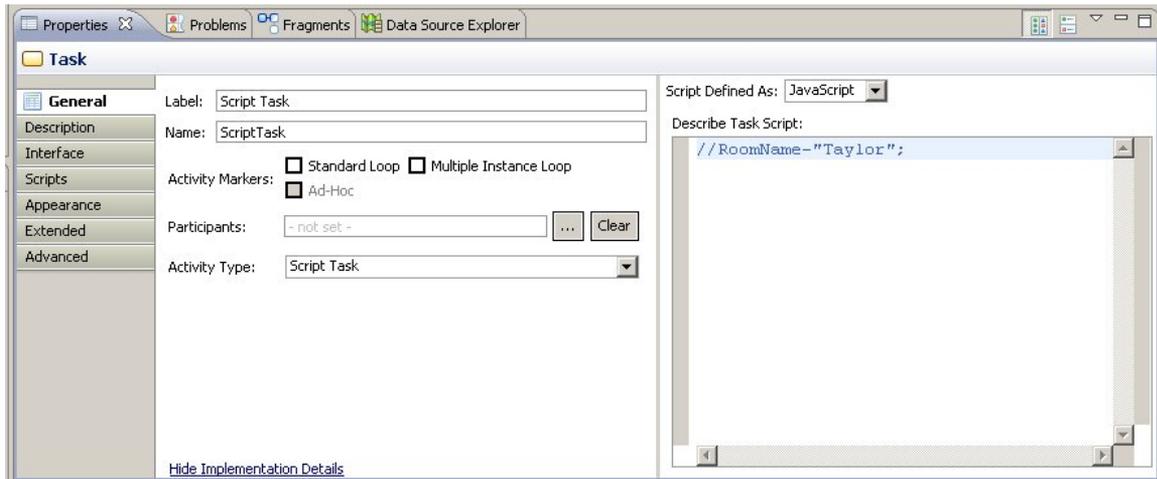
See the *TIBCO Business Studio Modeling User's Guide* for information on how to create tasks.

When a business analyst creates a process, they may include in the script task a plain text description of the desired behavior of the script. It is then part of the solution designer's task, with the Solution Design capability switched on, to enter JavaScript in order to implement the script task.

In the Properties view for a Script Task, you can select from the available script grammars, and enter a script that will be executed at runtime. Which script grammars are displayed depends on the

Destination Environment and the Eclipse capability you have selected. For example, for the BPM Destination Environment with the Solution Design capability selected, you can enter JavaScript:

- Clicking **Hide Implementation Details** switches off the Solution Design capability and displays the view which the business analyst would have.
- On the **General** tab, there may already be a text description of the required script. This description will be preserved as it is (and will usually contain a number of error markers) if you select **JavaScript** from the **Script Defined As** list. If you select **Unspecified** from the **Script Defined As** list, the description is lost. You can however recover the description by pressing **Ctrl + Z**.



Enter the script itself in the **Describe Task Script:** area. This area supports text editing assistance, as described in [Editing Scripts](#).

In addition to the main script for the task, which you enter in the **Describe Task Script:** area of the **General** tab as described above, you can create further scripts in the same way as described for other types of task in [Scripts on Other Tasks](#). You enter these on the **Scripts** tab from the **Process Manager Scripts** selection. For a Script task, you can enter the following types of additional script:

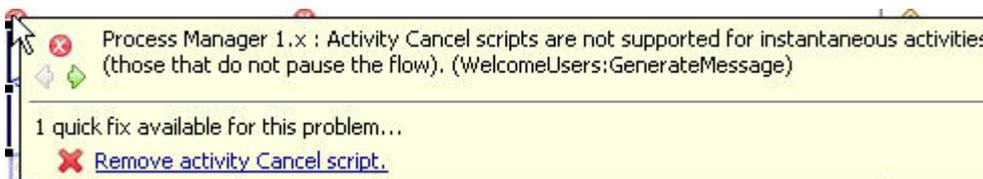
- **Initiate**
- **Complete**
- **Timeout**

Unsupported Script Types

BPM does not support **Cancel** scripts on a Script task.

The Cancel script type is shown as available on the **Scripts** tab of the Properties view for the Script task, even though it is not supported by BPM.

However, if a process has BPM set as a destination environment, any script task that has a Cancel script defined will show a warning message:



To remove this warning, you can right-click the validation error in the Problems view and select **Quick Fix**. This will remove the unsupported script.

Scripts on Other Tasks

As well as Script tasks, tasks of other types may have scripts attached to them. While running a script is not the main purpose of these tasks, you may still wish to have a script run when the task completes, for example, or if it is cancelled. These are called **action scripts**.

Some action script types use **Work Manager Scripting** which references information about individual work items, and about entities in the organization model. Other action script types use **Process Manager Scripting**, which references parameters and fields defined for your process in TIBCO Business Studio, and can also be used to access information about the organization model. See the table in [Supported Script Types](#) for information about what type of scripting is supported for each Action Script Type.

Supported Script Types

You can create a number of different types of script for **User tasks** (which are listed in the order in which they are executed):

Action Script Type	Description	Work Manager executed	Process Manager executed
Initiate	Executed when the activity initiates.		Y
Schedule	Executed when the process schedules a work item.	Y	
Reschedule	Executes when a work item is updated via a non-cancelling signal event on task boundary.	Y	
Open	Executed when the user opens the work item.	Y	
Close	Executed when the user closes the work item (returning it to their Inbox).	Y	
Submit	Executed when the user submits the work item.	Y	
Complete	Executed when the activity completes.		Y
Timeout	Executed when the activity times out.		Y
Cancel	Runs when the activity is cancelled.		Y

In addition, you can create the following types of script for all other task types, including **Pageflow User tasks**, **Service tasks** and **Manual tasks**:

- **Initiate**
- **Complete**
- **Timeout**
- **Cancel**

Sample Scripts

This section contains samples of work manager scripting and process manager scripting.

Example of Work Manager Scripting:

The JavaScript below is defined on the 'Schedule' Script meaning that when this work item is scheduled this script will execute. The script is using the `WorkManagerFactory.getWorkitem().getWorkItemOffers()` method to return the list of resources to which the work item is offered.

`WorkManagerFactory` methods are defined in 'Work Manager Scripting' in the *TIBCO ActiveMatrix BPM Business Data Services* guide.

```
var theOfferSet = WorkManagerFactory.getWorkItem().getWorkItemOffers();
var i = 0;
DFOfferSet.clear();
while (i<theOfferSet.size()){
//loop through resources in offer set
    var resourcesFromOfferSet = theOfferSet.get(i).getName();
    DFOfferSet.add(resourcesFromOfferSet);
//add resource to OfferSet array field
    i++;
}
```

Where 'DFOfferSet' is a data field defined in the process and added as an interface to the user task.

The `WorkManagerFactory.getOrganizationModel` method can be used to navigate around the organization model to identify a particular organizational entity, or a resource mapped to such an entity. For example:

```
var useResource =
WorkManagerFactory.getOrganizationModel().getAssociatedResources(E
asyAs.ClaimsDept.Manager)
```

Resources can be located using RQL queries, using the `getOrganizationModel().getResourcesByQuery` method. See "Resource Query Language" in *TIBCO Business Studio BPM Implementation Guide* for more information.

Examples of Process Manager Scripting:

The JavaScript below is defined on the 'Initiate' Script meaning that when this user activity is initiated, this script will execute. The script is using the `Process.getActivityLoopIndex()` method to return the loop index for a multiple instance user task.

```
var myActivityLoopIndex = Process.getActivityLoopIndex();
if (myActivityLoopIndex != null)
{
    UTAActivityLoopIndex = myActivityLoopIndex;
}
```

Where 'UTAActivityLoopIndex' is a data field defined in the process and added as an interface to the user task.

The following example shows extracts from a Process Manager script being used to get information about the organization model:

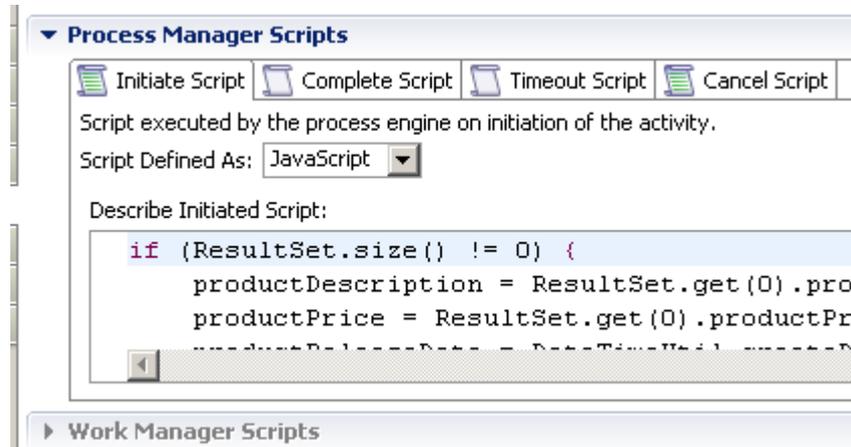
```
var orgModel = Process.getOrgModel();
var unit1 = orgModel.ouByGuid("_xI4ikMpPEd64gM7QE8RwxA");
var position1 = orgModel.positionByName("clerk");
```

Adding an Action Script to a Task

You can add an action script to a task.

Procedure

1. Click the task to which you want to add a script.
2. In the Properties view for the task, click the **Scripts** tab.
3. Select from **Process Manager Scripts** and **Work Manager Scripts** to display the different sets of scripts available. If either of those sets of scripts is not available for the task you have selected, the selection will be grayed out (so in the example below, Work Manager Scripts is grayed out).



4. Click the tab for the type of script that you want to add. See [Supported Script Types](#) for a list of supported types.

If a script is already defined under a tab, then this is shown by the script icon before the tab name. So in the example above, a script is defined under **Initiate Script**. If no script is defined, the tab name will be preceded by an empty script icon.

5. Depending on the destination environment and capability selected, there are several script types in the **Script Defined As** list. Select one of the available script types and enter your script in the area provided. You can press **Ctrl + Space** for Content Assist.
6. Once you have created the desired scripts, save the Package that contains the process.

Result

Content assist is available for action scripts in the same way as for script tasks. See [Editing Scripts](#) for details. However action scripts can also reference information about individual work items.

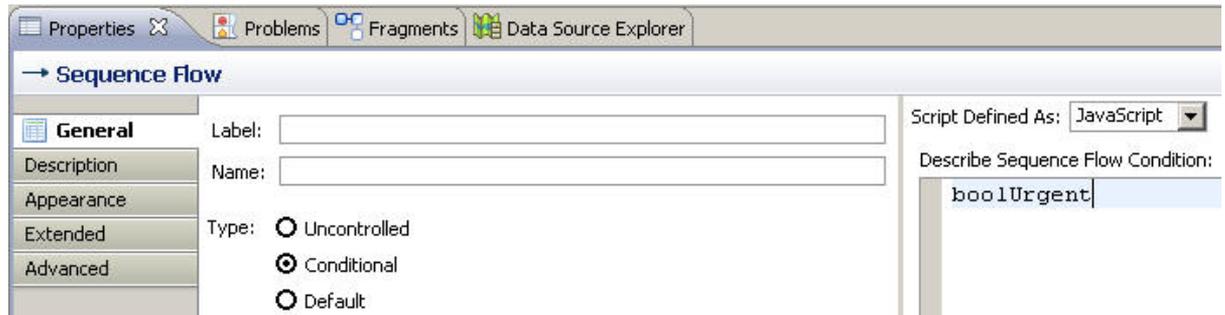
Associating a Script with a Conditional Flow

Scripts are associated with a Conditional Sequence Flow by entering the script in the Properties view for that Sequence Flow object. You can use a script to define the conditions that determine whether a conditional sequence flow is followed. At runtime, this causes the Sequence Flow to be followed only if the condition is met.

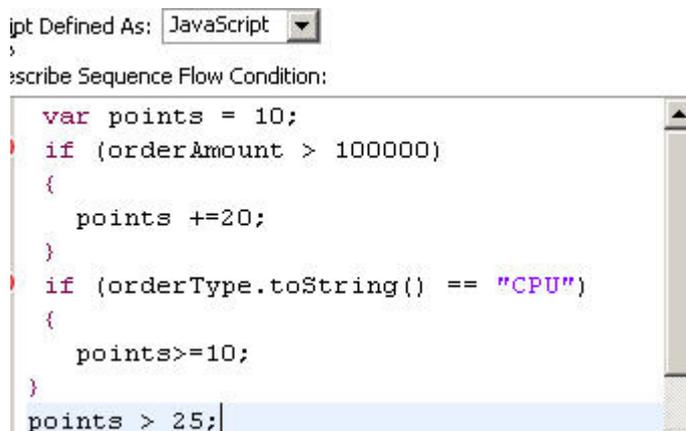
Procedure

1. Create the Conditional Sequence Flow object. On the **General** tab of the Properties view, note that the **Conditional** button is selected.
2. In the **Script Defined As** list, select **JavaScript**.

- Enter the script in the **Describe Sequence Flow Condition** area. The script can be a multi-line script and the result of the last statement of the script should be of boolean type and that determines the result of the script. This is an example of a simple script:



This is an example of a more complex script:



The **Describe Sequence Flow Condition** area supports the usual text editing assistance such as color syntax highlighting, content assist and error markers (see [Implementing Script Tasks](#) for more information).



On the **General** tab, there may already be a text description of the required script. This description will be converted to comments if you select **JavaScript** from the **Script Defined As** list. If you select **Unspecified** from the **Script Defined As** list, the description is lost. You can however recover the description by pressing **Ctrl + Z**.

Associating a Script with a Loop

Scripts can be associated with processing loops in the Properties view for the task to which a loop has been applied.

Procedure

- On the **General** tab of the Properties view for the task, select either **Standard Loop** or **Multi-instance Loop**.



When using a multi-instance loop, you should set an additional instance expression which must evaluate to 0. When the loop has finished it looks to see if this exists. If you do not set this to 0, then your multi-instance loop will not be able to end.

- When either of these is selected the **Loops** tab becomes available. This tab specifies the scripts you can enter for each type of loop. Select **JavaScript** from the **Script Defined As:** field and enter the script itself in the area provided.

Result

See the *TIBCO Business Studio Modeling User's Guide* for a full description of how to create a loop and the significance of the different types of script you can assign to it.

Timer Event Scripts

Scripts can be added to Start events or Catch Intermediate events in the Properties view, if the event is defined as triggered by a timer.

There are several script types available from the **Script Defined As** list:

- **Free Text** - the Business Analyst or person who created the process can use this field to enter text that describes the desired behavior for the script.



On the **General** tab, there may already be text comments describing the required script. These comments will be lost if you change the select **JavaScript** from the **Script Defined As** list. If you need to save these comments, copy them before changing the script type.

- **Constant Period** - this allows you to specify the timeout period after the event is initiated using the following time units.

Script Defined As:

Specify timeout as offset from event initiation:

Years:	<input type="text" value="0"/>	Hours:	<input type="text" value="0"/>
Months:	<input type="text" value="0"/>	Minutes:	<input type="text" value="0"/>
Weeks:	<input type="text" value="0"/>	Seconds:	<input type="text" value="0"/>
Days:	<input type="text" value="0"/>	Micro Seconds:	<input type="text" value="0"/>



- At runtime the timeout period is calculated using the **calcDeadline** API operation described in "BusinessDeadlineService" in the *TIBCO ActiveMatrix BPM Developer's Guide*. Note that if you specify a date without a time element (no hours or smaller units) then the period is assumed to be in working days.
- The **calendarLookAhead** property in the **dac.properties** file specifies how far ahead the algorithm should look when calculating the timeout. If there is not enough working time available to complete the task in the period defined by **calendarLookAhead**, an error is returned. The property defaults to a value of one month, but you should ensure that it is set to a large enough value to give correct results for your calculations.

See "Configuring TIBCO ActiveMatrix Calendar Properties" in *TIBCO ActiveMatrix BPM Administration* for more details of this property.

- **JavaScript** - this script type allows you to enter JavaScript statements in the space provided. The script should return a datetime or datetimesz type for an absolute datetime deadline, or a duration if the in-scope calendar is to be used. The script can be as long as you like, but the result of the script must be one of those types. For example:

Script Defined As:

Describe Timeout Script:

```
DateTimeUtil.createDatetime("2010-06-01T00:01:00");
```

In this example, the DateTime is specified using one of the DateTimeUtil factory methods described in 'Script Functions' in the *Business Data Services Guide*. The event will be fired at the date and time specified.

If only a Date were present, the event would fire immediately when the deadline is created (in the runtime environment) on the Date specified. If only a Time were specified, the event would fire at the specified time on the current date.

Task Scripts on Events

Any event whether timer triggered or not can have a task script assigned to it.

The possible scripts are as follows:

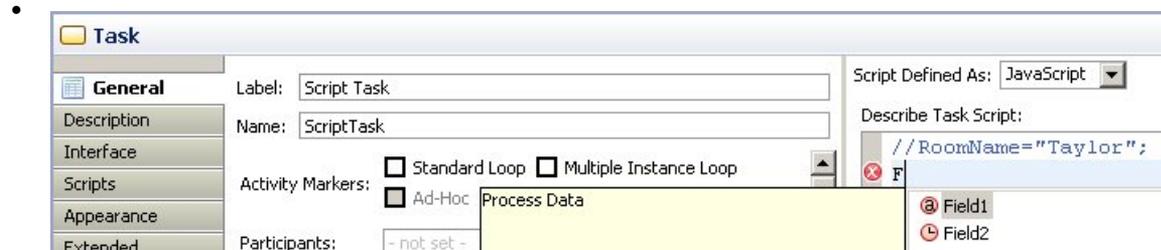
Event Type	Allowed Script Types
Start Event	Complete script
Intermediate Event	Initiate script Complete script Cancel script
End Event	Initiate script

You define task scripts for events on the **Scripts** tab of the Properties view for the event, in the same way that you define the corresponding scripts for tasks. See [Adding an Action Script to a Task](#) for information on how to do this.

Editing Scripts

All the areas in which you can enter scripts support the usual text editing assistance such as color syntax highlighting, content assist and error markers. For example, if you want to specify a data field called **Field**, enter the character "F", then press **Ctrl + Space**. All matching data fields are displayed:

- Content assist is case-sensitive. For example, it would differentiate between the class **Date** and a variable called **date**.



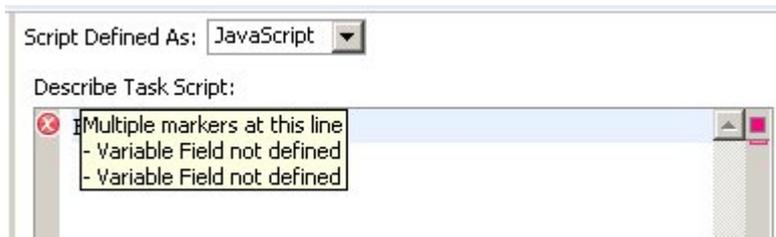
- For arrays and multiple-valued items, content assist displays only the item name; it does not prompt for any indication of the multiple values.

You can then select the desired data field from the list and continue entering JavaScript:



In this case there is an error marker next to the line. This is because validation has reported an error in the Problems view. TIBCO Business Studio provides validation for the JavaScript syntax.

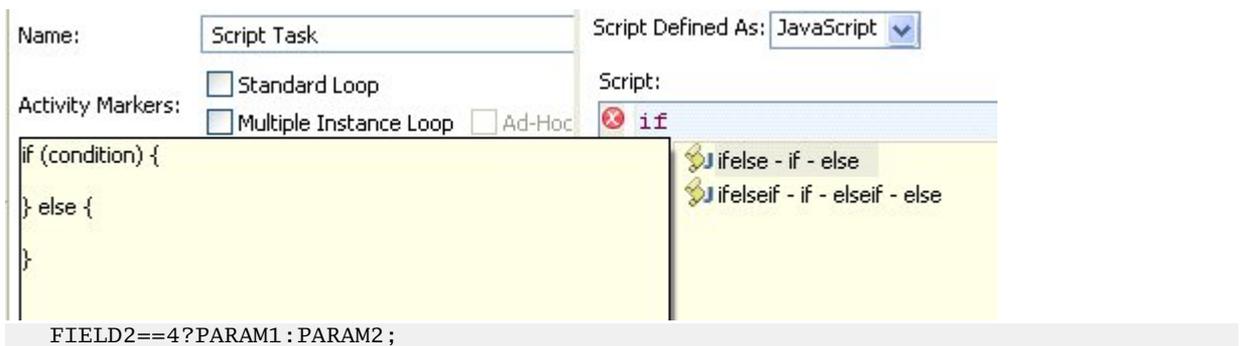
You can position the cursor over the error marker to display the reason for the error:



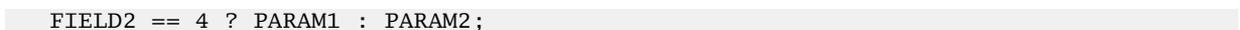
When these errors are corrected, the error marker and the corresponding entries in the Problems view are removed.

Content assist also provides templates for common JavaScript constructs. For example, if you enter **if**, then press **Ctrl + Space**, you can use the following template to construct an **if else** construct:

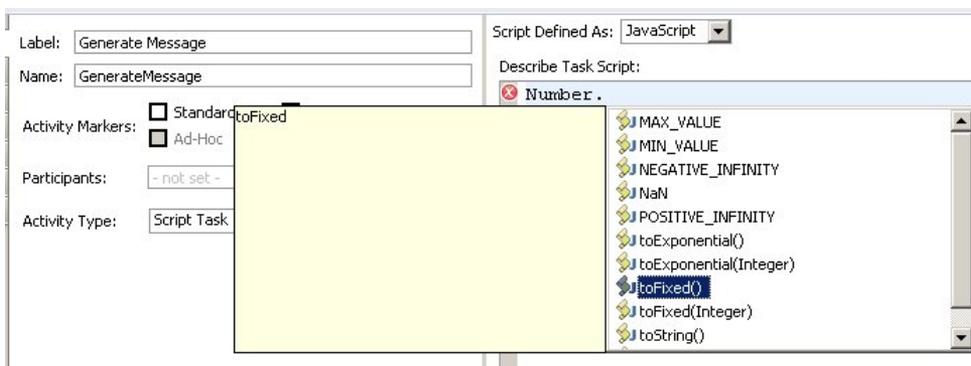
When using the colon character in a JavaScript conditional expression, ensure that you insert spaces before and after the colon. For example, the following expression is not valid:



The corrected expression is:



Content assist similarly provides prompting for the properties of common data types and for data type conversions, as in the following example where the user has selected content assist after typing **Number**.



(See 'Script Functions' in the *Business Data Services Guide* for details on data type transformations supported.)

Additional Functions

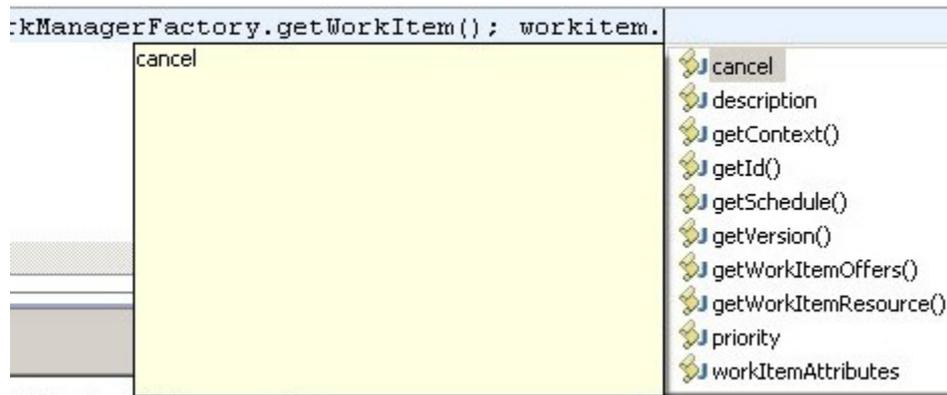
Some additional methods compatible with standard JavaScript are provided with BPM and can be used in the scripts that you write within TIBCO Business Studio. These are described in 'Script Functions' in the *Business Data Services Guide*.

Assistance for Action Scripts

Some action scripts can use data associated with individual work items.

See [Scripts on Other Tasks](#).

For these scripts, content assist provides prompting for certain common constructs involving the work item, as shown in the following illustration.



Using Process Data as Script Variables

You can use the following business process data in scripts: Parameters (whether they are defined at Package or Process level) and Data fields.

Any of these that are added to the script can be used by the script both in the TIBCO Business Studio interface and at runtime once the process containing the script has been deployed to BPM.

By default, all of a process's data fields are available in a script. If an interface is used then this will restrict the fields to just those available in the interface.

Using Structured Data Types

A range of factory methods are supplied for creating instances of structured data types within scripts. You use these methods for creating objects and performing operations on objects of specific types in scripts.

These methods are listed in 'ScriptUtil' in 'Script Functions' in the *Business Data Services Guide*.

The version of JavaScript used by BPM scripting does not support the **new** keyword; instead, you must create new objects using these factory methods.

For example, the following line could be used to initialize a datetime field called **dtYearEnd**, using one of the methods described in 'Script Functions' in the *Business Data Services Guide*:

```
dtYearEnd = DateTimeUtil.createDatetime("2010-12-31T23:59:59");
```

The normal JavaScript arithmetic operators (such as +, -, *, /) are not supported for use with variables containing objects of these types. Instead, you must use the method listed in ScriptUtil in 'Script Functions' in the *Business Data Services Guide* for the underlying Java type of the object.



Fields of the types listed in the table below, such as **Integer(subtype:Fixed)** and **Decimal(subtype:Fixed)** types have to be manipulated in this way; but the basic **Integer(subtype:Signed)** and **Decimal(subtype:Floating Point)** fields can be manipulated using the standard arithmetic operators.

The following table gives a summary of the types affected, with the factory methods used and the underlying Java types:

BPM Object Type	BPM Sub-type	Factory Method to Create	Underlying Java Type
Date		DateTimeUtil.createDate	XMLGregorianCalendar
Time		DateTimeUtil.createTime	XMLGregorianCalendar
Datetime		DateTimeUtil.createDatetime	XMLGregorianCalendar
Datetimetz		DateTimeUtil.createDatetimetz	XMLGregorianCalendar
Duration		DateTimeUtil.createDuration	Duration
Integer	Fixed	ScriptUtil.createBigInteger	BigInteger
Decimal	Fixed	ScriptUtil.createBigDecimal	BigDecimal

For example if you wanted to assign another variable, **dtNextYearStart**, to be one second later than the **dtYearEnd** field that was assigned in the example above, you could add one second to the previous field, like this:

```
dtNextYearStart = ScriptUtil.copy(dtYearEnd);
dtNextYearStart.add(DateTimeUtil.createDuration("PT1S"));
```

or

```
dtNextYearStart = DateTimeUtil.createDatetime(dtYearEnd);
dtNextYearStart.add(DateTimeUtil.createDuration("PT1S"));
```

Care must be taken as the `add()` method on the `XMLGregorianCalendar` updates the object, and does not return a value.

Dynamically Created Factory Methods

In addition, BPM scripting supports the use of dynamically created factory methods to create new Business Object Model objects.

This enables you to populate a data field with an instance of a Class defined in a Business Object Model that is referenced (directly or indirectly) from the business process. You can do this if the first use of that data field is in a Script task; it is not necessary if the field has already been initialized, for example in a form. You can call:

```
field=BOM_name_Factory.createClass();
```

where:

- *field* is the name of the data field to be populated, and must correspond to a process field defined as an External Reference to *class* in the Business Object Model,
- *BOM_name* is the Business Object Model name, with any dots replaced by underscores (so **com.example.BOM** would become **com_example_BOM**),



Note that this is the Name of the model, not the Label.

- *Class* is the name of the Business Object Model Class object,

For example, assume that there is a Business Object Model called **com.myorg.customermodel**, and that it contains a Class called **Customer**. To use a script to create an instance of this Class in the field **cust**, you would call the corresponding factory method as follows:

```
cust = com_myorg_customermodel_Factory.createCustomer();
```

As noted above this only needs to be done if the **cust** field has not already been initialized. You could check to see whether it has been created before invoking this line of the script, for example:

```
if (cust == null)
{
cust = com_myorg_customermodel_Factory.createCustomer();
}
```

The Business Object Model object needs to have been created before any attributes defined in the Business Object Model can be assigned.

See the *TIBCO Business Studio Concepts Guide* and *TIBCO Business Studio Modeling User's Guide* for more details on business object models.

JavaScript Exclusions

Certain facilities of standard JavaScript are not supported in TIBCO Business Studio.

These are:

- The JavaScript **new** operator is not supported for creating new objects. The factory methods described in [Using Structured Data Types](#) and listed in the *Business Data Services Guide* are used instead.
- The JavaScript arithmetical operators are not supported for use with the new data types described in [Using Structured Data Types](#). Instead, use the methods listed in the *Business Data Services Guide* for the appropriate underlying data type.
- You cannot define functions in scripts: that is, the JavaScript **function()** method is not supported. An error saying **Local method definition is not allowed** is generated.

It is not regarded as good practice within TIBCO Business Studio for scripts to be too large or to provide behavior which would be better and more clearly provided by the diagrammed business process. Scripts should provide only the necessary connections between the process, the services it uses, and work items.

- The **switch(){case: default:}** statement is not supported.
- JavaScript regular expressions are not supported.
- The **valueOf()** method is not supported.
You can achieve the same results by using **toString()** instead.
- The Try/Catch statement is not supported.
- The **===** operator is not supported
- **"in"** is a reserved keyword in JavaScript and is not supported.



Note that braces, {..}, are required in an **if** statement and in **for** and **while** loops by the TIBCO Business Studio script editor, although they are only optional in JavaScript.

Although braces are not compulsory in JavaScript **if**, **for** and **while** statements their use is in any case good practice.

Customizing JavaScript Presentation Preferences

You can customize some of the settings for formatting, content assist, and the templates that are used by JavaScript.

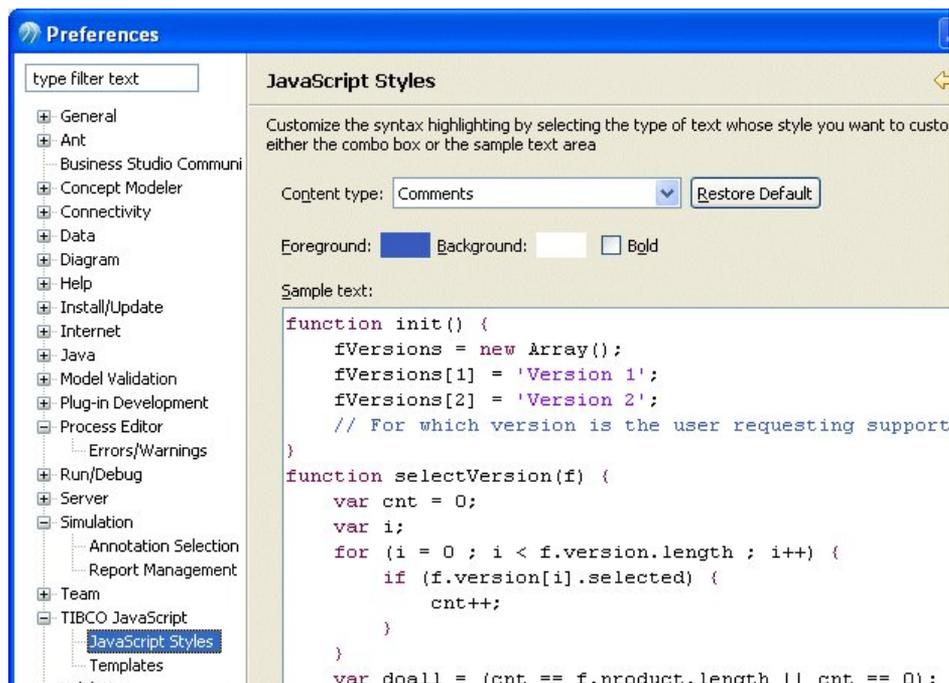
Procedure

1. Select **Window > Preferences**.
2. Select **TIBCO JavaScript**. The following dialog is displayed:



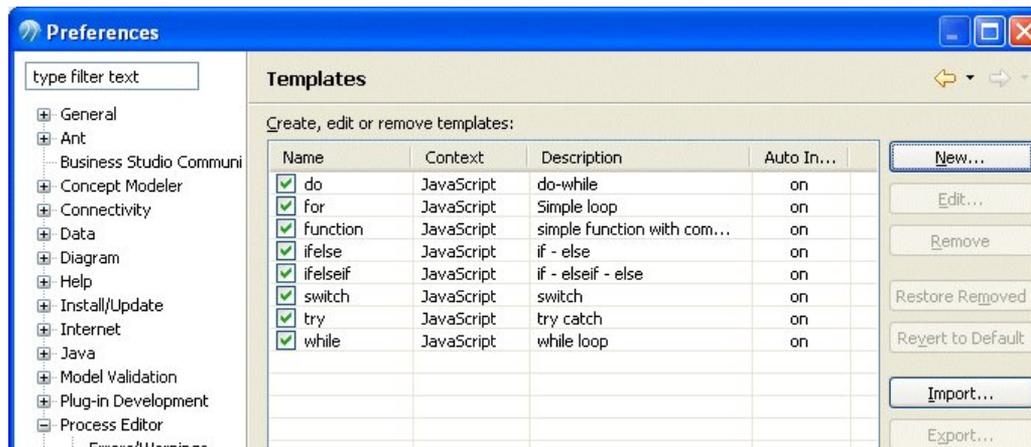
Make any desired changes to the **Formatting** or **Content Assist** sections, then click **Apply**.

3. To customize JavaScript styles, expand **TIBCO JavaScript** and select **JavaScript Styles**. The following dialog is displayed:



From the **Content type** drop-down list, select the type of text whose behavior you want to modify. Make the desired changes and click **Apply**.

- To change the installed templates, expand **TIBCO JavaScript** and select **JavaScript > Templates** . The following dialog is displayed:



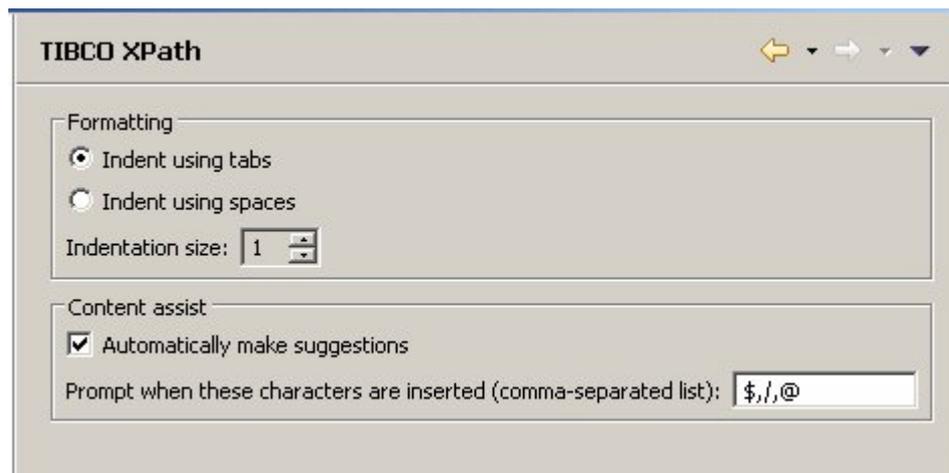
You can edit, modify, create, or import templates using the controls provided. When you have finished making changes, click **Apply**.

Customizing XPath Presentation Preferences

You can customize some of the settings for formatting, content assist, syntax highlighting, and the templates that are used by XPath scripts.

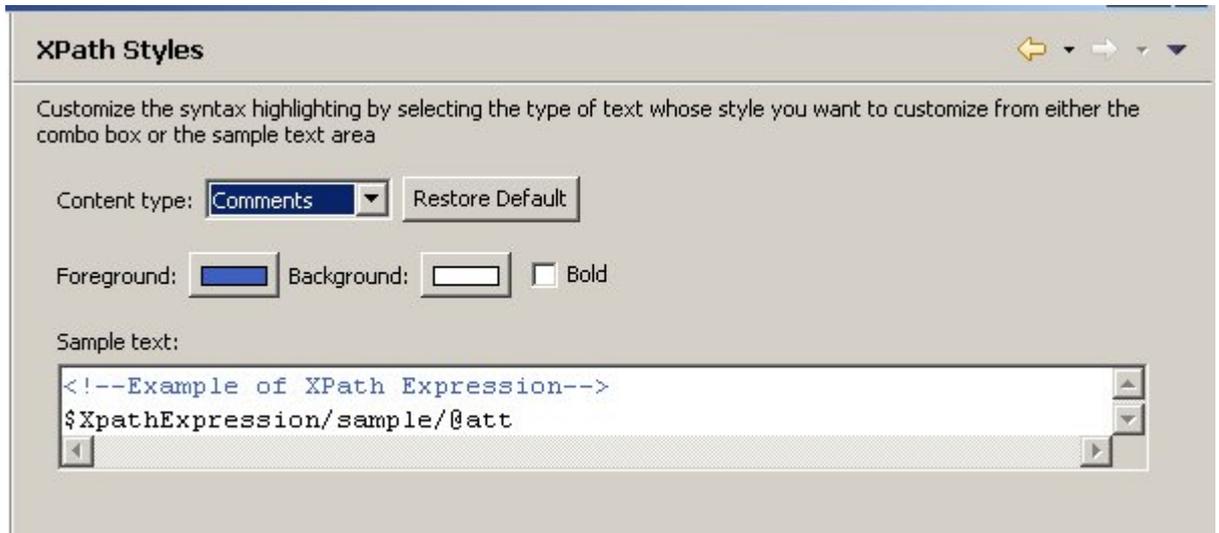
Procedure

- Select **Window > Preferences** .
- Select **TIBCO XPath**. The following dialog is displayed:



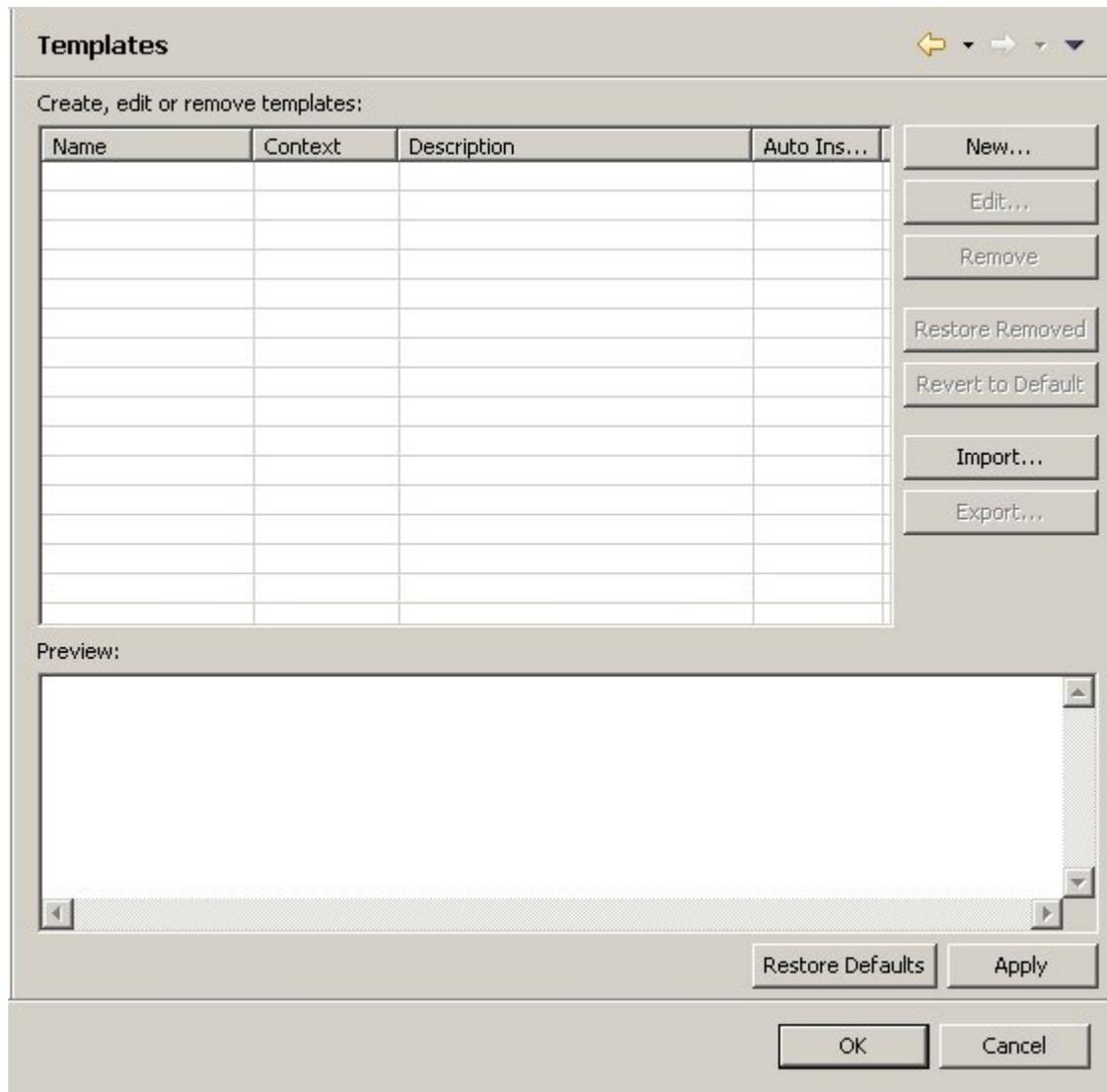
Make any desired changes to the **Formatting** or **Content Assist** sections, then click **Apply**.

- To customize XPath styles, expand **TIBCO > XPath** and select **XPath > Styles** . The following dialog is displayed:



From the **Content type** drop-down list, select the type of text whose behavior you want to modify. Make the desired changes and click **Apply**.

4. To change the installed templates, expand **TIBCO XPath** and select **XPath > Templates** . The following dialog is displayed:



You can edit, modify, create, or import templates using the controls provided. When you have finished making changes, click **Apply**.

Scripts at Runtime

When an application runs under BPM, it produces work items. Each instance of the process that you have designed in TIBCO Business Studio starts with the initial task described in your process and follows the flow of processing that you have defined. The first work item that the process instance presents to the user, therefore, may correspond to the first user task defined in your process. Subsequent forms may be presented to the same user or to someone else.

The work items are assigned to users as described in [Managing Work Using Organization Models](#). Each work item appears in the work list of the user to whom it is assigned. When the user opens that work item, they are presented with a form, or with the first of a series of forms, that enables them to process that user item.

Except in the very simplest of processes, the sequence of forms that the user is presented with, and the outcome of the work item, will depend on the conditional processing written into the application. That conditionality is governed by scripts. For example, a script could take information from fields completed in one work item and pass it to another, depending on the value of those fields.

In addition, scripts can include operations to create, update or delete data objects. Examples would include creating or updating a customer record, or a new insurance claim.

There are two categories of action scripts, one that has access to Work Manager data, and the other that has access to the Process Manager scripting object (see [Scripts on Other Tasks](#)). Action scripts can act directly on work items at run time and can use attributes of those work items as parameters.

The following types of action script are supported:

Types of Action Script	Description
Open Work Item	When a user opens a work item in a work list, any Open script defined for the event will run. If necessary this script can force a cancellation of the work item open and keep the work item in the work list.
Close Work Item	When a user closes a work item in a work list (i.e. saves the state of the work item rather than completing it), any Close script defined for the event will run.
Submit Work Item	When the work item is submitted, after all standard validations have run but before the task completion message is returned, any Submit script defined for the event will run. If necessary this script can force a cancellation of the work item submit and keep the work item open and with the end user.
Schedule Work Item	When the work item is scheduled from an initiated activity, after all standard validations have run but before the task completion message is returned, any Schedule script defined for the event will run. If necessary this script can force a cancellation of the work item schedule and keep the work item open and with the end user.
Initiate Activity	Initiates the user activity within the process that generated the work item.
Complete Activity	Completes the user activity within the process that generated the work item.
Timeout Activity	Executes when the user activity within the process that generated the work item times out.
Cancel Activity	Executes when the user activity within the process that generated the work item is cancelled.

Data Mapping

Data Mapper is a script grammar available for script tasks, task scripts, web-services, sub-processes, and global signals. With Data Mapper, you can graphically map data across datafields and parameters to create complex BOM objects from a combination of process data fields and parameters.



For data mapper, the script created employs a create-or-merge strategy at runtime. If the parent-tree for the target element does not exist, TIBCO Business Studio creates it prior to assigning the mapped element. If there is any invalid mapping in the process data, an appropriate validation error is displayed in the Problems tab.

The process data can include data fields, parameters, user-defined scripts, process information, and work item information. The left hand side of the data mapper shows the source list, and the right hand side shows the target list. The mappings defined using the process data mapper are applied to the target datafields and parameters at runtime.

- Data Mapper is available on Script Tasks as an option in the **Script Defined As:** list on the **General** tab of the Properties view.
- For a Task script, Data Mapper is available as an option in the **Script Defined As:** list in the process manager and work manager scripts in the **Script** tab of the Properties view.
- For a Web Service, Data Mapper is available using the **Input To Service** and **Output From Service** tabs in the Properties view.
- For a Call Sub-Process, Data Mapper is available using the **Map To Sub-Process** and **Map From Sub-Process** tabs in the Properties view.
- For a Catch error event (including Web Service fault catch event, catch-all error events), Data Mapper is available using the **Map From Error** tab in the Properties view.



If you change the script grammar from Data Mapper to JavaScript, TIBCO Business Studio generates a new JavaScript with the data you have already defined in the Data Mapper. In complex mapping scenarios, you can draw the main assignments easily and then modify the result to perform more complex actions. However, changing the grammar from JavaScript to Data Mapper results in loss of the script.



The overall strategy for handling mappings into the children of complex type target data is to create the target element if it does not exist prior to performing assignment to the child element. If the target parent element already exists then the assignment is made to the child of the existing element.



Existing (created before TIBCO Business Studio version 4.1) catch-all error events continue to use the JavaScript mapping selection. New catch-all error events use DataMapper. You can switch between these using the Script Grammar dropdown.

Array Mapping Strategies

Specific array handling strategies can be selected for mappings to multi-instance target data.

Overwrite, append, and merge are the three mapping strategies that are applicable for array types. By default, the target list is overwritten with the source list.

Each mapping strategy affects the target list differently:

Overwrite

When mapping directly between arrays, clears the target array and copies the elements from the source array.

When mapping between children of arrays, clears the target array and creates a new target element for each element in the source array. The assignments implied by the child mappings are then applied to parent elements in the same location in the source and target array.

Append

When mapping directly between arrays, copies elements in the source array to the end of the target array.

When mapping between children of arrays, creates a new element for each element in the source array and appends to the target array. The assignments implied by the child mappings are then applied to new target elements from the source elements.

Merge

When mapping directly between arrays, overwrites each element in the target array for which there is an element at the same location in the source array by a copy of the source element. If there are further elements in the source array then they are appended to the target array.

When mapping between children of arrays, each element in the target array for which there is an element at the same location in the source array will have the assignments implied by the child mappings applied from that source element.

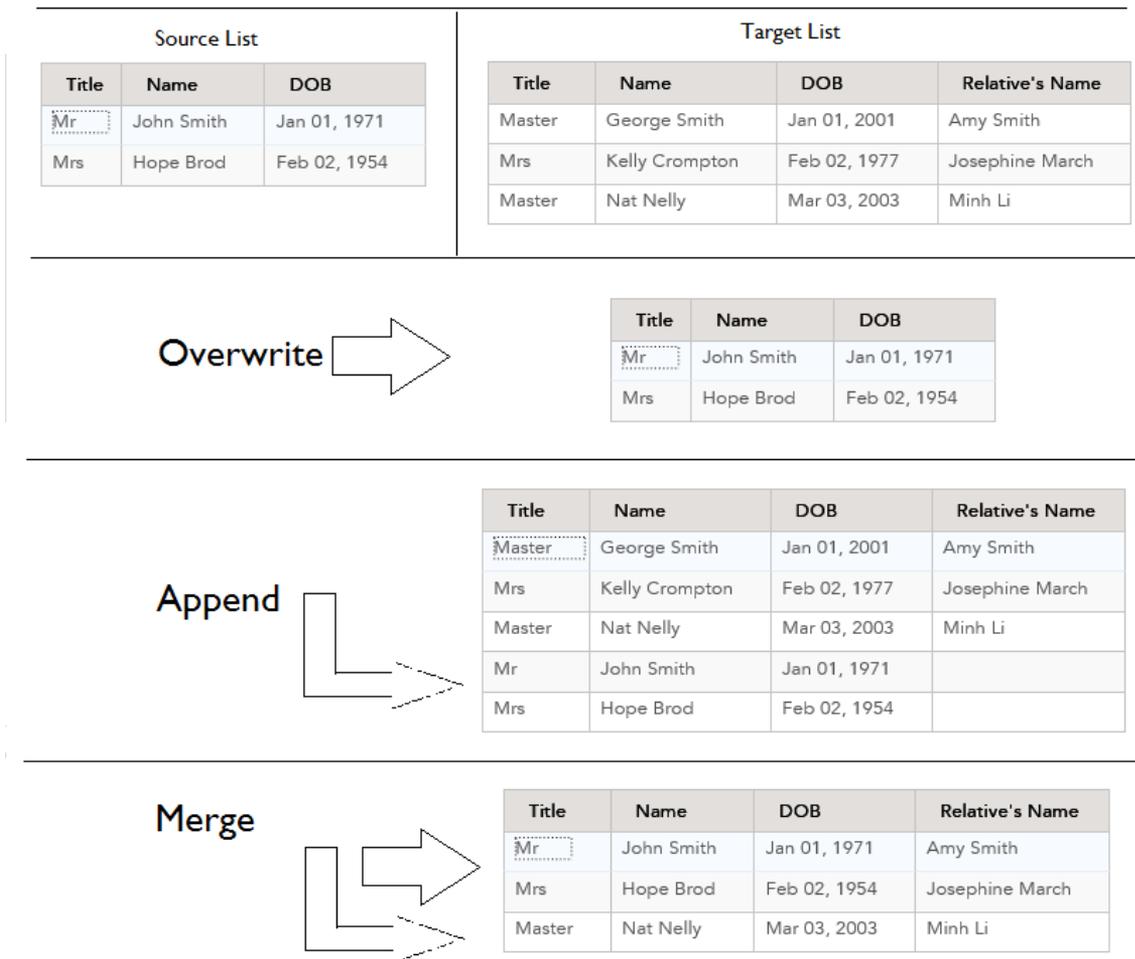
If there are further elements in the source array then new target elements are created and appended to the target array, and the assignments will be applied to the children of these new elements.

Nested Array Handling

In the case where it is desired to map to an array nested within another array, either directly or mapping into the child content, then you must do it from equivalently nested source arrays.

For example, if your target content is an `Orders` array, with a child array of `OrderLines` then in order to map into `Orders[]->OrderLines[]->LineId` then the mapping must be from an element within a second level nesting source array (`MyConfirmations[]->ConfirmationLine[]->LineConfirmId`).

Result of Each Mapping Strategy



Like Mapping

Like mapping is useful in automatically mapping complex elements that are of different types but contain equivalent content.

For instance, a BPM application may have two different types `VisitorDetails` and `PatientDetails` that share some commonly named children. Performing a like mapping between these two objects is the equivalent of manually mapping all the same named simple type content from the source list to the target list.

It is possible to perform like mapping between arrays of complex types. In this case the equivalently-named child content mappings implied by the like mapping are applied to each element in the array according to the chosen array mapping strategy. If there are nested child arrays that would be like-mapped then the array mapping strategy is selected separately for each.

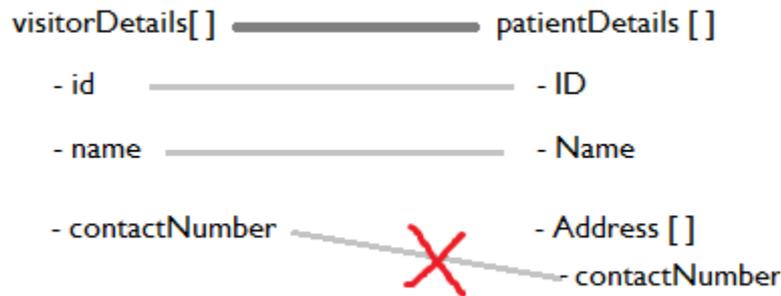
A like mapping scans the target element tree looking for the same named content in the source tree. Initially, all equivalently-named child objects get mapped regardless of their type. If the objects are of different types, the mapping shows an error decoration. You can fix the problem in the BOM definition by excluding certain target child elements from the mapping (as described in the [step to exclude child elements](#)). The overall strategy for handling mappings into the children of complex type target data is to create the target element if it does not exist prior to performing assignment to the child element. If the target parent element already exists then the assignment is made to the child of the existing

element. For mappings from children of source items, the assignment is only performed if the parent itself has a value. If the parent element is not assigned, then the target element remains unchanged.



Like mapping is case-insensitive for the names of child objects.

Like mappings are fully recursive, mapping similarly named composite objects that are within a particular target being mapped. However, only the child objects that are at the same level are included in the like mapping.



If there are nested child arrays that would be like-mapped then the array mapping strategy is selected separately for each array.

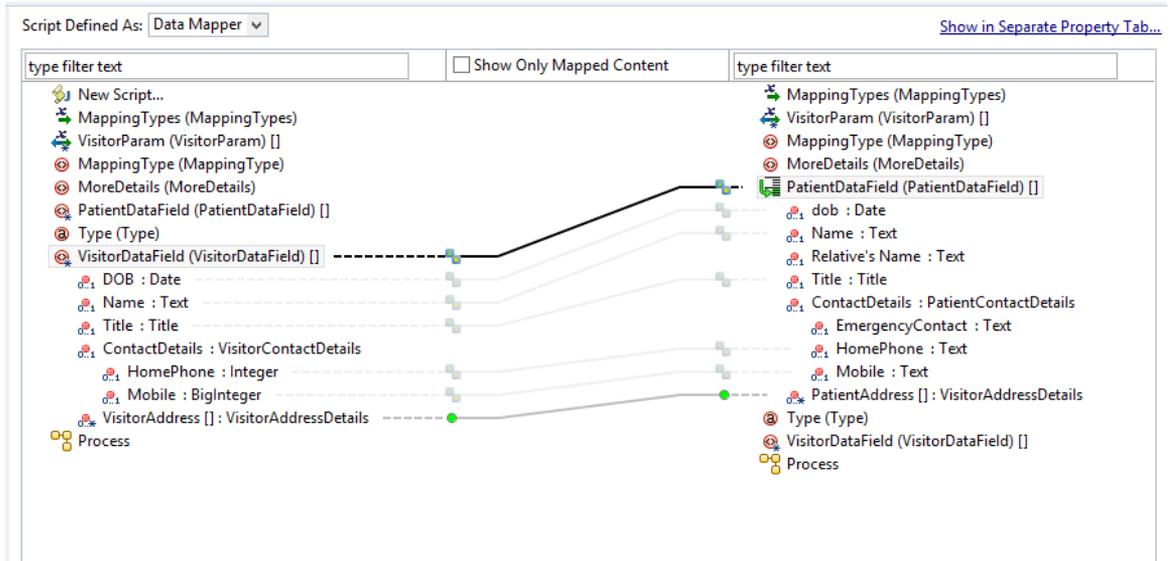
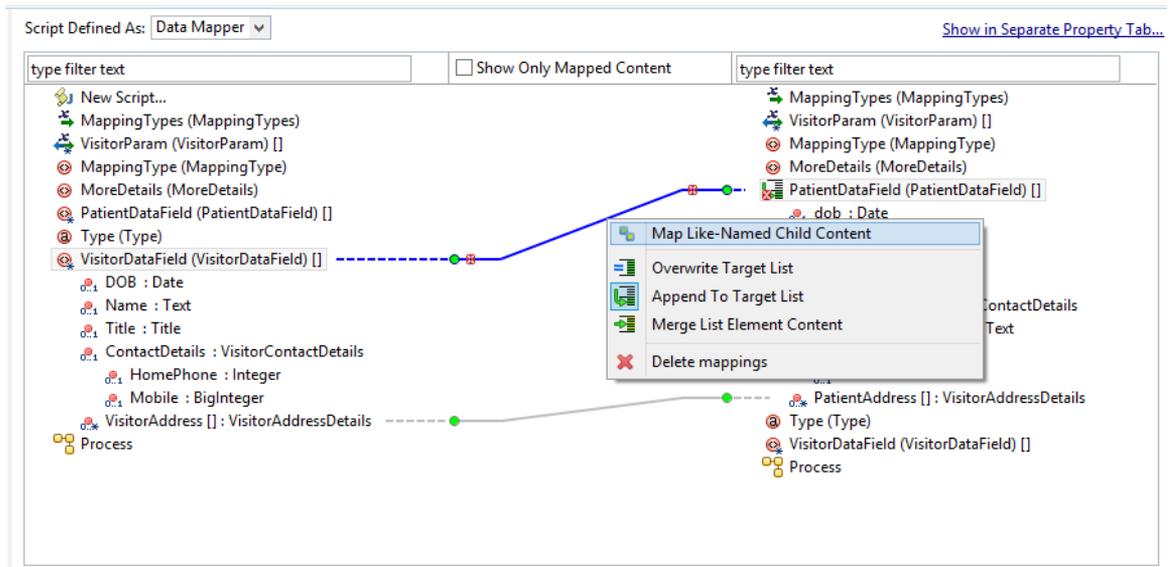
Mapping Contents in Data Mapper

The Data Mapper shows Process and Work Item JavaScript Factory classes as top-level elements on both the sides with the available information to be used for mapping.

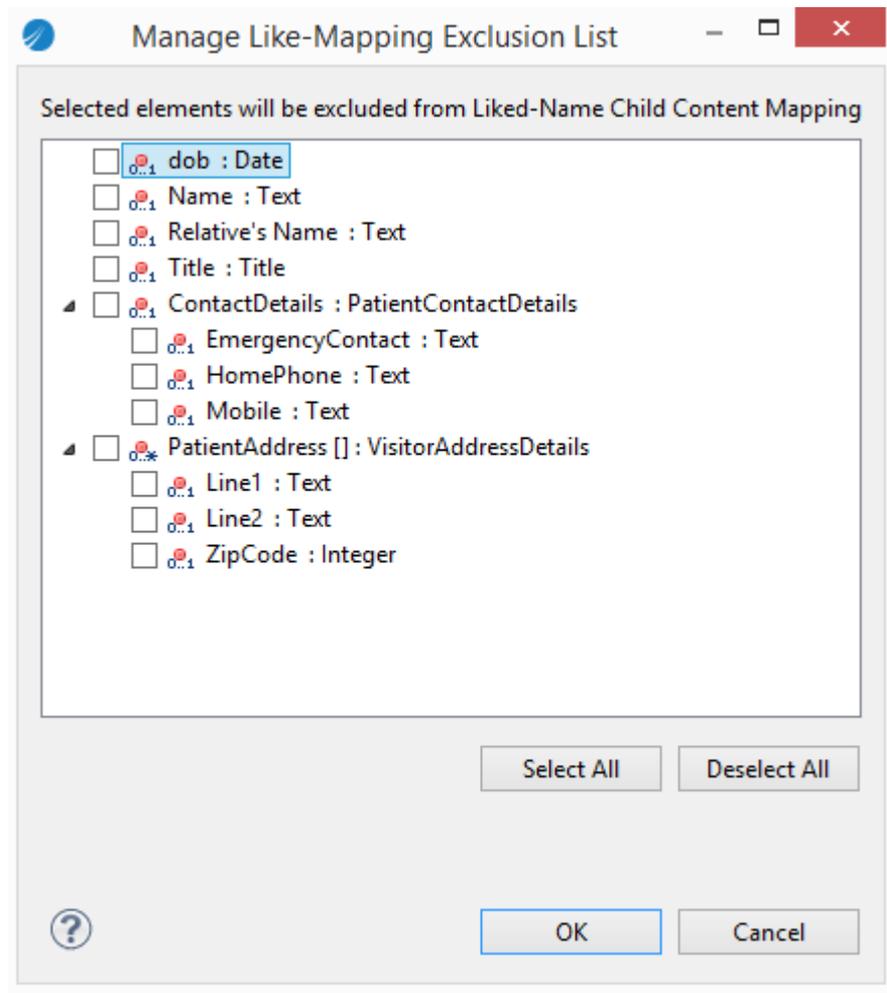
For more information, see [Data Mapping](#).

Procedure

1. Select Data Mapper in the **Script Defined As:** list, that lists the script grammars.
 - For the Script Task, the list is available on the **General** tab of the Properties view. Clicking the link **Show in Separate Property Tab...** displays the **Data Mapper** tab in the Properties view.
 - For a Task script, the Data Mapper grammar is available in the process manager and work manager scripts in the **Script** tab of the Properties view.
 - For a Web Service, Data Mapper is available using the **Input To Service** and **Output From Service** tabs in the Properties view.
 - For a Call Sub-Process, Data Mapper is available using the **Map To Sub-Process** and **Map From Sub-Process** tabs in the Properties view.
 - For a Catch error event (including Web service fault catch event, catch-all error events), Data Mapper is available using the **Map From Error** tab in the Properties view.
2. Drag an element from the source list or LHS and drop it on a corresponding element in the target list or RHS.
A mapping is established between the two elements. If the data types of the two elements being mapped are incompatible, an error decoration is displayed on the mapping.
3. Right-click the mapping and select **Map Like-Named Child Content** from the context menu.
The child elements with the same name and object type are automatically mapped.



4. If you do not want a few child elements mapped with like-named child elements, right-click the mapping at the parent level, and select **Open Like-Mapping Exclusion List**. Manage Like-Mapping Exclusion List dialog opens, which lists all the like-mapped elements.



5. Select the elements you do not want mapped, and click **OK**.
6. Define one of the following mapping strategies by right-clicking the target tree array items:
 - **Overwrite Target List**
 - **Append to Target List**
 - **Merge List Element Content**

For more information on these three options, see [Mapping Strategies](#).

Mapping Process and Work Manager JavaScript Class Attributes

Attributes from the JavaScript classes, `Process` and `WorkManagerFactory` (user task only) are made available for mapping to and from process data in the process manager content.

Executing Java Classes from a Process

This section describes how to call Java classes from a TIBCO Business Studio process by using a service task.

1. Create an Eclipse plugin in your workspace that contains code that you want to call as well as any dependencies.

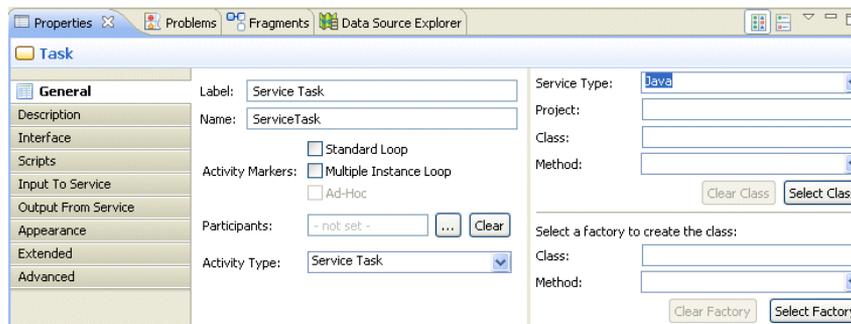


For more information about creating or importing Java projects, see the *Java Development User Guide* in the Eclipse documentation.



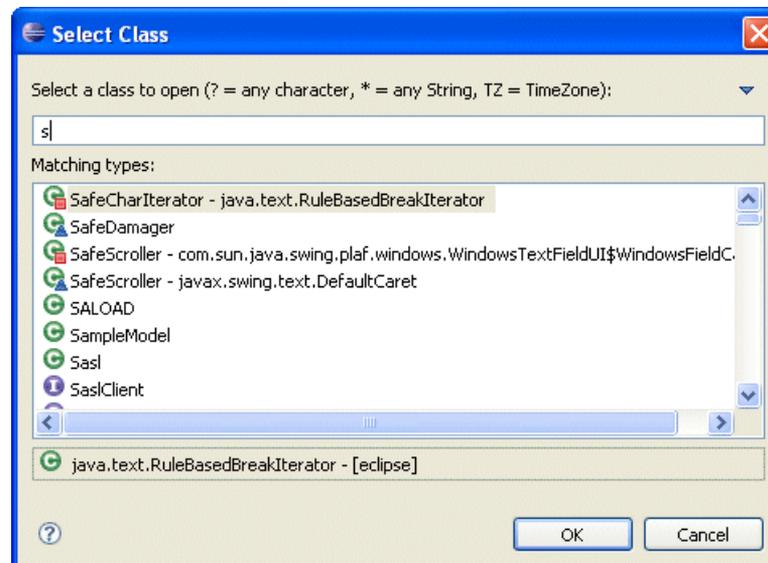
When you use a third party library for your Java projects (which in turn is called from the BPM process) then you should ensure that those third party library/jars are not mentioned on the project classpath (project/runtime/classpath) but they should be mentioned on the manifest file.

2. Select the Service Task, then on the **General** tab of the Properties view for the Service Task, select the **Java** option from the **Service Type** drop-down list:



3. Click **Select Class**. The Select Class dialog is displayed:

An alternative to selecting the Class is to select a factory to create the Class. To do this, click **Select Factory**. If a factory is available, selecting it populates the **Class** and **Method** fields (if the method is a static method with no arguments).



Select the appropriate class that you want to use. If you begin typing, the matching classes are displayed. In the previous example, the character **s** was entered, and all classes starting with **s** are displayed.

4. Select from the drop-down list the **Method** that you want to use. For example:

Service Type:

Project:

Class:

Method:

Select a factory to:

Class:

Method:

5. Click the **Scripts** tab and add any audit scripts you require Only Initiate scripts are supported.



- You must use a factory static method to instantiate the Plain Old Java Objects (POJO).
- JavaBeans and arrays are treated as pass by reference. They must also be "true" JavaBeans in terms of public getters and setters.

Java Deployment

A POJO service is embedded in the DAA generated for the TIBCO Business Studio project.

Complete the Parameter Mapping

This topic covers input and output mapping, and things you should consider when mapping.

You should click the **Input To Service** and **Output From Service** tabs to complete the input and output mapping between any parameters or data fields in your process and the Java code.

You can choose to use automapping to map automatically. To do this, use the automap button above the mappings. See "Automapping" in [Calling a Web Service](#) for more information.

Mapper Restrictions and Notes

- If you attempt to map data fields or parameters to parameters of a different type (for example, mapping a String to an Integer), an error is generated in the Problems view.
- If numbers are being mapped, you must perform a one to one mapping.
- Floats and doubles are limited in precision and TIBCO Business Studio validates the mapping of these data types as follows:

Data Type	Process to Java	Java to Process
Float	6 maximum	6 minimum
Double	15 maximum	15 minimum

From process to Java, errors are generated if these limits are exceeded. From Java to process, warnings are generated if these minimum values are not adhered to.

- Use of system exit is prohibited.
- Mapping of process Decimal numbers to Java Integer numbers can lead to a loss of precision (there is a validation rule for this).
- Mapping of Java Decimal numbers to process Integer numbers can lead to a loss of precision (there is a validation rule for this).

- There is a validation rule that checks that all the input parameters for the POJO service have been mapped.

Web Service Definition Language (WSDL) Documents

This section describes the content and structure of WSDL documents. Web services are described in documents expressed in WSDL.

When interacting with a web service, a process will adopt one of two roles - supplier or consumer:

- A *service supplier* publishes a WSDL document that describes the services it offers.
- A *service consumer* uses the published WSDL document to determine the services offered by the supplier and the messages required to access those services.

Abstract and Concrete WSDL Documents

WSDL documents are described as either *abstract* or *concrete*:

- An *abstract WSDL* document describes what the web service does, but not how it does it or how to contact it. An abstract WSDL document defines:
 - the operations provided by the web service.
 - the input, output and fault messages used by each operation to communicate with the web service, and their format.
- A *concrete WSDL* document adds the information about how the web service communicates and where you can reach it. A concrete WSDL document contains the abstract WSDL definitions, and also defines:
 - the communication protocols and data encodings used by the web service.
 - the port address that must be used to contact the web service.

WSDL Document Structure

There are two types of WSDL documents: abstract and concrete:

- An *abstract WSDL* document defines an abstract messaging model without reference to protocols or encodings.
- A *concrete WSDL* document contains the abstract definitions and the communication protocols and data formats by which the operations defined in the abstract WSDL document can be invoked.

Abstract WSDL Documents

An abstract WSDL document contains the following elements: Definitions, Types, Messages, Parts, Operations and Port types.

- **Definitions**
is the root element. It enumerates the namespaces referenced in the WSDL document and contains all other elements.
- **Types**
describe the data types of the objects that may be passed in messages.
- **Messages**
consist of one or more logical parts. Each part is associated with a type from a type system using a message-typing attribute.
- **Parts**

a mechanism for describing the logical abstract content of a message.

- **Operations**

are composed of sequences of messages. The direction of the messages (input or output) is from the perspective of the service *provider*.

- **Port types**

(also referred to as interfaces) are collections of operations.

The following WSDL document fragment contains the abstract WSDL elements for a stock quote service.

```
<definitions name="StockQuote"
  targetNamespace="http://ns.tibco.com/StockQuote"
  xmlns:tns="http://ns.tibco.com/StockQuote"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  <types>
    <xs:schema          targetNamespace="http://ns.tibco.com/StockQuote/">
      <xs:element name="QuoteRequest" type="xs:string"/>
      <xs:element name="QuoteResponse" type="xs:float"/>
    </xs:schema>
  </types>
  <message name="QuoteInput">
    <part name="symbol" element="tns:QuoteRequest"/>
  </message>
  <message name="QuoteOutput">
    <part name="quote" element="tns:QuoteResponse"/>
  </message>
  <portType name="StockQuotePortType">
    <operation name="getQuote">
      <input message="tns:QuoteInput"/>
      <output message="tns:QuoteOutput"/>
    </operation>
  </portType>
  ...
</definitions>
```

Concrete WSDL Documents

The concrete WSDL document adds the following elements to the abstract WSDL document: Bindings, Ports and Services.

- **Bindings**

connect a port type to a protocol and data format

- **Ports**

(also referred to as endpoints) are comprised of a binding and a network address

- **Services**

are collections of ports

The following WSDL document fragment contains the concrete WSDL elements of the stock quote service (for SOAP over HTTP).

```
<binding name="StockQuoteSoapBinding"
  type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getQuote">
    <soap:operation soapAction="http://ns.tibco.com/getQuote"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
```

```

        <soap:body use="literal"/>
    </output>
</operation>
</binding>
<service name="StockQuoteService">
    <port name="StockQuotePort"
        binding="tns:StockQuoteSoapBinding">
        <soap:address location="http://ns.tibco.com/stockquote"/>
    </port>
</service>

```

A similar WSDL document fragment for SOAP/JMS is as follows:

```

<wsdl:binding name="SOAPService_Binding1"
    type="tns:helloWorld">
    <soap:binding style="document" transport="http://www.tibco.com/namespaces/ws/
2004/soap/binding/JMS" />
    <jms:binding messageFormat="text" />
    <wsdl:operation name="Hello">
    <wsdl:documentation />
    <soap:operation soapAction="Hello" style="document" />
    <wsdl:input>
        <soap:body parts="parameters" use="literal" />
    </wsdl:input>
    <wsdl:output>
        <soap:body parts="parameters" use="literal" />
    </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="NewComponent_1.0.0.201205021319_service_NewService_NewService">
    <wsdl:port binding="tns:SOAPService_Binding1" name="SOAPService_Binding1">
    <soap:address location="" />
        <jndi:context>
            <jndi:property
name="java.naming.factory.initial" type="java.lang.String">com.tibco.tibjms.naming.T
ibjmsInitialContextFactory</jndi:property>
            <jndi:property
name="java.naming.provider.url" type="java.lang.String">tibjmsnaming://uk-n2-base:
7222</jndi:property>
        </jndi:context>
        <jms:connectionFactory>QueueConnectionFactory</jms:connectionFactory>
    <jms:targetAddress destination="queue">queue.sample</jms:targetAddress>
    </wsdl:port>
</wsdl:service>

```

Web Service Configuration Properties

This section describes configuration properties for tasks or events that invoke or expose web service operations, and the associated system participants that define or identify web service endpoints.

Web Service Implementation Properties

The following table describes the fields that appear on the **Properties** page when one of the following tasks or events is defined with a **Service Type** (or, for a message event, **Implementation**) of **web service**: message event (start, catch/throw intermediate or end), service task or send/receive task.

Property	Description
Service Type	Must be Web Service .
Operation	<p><i>This section defines the operation that is to be invoked or exposed by the task or event.</i></p> <p>Click:</p> <ul style="list-style-type: none"> • Select to choose an operation from a WSDL document that already exists in the current workspace. • Clear to clear all the currently selected fields in the Operation and Endpoint Resolution sections. • Import WSDL to import a WSDL document from an external source (a file, URL or service registry), then choose an operation from the imported WSDL document. • Generate WSDL to automatically create a WSDL document (abstract or concrete), an operation and the required data mappings from the data fields and types defined on the Interface tab. <p><i>This option is only available on a service task.</i></p> <ul style="list-style-type: none"> • Set Default to reset the task or event to use the default web service operation. <p><i>This option is only available if the task or event is exposing a web service, and if the default WSDL document is not already selected.</i></p>
Port Type	<p>The port type (set of operations) that contains the Operation Name operation, as defined by the name attribute of the <code>portType</code> definition in the WSDL document.</p> <p>The <code>portType</code> definition is part of the abstract WSDL document.</p>
Operation Name	<p>The operation that the process wishes to call or expose, as defined by the name attribute of the <code>operation</code> definition in the WSDL document.</p> <p>The <code>operation</code> definition is part of the abstract WSDL document.</p>
Port Name	<p>The port that defines the binding (protocol and data format) and network address used by the Service Name service, as defined by the name attribute of the <code>port</code> definition in the WSDL document.</p> <p>The <code>port</code> definition is part of the concrete WSDL document. (It is therefore not shown for an operation selected from an abstract WSDL document.)</p>

Property	Description
Service Name	<p>The service that contains the Port Name port, as defined by the name attribute of the <code>service</code> definition in the WSDL document.</p> <p>The <code>service</code> definition is part of the concrete WSDL document. (It is therefore not shown for an operation selected from an abstract WSDL document.)</p>
Transport	<p>The transport mechanism used by the Service Name service.</p> <p>The available transport options are Service Virtualization, SOAP over HTTP, SOAP over JMS.</p>
Endpoint Resolution	<p><i>This section defines how the location of the web service will be resolved at runtime.</i></p>
WSDL	<p>Defines whether a local or remote WSDL will be used. For BPM, the Use remote option is automatically set when the WSDL document is selected.</p> <p>The Use local option is not supported by BPM.</p>
Location	<p>Defines the location of the WSDL document that defines the specified Operation.</p> <p>The field displays the string "This is taken from the Alias URL at Run-time". This indicates that the <code>soap:address location</code> element specified in the WSDL file will not be used at runtime.</p> <p>The runtime connection to the web service is defined by BPM.</p>
Endpoint Name	<p>The name of the system participant that is used to either:</p> <ul style="list-style-type: none"> • define the web service endpoint to be provided by the task or event, or • identify the web service endpoint to be called by the task or event. <p>See System Participant Shared Resource Properties for more information about the configuration of this system participant.</p>
Security Profile	<p>This option is currently not used by BPM.</p>

See [Web Service Definition Language \(WSDL\) Documents](#) for more information about the content and structure of WSDL documents.

System Participant Shared Resource Properties

The following table describes the fields that appear on the **Shared Resource** section on the **Properties** page for a system participant that is assigned to a task or event that calls or exposes a web service.

Acting As	Description
Provider	<p>Displays the different binding types that will be used to expose the service to a client application. These are:</p> <ul style="list-style-type: none"> • Virtualization - There are no further properties for this binding type. • SOAP over HTTP - When this option is selected, the binding details are displayed - see SOAP over HTTP Binding Details (Provider) . • SOAP over JMS - When this option is selected, the binding details are displayed - see SOAP over JMS Binding Details (Provider) .
Consumer	<p>Displays an Invoke Using section, which displays the binding type that will be used to call the service. These are:</p> <ul style="list-style-type: none"> • Virtualization - There are no further properties for this binding type. • SOAP over HTTP - When this option is selected, the binding details are displayed - see SOAP Over HTTP Binding Details (Consumer) . • SOAP over JMS - When this option is selected, the binding details are displayed - see SOAP Over JMS Binding Details (Consumer) .

SOAP over HTTP Binding Details (Provider)

The following table describes the binding details that are displayed for the selected SOAP over HTTP binding, where the shared resource is acting as a provider.

Property	Description
Binding Name	Defines the name used to identify this binding. The default value is SoapOverHttp .
Style/Use	<p>Defines the style/use used by the SOAP binding. One of:</p> <ul style="list-style-type: none"> • RPC/literal • Document/literal <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  If a concrete WSDL document is being used to expose the service, the style/use used here should match the style/use defined in the WSDL document. If the two differ, at runtime the style/use defined in the WSDL document takes precedence and the style/use defined here will be ignored. </div>
SOAP Version	<p>Defines the SOAP version used by this binding. SOAP versions 1.1 and 1.2 are supported.</p> <p>The default value is 1.1.</p>

Property	Description
Endpoint URI Path	<p>Defines the URI <i>path</i> component of the endpoint address that will be used to expose the service. The full endpoint address is defined as:</p> <p style="text-align: center;"><i>protocol://host:port/path</i></p> <p>The construction of the endpoint address depends upon whether a concrete or an abstract WSDL document is being used to expose the web service - see SOAP over JMS Binding Details (Provider) .</p> <p>If you use multiple bindings to expose the same service (for example, one for SOAP 1.1 and one for SOAP 1.2, each binding must have a unique Endpoint URI path.</p>
HTTP Connector Instance	<p>Logical name to identify the HTTP Connector resource instance in the BPM runtime that will be used to expose the service to client applications.</p> <p>The default value is httpConnector, which is the name of the default resource instance used by the BPM runtime to expose any services to clients over an HTTP connection.</p> <p>If you wish to use a different HTTP Connector resource instance, you can do so using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable HTTP Connector resource instance that already exists on the BPM runtime. (The mapping to the HTTP Connector resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the HTTP Connector resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the HttpInboundConnectionConfig property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator will need to configure the HTTP Connector resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>

Endpoint Address Construction

The construction of the endpoint address depends upon the type of WSDL document being used to expose the web service.

Generated (Abstract) WSDL Document

The URI *path* has the following format:

ContextRoot/PackageName/ProcessName

where:

- *ContextRoot* is an (optional) project-specific or workspace-specific prefix. See [Setting a Common Context Root for Web Service Endpoint URIs](#) for more information.
- *PackageName* is the name (not the label) of the parent package that exposes the web service.
- *ProcessName* is the name (not the label) of the parent process that exposes the web service.

The following example shows the URI path generated for a process called **ClaimsProcess** in a package called **ProcessPackage**, in a project which has a context root defined as **/EasyAs/BPM** (either at the project or workspace level):

/EasyAs/BPM/ProcessPackage/ClaimsProcess

The *protocol*, *host* and *port* components of the endpoint address are taken from the runtime configuration of the HTTP Connector resource template that is referenced by the [HTTP Connector Instance](#) property.

A Different Abstract WSDL Document

The URI *path* has the following format:

WSDLfileName/portType

where:

- *WSDLfileName* is the name of the WSDL file.
- *portType* is the name of the selected operation's parent portType.

Concrete WSDL Document

The URI *path* is taken from the `soap:address location` element in the WSDL document.

The *protocol*, *host* and *port* components in the `soap:address location` element in the WSDL document address are taken from the runtime configuration of the HTTP Connector resource template that is referenced by the property.

SOAP over JMS Binding Details (Provider)

The following table describes the binding details that are displayed for the selected SOAP over JMS binding, where the shared resource is acting as a provider.



Topic destinations are not supported for SOAP/JMS bindings.

Property	Description
Binding Name	Defines the name used to identify this binding. The default value is SoapOverJms .
Style/Use	<p>Defines the style/use used by the SOAP binding. One of:</p> <ul style="list-style-type: none"> • RPC/literal • Document/literal <p> If a concrete WSDL document is being used to expose the service, the style/use used here should match the style/use defined in the WSDL document. If the two differ, at runtime the style/use defined in the WSDL document takes precedence and the style/use defined here will be ignored.</p>
SOAP Version	<p>Defines the SOAP version used by this binding. SOAP versions 1.1 and 1.2 are supported.</p> <p>The default value is 1.1.</p>

Property	Description
Inbound: Connection Factory Configuration	<p>Logical name to identify a JMS Connection Factory Configuration resource instance in the BPM runtime. It creates an inbound connection to a JMS server to enable inbound receipt of JMS messages.</p> <p>The default value is amx.bpm.userapp.jmsConnFactoryConf.</p> <p>You can bind this logical name to the appropriate JMS Connection Factory Configuration resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable JMS Connection Factory Configuration resource instance that already exists on the BPM runtime. (The mapping to the JMS Connection Factory Configuration resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the JMS Connection Factory Configuration resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the JmsInboundConnectionFactory property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator will need to configure the JMS Connection Factory Configuration resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>
Inbound: Destination Configuration	<p>Logical name to identify a JMS Destination Configuration resource instance in the BPM runtime. It specifies what queue to listen to for inbound messages.</p> <p>The default value is amx.bpm.userapp.jms.request.conf.</p> <p>You can bind this logical name to the appropriate JMS Destination Configuration resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable JMS Destination Configuration resource instance that already exists on the BPM runtime. (The mapping to the JMS Destination Configuration resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the JMS Destination Configuration resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the JmsInboundDestination property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator will need to configure the JMS Destination Configuration resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>

Property	Description
Outbound: Connection Factory	<p>Logical name to identify a JMS Connection Factory resource instance in the BPM runtime. It is used to create an outbound connection to a JMS server.</p> <p>The default value is amx.bpm.userapp.jmsConnFactory.</p> <p>You can bind this logical name to the appropriate JMS Connection Factory resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable JMS Connection Factory resource instance that already exists on the BPM runtime. (The mapping to the JMS Connection Factory resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the JMS Connection Factory resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the JmsOutboundConnectionFactory property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator will need to configure the JMS Connection Factory resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>

SOAP Over HTTP Binding Details (Consumer)

The following table describes the binding details that are displayed for the selected SOAP over HTTP binding, where the shared resource is acting as a consumer.

Property	Description
HTTP Client Instance	<p>Logical name to identify the HTTP Client resource instance in the BPM runtime that will be used to call the web service.</p> <p>The default value is the name of the system participant.</p> <p>You can bind this logical name to the appropriate HTTP Client resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable HTTP Client resource instance that already exists on the BPM runtime. (The mapping to the HTTP Client resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the HTTP Client resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the HttpOutboundConnectionConfig property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator will need to configure the HTTP Client resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>
<i>Security Configuration:</i>	<p>This section defines the security configuration to be applied to the binding. You should obtain the required information for the following fields from the web service provider and/or the administrator of your <i>BPM</i> runtime. See Configuring Security on an Outgoing Service Call for more information.</p>
Policy Type	<p>Defines the type of security policy required to invoke the service - one of the following values:</p> <ul style="list-style-type: none"> • None - to invoke an unsecured service. (This is the default value.) • Username Token, X509 Token or SAML Token - to authenticate the outgoing SOAP request using a Web Services Security (WSS) token of the indicated type. • Custom Policy - to apply a custom security policy to the outgoing SOAP request and, if required, to the incoming SOAP response. <p> You must use a Custom Policy if the SOAP response message returned by the service contains a security header. The Username Token, X509 Token or SAML Token policies do not handle an incoming SOAP response that contains a security header.</p>

Property	Description
Governance App. Name	<p>Defines the name of the identity provider application from which the BPM runtime will obtain the authentication information needed to contact the service.</p> <p>The BPM runtime will use this information to construct the WSS token (of the specified type) that will be used to authenticate the outgoing SOAP request.</p> <p>This field must be completed (and is only displayed) if Policy Type is set to Username Token, X509 Token or SAML Token.</p>
Custom Policy Set	<p>Defines the name of an <i>external policy set</i> that the BPM runtime will apply to the outgoing SOAP request (and, if appropriate, to the incoming SOAP response).</p> <p>This field must be completed (and is only displayed if) Policy Type is set to Custom Policy.</p> <p>The external policy set:</p> <ul style="list-style-type: none"> • must contain the security information required to construct the outgoing SOAP request and, if appropriate, to also handle the resultant incoming SOAP response. • must be defined in an XML file (with the extension .policysets) that is available in the same workspace. <p>Note: TIBCO Business Studio does not validate whether the external policy set is applicable to and correct for the target service. Using an incorrect policy type or a wrongly configured policy will result in an error, either during DAA configuration or at runtime.</p> <p>The BPM runtime supports a wide range of policies and policy sets that can be used to address different security requirements and scenarios. For more information about external policy sets and how to create them, see the following topics:</p> <ul style="list-style-type: none"> • "Policy Management", in <i>Composite Development</i> • "Security Resource Templates", in <i>SOA Administration</i>. (This guide is not included in the TIBCO Business Studio documentation set. You can access it either from the BPM runtime documentation set, or from the Help in the Administrator interface in the BPM runtime.)

SOAP Over JMS Binding Details (Consumer)

The following table describes the binding details that are displayed for the selected SOAP over JMS binding, where the shared resource is acting as a consumer.



Topic destinations are not supported for SOAP/JMS bindings.

The binding details define the JMS Destination and JNDI Connection resource instances that will be used to contact the web service. For more information about these resource instances and how to create them, see the following topics:

- Resource Templates > JMS Resource Templates, in the SOA Administration guide (supplied as part of the BPM runtime documentation set)
- Bindings > JMS Bindings, in the *Composite Development* guide

Property	Description
Inbound Destination	<p>(Optional) Logical name to identify a JMS Destination resource instance in the BPM runtime. This resource instance identifies the JMS queue that is used to obtain the output data from the web service.</p> <p>The default value is amx.bpm.userapp.jmsDestOutbound.</p> <p>You can bind this logical name to the appropriate JMS Destination resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable JMS Destination resource instance that already exists on the BPM runtime. (The mapping to the JMS Destination resource instance is then done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the JMS Destination resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the JmsInboundDestination property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator must configure the JMS Destination resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p> <p>If an Inbound Destination name is not specified, the service provider must send the response using an internal queue.</p>

Property	Description
Outbound Connection Factory	<p>Logical name to identify a JNDI Connection resource instance in the BPM runtime. This resource instance provides the connection details for the JMS server that hosts the inbound and outbound destinations.</p> <p>The default value is amx.bpm.userapp.jmsDest.</p> <p>You can bind this logical name to the appropriate JNDI Connection resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable JNDI Connection resource instance that already exists on the BPM runtime. (The mapping to the JNDI Connection resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the JNDI Connection resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the JmsOutboundConnectionFactory property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator must configure the JNDI Connection resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>

Property	Description
Outbound Destination	<p>Logical name to identify a JMS Destination resource instance in the BPM runtime. This resource instance identifies the JMS queue that will be used to send the input data to the web service.</p> <p>The default value is amx.bpm.userapp.jmsDest.</p> <p>The name specified here must match the one specified for the provider in the Inbound Destination Configuration. See SOAP over JMS Binding Details (Provider).</p> <p>You can bind this logical name to the appropriate JMS Destination resource instance using either of the following methods:</p> <ul style="list-style-type: none"> • Early binding: Replace the name here with the name of a suitable JMS Destination resource instance that already exists on the BPM runtime. (The mapping to the JMS Destination resource instance will then be done automatically when you deploy the application to the BPM runtime.) • Late binding: Change or create the JMS Destination resource instance to be used when you deploy the application to the BPM runtime. You do this by changing the value assigned to the JmsOutboundDestination property on the Property Configuration page of the DAA Deployment Wizard. See Using Pageflow Processes and Business Services for more information about how to do this. <p>If you instead export the project to a Distributed Application Archive for subsequent upload to the BPM runtime, a BPM administrator will need to configure the JMS Destination resource instance to be used. See the Administrator interface documentation for your BPM runtime environment for more information about this.</p>
<i>Security Configuration:</i>	<p><i>This section defines the security configuration to be applied to the binding. You should obtain the required information for the following fields from the web service provider and/or the administrator of your BPM runtime. See Configuring Security on an Outgoing Service Call for more information.</i></p>
Policy Type	<p>Defines the type of security policy required to invoke the service - one of the following values:</p> <ul style="list-style-type: none"> • None - to invoke an unsecured service. (This is the default value.) • Username Token, X509 Token or SAML Token - to authenticate the outgoing SOAP request using a Web Services Security (WSS) token of the indicated type. • Custom Policy - to apply a custom security policy to the outgoing SOAP request and, if required, to the incoming SOAP response. <p>Note: You must use a Custom Policy if the SOAP response message returned by the service contains a security header. The Username Token, X509 Token or SAML Token policies do not handle an incoming SOAP response that contains a security header.</p>

Property	Description
Governance App. Name	<p>Defines the name of the identity provider application from which the BPM runtime will obtain the authentication information needed to contact the service.</p> <p>The BPM runtime will use this information to construct the WSS token (of the specified type) that will be used to authenticate the outgoing SOAP request.</p> <p>This field must be completed (and is only displayed) if Policy Type is set to Username Token, X509 Token or SAML Token.</p>
Custom Policy Set	<p>Defines the name of an <i>external policy set</i> that the BPM runtime will apply to the outgoing SOAP request (and, if appropriate, to the incoming SOAP response).</p> <p>This field must be completed (and is only displayed if) Policy Type is set to Custom Policy.</p> <p>The external policy set:</p> <ul style="list-style-type: none"> • must contain the security information required to construct the outgoing SOAP request and, if appropriate, to also handle the resultant incoming SOAP response. • must be defined in an XML file (with the extension .policysets) that is available in the same workspace. <p>Note: TIBCO Business Studio does not validate whether the external policy set is applicable to and correct for the target service. Using an incorrect policy type or a wrongly configured policy will result in an error, either during DAA configuration or at runtime.</p> <p>The BPM runtime supports a wide range of policies and policy sets that can be used to address different security requirements and scenarios. For more information about external policy sets and how to create them, see the following topics:</p> <ul style="list-style-type: none"> • "Policy Management", in <i>Composite Development</i> • "Security Resource Templates", in <i>SOA Administration</i>. (This guide is not included in the TIBCO Business Studio documentation set. You can access it either from the BPM runtime documentation set, or from the Help in the Administrator interface in the BPM runtime.)
<i>Message Configuration:</i>	<p><i>This section defines the message configuration to be applied to the binding - see Configuring SOAP/JMS Message Timeouts on a Request-Response Operation for more information about the use of the timeouts.</i></p> <div style="border-left: 1px solid black; padding-left: 10px; margin-top: 10px;">  <p>If you migrate a pre-4.0 version of a process to this version, default values are <i>not</i> automatically set for these properties. You must set them manually.</p> </div>
Request-Response Timeout	<p>Time (in seconds) after which a web service call will time out if a response message has not been received from the web service. If this occurs, a Timeout exception error is thrown by the web service task.</p> <p>Default value: 6</p>

Property	Description
Request Expiration Timeout	<p>Time (in seconds) within which the called web service must pull the <i>request</i> message from the JMS message queue. If this timeout expires, the JMS server will be instructed to purge the request message from the JMS message queue. (This ensures that if the web service call is retried, the JMS message queue does not contain duplicate copies of the same request message for the web service to consume.) The web service call itself does not timeout when this timer expires, so no Timeout exception error is thrown.</p> <p>Default value: 3</p>
Delivery Mode	<p>Delivery mode of the message. One of the following values:</p> <ul style="list-style-type: none"> • Persistent: Messages are stored and forwarded. • Non-persistent: Messages are not stored and could be lost due to failures in transmission. <p>Default value: Persistent</p>
Priority	<p>Priority of the message, in the range 0 (lowest) to 9 (highest).</p> <p>Default value: 4 (Normal)</p>

Generating a DAA from the Command Line using an Ant Task

Procedure

1. In TIBCO Business Studio, create any type of project, for example **File > New Project > General > Project**.
2. Inside the project, create a file called **build.xml** (**New > File > build.xml**)
3. Open this file and type your script, similar to the following:



Press **Ctrl + Space** in an empty editor and choose **Buildfile template** to give you an initial and buildfile template.

```
<project name="project" default="default">
  <target name="default" description="Test of the tbs.import feature">
    <tbs.importProjects dir="C:\220\V22Build\JIRAs\XPD5062\WS01\HelloWorld01"></tbs.importProjects>
    <bpm.generateDAA daalocation="C:\220\V22Build\JIRAs\XPD5062\DAALocation"/>
  </target>
</project>
```

- You can use all standard Ant tasks and TIBCO Business Studio specific tasks, starting with **tbs.**, **sds.** or **bpm.**



Type in **tbs.**, **sds.** or **bpm.** and press **Ctrl + Space** for content assist.



bpm.generateDAA is used to create DAAs from BPM projects and **sds.createDAA** is used to create DAAs from SOA composite applications.

You can use the **Buildbeforegenerating** flag, set to **true**, to perform a clean build of the Workspace before generating the DAA's Workspace to make sure everything has already been built. For example:



```
bpm.generateDAA buildbeforegenerating="true" projectname="Utilities"
daalocation="${env.PROJECTLOC}"/>
```

However, this will slow down the generation and is not a required process since an incremental build is performed while importing the processes.

We recommend that you use **tbs.importProjects** rather than **sds.importProject** for BPM projects, as it provides far more stable post import migration and building.

sds.importProject is included for backward compatibility and **tbs.importProjects** is recommended for new scripts.

If you use **tbs**, the following options are available:

- **dir** - The root directory to use for simple format folder structure imports.
- **file** - The archive file to use for simple format archive imports (including zip, tar, tar.gz)
(**dir** and **file** are mutually exclusive)
- **copyProjects** - (defaults to true) Ensures a copy of any imported project is made to the workspace. Applicable only to folder structure imports.
- **useArchives** - (defaults to false) Ensures archive import is attempted when using nested element format.

When a simple format archive import, using the **file** attribute, is used the **useArchives** attribute is not required - it is implicitly set to true.
- **skipPostImportTask** - (defaults to false) Post import tasks are skipped when set to true.



- The example above uses 2 tasks: the first is used for importing a project from a specified location (**projectloc**) into the workspace and the second is used to generate DAA/s from the workspace project/s into a specified location (**daalocation**).
- The above script imports a single project, but you could also import a number of projects by specifying the parent folder in the script. For example:

```
<target name="default" description="Test to generate DAA for all projects under projectloc">
<tbs.importProjects dir="C:/workspace-3.5.0"/>
<bpm.generateDAA daalocation="C:/daaOut"/>
</target>
```

- By default the **generateDAA** task will generate DAA/s for all relevant projects in the workspace.



Command line generation does not overwrite the workspace, therefore every time the command is run, you must ensure that you provide an empty project workspace.

Also note that before importing any projects please ensure that they have been migrated first using the user interface, as command line generation does not auto migrate.

Using External Tools to run an Ant task within TIBCO Business Studio

You can run an Ant script from the command line or as an automated process.

From the command line

You can run the Ant script from the command line. You have to do this by using **amx_eclipse_ant.exe** which is located in the *installation directory* `/studio/3.7/eclipse` directory (which is in the same directory as the TIBCO Business Studio `.exe`).

For example:

```
D:\apps\TIBCO\studio-bpm-3.5-V24\studio\3.5\eclipse>amx_eclipse_ant.exe -f C:\Users\Tester\workspace-3.5.0-V24\TestProject\build.xml -data c:\tempws
```

This starts Eclipse and executes the Ant script in that environment (this may take some time). It creates a `.daa` in the output directory (in this example this is called `daaOut`).

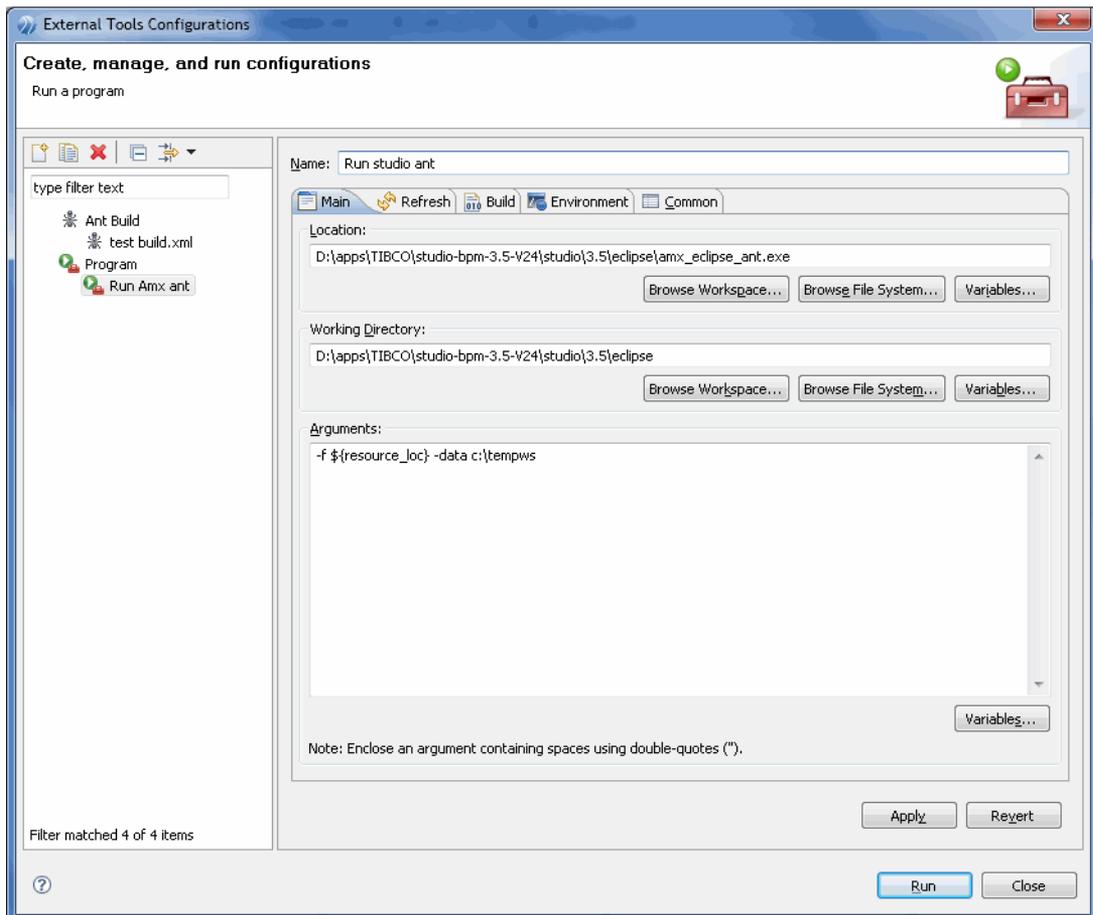
You need to specify your Ant script location after `-f` and the location of your workspace after `-data` (This will be a temporary workspace used during this execution. It does not have to exist previously.).

As an automated process

You can use Eclipse External Tools to run the Ant script from within TIBCO Business Studio.

Go to **Run > External Tools > External Tools Configurations...** and create new **Program**

The example shows the values to enter to run an equivalent of the above command line example:



Resource Query Language

The Resource Query Language (RQL) is used to identify resources within the BPM destination environment that meet a defined set of criteria. A RQL query returns a set of resources that match the criteria expressed in the query. Work can then either be allocated to one of those resources, or offered to multiple individual resources.

RQL is dynamic, and is evaluated when the work item is created and whenever it changes. This means that if the items referred to by the RQL change in some way (for example if the resources mapped to an organizational position are changed) this will be reflected in the set of resources associated with the work item.

See [Assigning Participants to a User Task](#) and the *TIBCO Business Studio Modeling User's Guide* for details of how RQL is used.

You may want to consider whether to use Resource Query Language or Dynamic Organization Participants. See "Dynamic Organization Participants" in the *TIBCO Business Studio Concepts Guide*.

RQL Expression Evaluation

You can specify the way in which RQL expressions are evaluated, which can have a significant impact on the amount of time it takes to evaluate the expressions. This is done by specifying the version of the RQL expression algorithm to use.

The RQL expression algorithm is available in two versions:

- 1 - This is the original algorithm in which RQL expressions are resolved in memory, retrieving resources as the algorithm traverses the elements in the expression.
This version of the algorithm can result in expressions taking a long time to execute. Use this version if you are using a small number of RQL expressions, against a small number of resources.
- 2 - (Default) In this version, the RQL is translated to SQL. Resources are retrieved based on the final SQL statement.

This version of the algorithm results in much faster processing.

Specifying the RQL Expression Algorithm Version

The RQL expression algorithm version is specified using the **ResourceQueryVersion** property in the `de.properties` file.

If you change the RQL expression algorithm from version 1 to version 2, you must then thoroughly test your application to ensure that the RQL-to-SQL algorithm returns the same result set as the previous algorithm. If you find that the result set is different, you may want to revert to version 1 of the algorithm (see below).



If you have performed a new installation of ActiveMatrix BPM version 4.3 or later, the **ResourceQueryVersion** property is included in the `de.properties` file with a default value of 2. However, if you have upgraded ActiveMatrix BPM from a pre-4.3 version, the property is *not* included in the `de.properties` file by default (and algorithm version 1 is used in this situation). In the upgrade scenario, to use version 2 of the algorithm, you must add the **ResourceQueryVersion** property to `de.properties` and set the value to 2.

Reverting to Version 1 of the RQL Expression Algorithm

If you decide to revert to version 1 of the RQL expression algorithm, you must reset the RQL queries.

The RQL expression algorithm is used to evaluate all RQL queries, that is, queries that originate from Dynamic Process Participants, Performer Fields, as well as queries submitted via the ActiveMatrix BPM public API (using the **executeQuery** operation in the **ResourceQueryService**). When RQL queries originate from Dynamic Process Participants or Performer Fields, those queries are *registered* in the

database. As those expressions are in the BPM database, they must be "reset" if you change from version 2 to version 1 of the algorithm.

Resetting the queries in the database is accomplished using the `resetQueries` operation in the `ResourceQueryService`. ActiveMatrix BPM does not provide a user interface to perform this operation. Therefore, you will need to use the ActiveMatrix BPM public API to reset the queries when changing from version 2 to version 1 of the algorithm.

For information about the `resetQueries` operation, see the *TIBCO ActiveMatrix BPM Developer's Guide*.



The amount of time it takes to reset the queries in the database depends on the complexity of the expressions, and the size of the result set -- but it could be hours. While that processing is taking place, RQL result sets are based on the previous expression evaluation.

Best Practices When Using Resource Query Language (RQL)

When using Resource Query Language (RQL) there are a number of issues you should bear in mind to ensure that your processes operate most efficiently when deployed. Planning when designing your process and RQL can save problems later on.

- Make a determination about which version of the RQL expression algorithm you should be using. For more information, see [RQL Expression Evaluation](#).
- All RQL is dynamic, so there are design implications because of that. You should be aware that changes to the deployed organization model will be reflected in the RQL results set after a period of time.
- Model your system as relevant to the application - this will almost certainly not be the same as the durable organization you may see in an organization chart. If you do you will not need to use RQL in most cases.
- RQL is more complex than using basic participants. Where possible use basic participants - and only use RQL for special cases.
- In both new and existing projects, consider whether you need to use RQL at all, or whether it can be replaced more efficiently with Dynamic Organization Participants. Dynamic Organization Participants allow you to do many actions without the overhead of RQL and are quicker. Dynamic Participants only allow valid actions, and at runtime they perform better than RQL because you are explicitly specifying an internal identifier for an organization entity (GUID). So there is instant lookup, with no parsing. Using Dynamic Organization Participants is the preferred way of dynamically assigning work. See "Dynamic Organization Participants" in the *TIBCO Business Studio Concepts Guide*.
- Work items allocated using RQL do not appear in any managed work lists as they are not directly associated with specific organizational entities. You should use dynamic performer fields to get the association of work items to organizational entities functionality. See [Using a Performer Data Field or Parameter to Dynamically Define a Participant](#).
- Use RQL when you want to reference LDAP attributes or reference capabilities.
- When you deploy an organization model it is cached in memory. Actions using this in memory cache are quick (for example, RQL using references to an orgunit in the model). However, RQL that references resources will make calls to the database takes time.
- If you are looking at all LDAP attributes on a resource, looking them up in LDAP is time-consuming. If combined with operators such as OR, then it will be even slower.
- You need to use RQL for more complicated queries - those involving union, intersect, and qualifying LDAP attributes for example.
- You could produce valid RQL which does not identify any actual resources. You will not receive any warning that this will happen, so need to be careful to understand the results of the RQL you are providing.

- RQL cannot refer to Dynamic Organization Units (as these are replaced with dynamic instances at runtime).

Examples of Usage of RQL

When deciding what RQL queries to make, you should plan to minimise the number of calls they make to the database to make them efficient. For information on what options are available, see [Organization Entities](#).

```
orgunit(name='KEYTeam').position(name='AdditionalStaff') union
orgunit(name='Agency').position(name='Contractor')
```

This example queries the organization unit called KEYTeam and for positions called "AdditionalStaff". It then joins the result of this to a query of the organization unit called Agency, and the results who have the position Contractor.

```
orgunit(name='KEYTeam').position(name='Director' or name='Management').capability
(name='Language' qualifier='French' )
```

This example queries the organization unit called KEYTeam, and for positions named either Director or Management where the resource in the position has a capability of French language.

```
orgunit(name='KEYTeam').position(name='AdditionalStaff') intersect resource(attribute.Area='DACH')
```

This example will find all the resources which have an LDAP attribute of AREA that is equal to DACH. This will take some time because of the querying of LDAP for all resources defined in BPM.

```
(orgunit(name='KEYTeam').all() intersect not
orgunit(name='Management').position(name='*')) union orgunit(name='Vice
Presidents*').position(name='Permanent')
```

This is an example of a complex RQL statement that takes excessive time, as it makes multiple calls on the database to resolve resource queries.

RQL Structure

RQL expressions are made up of the following components: Keywords, Operators, Organization Entities and Combinators:

See

- [Keywords](#)
- [Operators](#)
- [Organization Entities](#)
- Combinators. See [Combining Expressions](#).

Keywords

The keywords described in this topic are permitted.

- **name**. The name of an entity. For example:
name="javaDeveloper"
- **type**. The type of the entity within the organization model schema, if it has a type. See the *TIBCO Business Studio Modeling Guide* for information about types and the schema. For example:
type="JavaProgrammer"
- **attribute**. An attribute of a resource that has been extracted from the LDAP source. This has the form **attribute.attributeName**. For example:
attribute.phone = "+44(0)1793441300"

The values associated with keywords can be qualified by combining them with other keywords, for example:

```
(name="javaDeveloper" or type="JavaEngineer")
```

- **qualifier.** The qualifier for a [Capability](#) or a [Privilege](#). For example:

```
qualifier=" > 1000"
```

The **qualifier** keyword is used by combining it with the **name** of a [Capability](#) or a [Privilege](#), for example:

```
capability(name="Language" qualifier=Arabic)
```

The '*' and '?' wildcard characters can be used:

```
name="uk-*
```

Operators

The operators listed in this topic are permitted

The operators are permitted in the following order of precedence:

- (,)
- =, <>, <, <=, >, >=
- not
- and
- or

Organization Entities

The organization entities listed can be referred to. One of these must be used as the starting point of every query.

- Organization
- Orgunit (organization unit)
- Position
- Location
- Capability
- Privilege
- Group
- Resource



A Push Destination assigned to an organization entity (Group, Position, Organization Unit, etc.) will only work when the organization entity is explicitly identified as the participant, and not when it is defined by the Resource Query Language.

The significance of specifying each of these organizational elements is set out in the following sections.

Organization

For example:

```
organization(name="EasyAs")
```

This expression returns all the resources allocated to any Position which is allocated to any Organization Unit within the Organization named **EasyAs**.

Orgunit (organization unit)

For example:

```
orgunit(name="Support-SWI")
```

This expression returns all the resources allocated to all the Positions within the organization unit named **Support-SWI**.

By default Organization Units are not treated recursively. That is, an expression specifying an organization unit does not return positions in a sub-unit of that organization unit. See [Navigating the Organization Model with RQL Queries](#) for more details.

Position

For example:

```
position(name="Manager")
```

This expression would return all the resources in a Position named **Manager**.

You can also locate resources based on their type in the organization model schema, for example:

```
position(type="UnitManager" or name="Manager")
```

This expression would return all the resources in an position which is an instance of the position type named **UnitManager**, or is a Position called **Manager**.

Location

For example:

```
location(name="NewYork")
```

This expression returns all the resources that:

- Have a location of **NewYork**, or
- Are allocated to a Position that:
 - is located in **NewYork**, or
 - is allocated to an Organization Unit that:
 - is located in **NewYork**, or
 - is allocated to an Organization that is located in **NewYork**.

Capability

For example:

```
capability(name="JavaProgrammer")
```

This expression includes all the resources that:

- Have the capability named **JavaProgrammer**



RQL only makes use of capabilities that are assigned to Resources.

Capabilities may be assigned to Positions and Groups in the organization model. These represent "entry criteria" - that is, only resources with that capability should be assigned to that group or position. However, this is not enforced. Capabilities assigned to Resources indicate that the person represented by that Resource does actually have the capability in question.

Capabilities may be further refined by using a qualifier, as described for [Privilege](#).

Privilege

For example:

```
privilege(name="signoff" qualifier>10000)
```

All the resources allocated to a Position with:

- the privilege **signoff** with a qualifier value greater than 10000, or
- belonging to an Organization Unit with the privilege **signoff** with qualifier value greater than 10000.

An expression can also specify the privilege without specifying a qualifier. For example:

```
privilege(name="signoff")
```

This would return all the resources allocated to a Position or an Organization Unit with the privilege **signoff**, regardless of any qualifier.

Group

For example:

```
group(name="Health&Safety")
```

This expression returns all the resources in a group named **Health&Safety**.

Groups can operate recursively - that is, groups can contain groups. So this expression would also include all resources belonging to all sub-groups. This can be overridden - see [Navigating the Organization Model with RQL Queries](#).

Resource

For example:

```
resource(name="Clint Hill")
```

This expression returns all the resources with the name **Clint Hill**.

Unlike the other entities listed, Resources refer directly to actual users as defined in LDAP. When selecting resources you can start with a particular resource by referencing its name, for example:

```
resource(name="Clint Hill").position(name="abc").orgunit(type="efg").privilege(name="signoff"
qualifier > 10000)
```

This expression returns all Resources that belong to an OrgUnit of type "efg" in which Clint Hill holds the position named "abc", and in which he holds the "signoff" privilege, qualified for a value of greater than 10000.

You can also query attributes from the LDAP database that have been passed to BPM, for example:

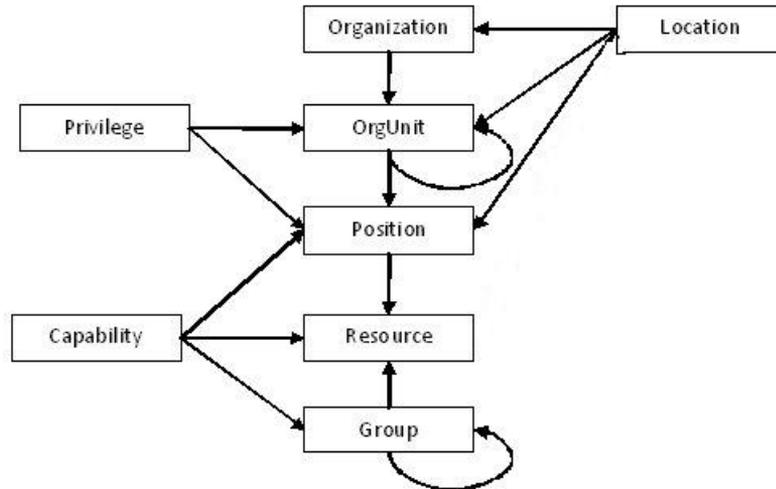
```
resource(attribute.phone="+44(0)1793*" and attribute.language="*Spanish*")
```

This returns the resources with a Swindon dialing code (+44 (0)1793) on their **phone** attribute and whose language attribute includes the text **Spanish**.

Navigating the Organization Model with RQL Queries

RQL expressions, starting with one of the elements listed in [Organization Entities](#), locate resources by following the paths indicated by arrows in the diagram below. For example, an expression that specifies a capability will find all the positions, resources and groups with that capability; and then these will cascade down to find the resources in the positions and groups.

The following diagram shows how resources can be located from each of these starting points.



Using the . Operator

You can restrict the results returned by an expression by using the dot operator ("."). This can be used in two ways.

Using the dot operator to qualify organization entities

You can use the dot operator to specify a qualification of an organizational entity. For example:

```
orgunit(type="Sales").privilege(name='signoff' qualifier>10000)
```

This expression returns resources in those Organization Units of type **Sales** that have the privilege qualified by a value of greater than 10000.



This is not a simple "and" operator; the expression does not return resources that are in organization units of type **Sales** and that have the required privilege; it returns resources where the organization unit itself has that privilege.

The dot operator is not commutative: in other words, changing the order of the expression changes its meaning.

So, for example:

```
position(name='abc').orgunit(type='Sales')
```

Expressed in natural language, this example means:

All Resources, in all Positions of Org-Unit of type "Sales", in which the Position named "abc" .

Another example is:

```
orgunit(type="Sales").position(name="abc")
```

Expressed in natural language, this example means:

All Resources in the Position named "abc" of Org-Unit of type "Sales".

Using the dot operator in hierarchical relationships

The dot operator works differently when used to express hierarchical relationships between two organizational entities. For example:

```
orgunit(type="area office").position(type="manager")
```

returns Positions of type **manager** that are in organization units of type **area office**.



You can only use the dot operator to link organizational entities that are **directly** connected by arrows in the diagram in [Navigating the Organization Model with RQL Queries](#) . For example:

```
organization(type="public company").position(type="manager")
```

would not be valid, because **organization** and **position** are not directly linked.

Using only children and all

In organization models, both Organization Units and Groups can be linked to other elements of the same type in a hierarchy.

See the *TIBCO Business Studio Modeling User's Guide* for more details about this. However RQL expressions do not by default navigate this hierarchy.

For example the expressions:

```
orgunit(name="AreaNorth")
```

```
group(name="HomeInsurance")
```

would return only resources assigned directly to the named organization unit or group, not to other organization units or groups in a hierarchical relationship below it.

To override this default behavior, you can use one of the following modifiers:

- **only()** . For example

```
group(name="HomeInsurance").only()
```

This returns only resources assigned to the named element. (This is the default behavior both for Organization Units and for groups.)

- **children()**. For example

```
group(name="HomeInsurance").children()
```

This returns resources assigned to the named element and to any first level child elements, but not to elements any further below in a hierarchy.

- **all()**. For example

```
orgunit(name="AreaNorth").all()
```

This returns resources assigned to the named element and to any subordinate elements.

Combining Expressions

All expressions in RQL return the set of resources that correspond to that expression. You can combine expressions to produce more sophisticated queries using the following set operators:

- **union**. Returns resources that are in either expression. For example:

```
orgunit(name="Drivers").position(name="Developer") union  
orgunit(name="OsakaDev").position(name="JavaDeveloper")
```

returns any resource who is either a Developer in the Drivers organization unit or a Java Developer in the OsakaDev organization unit.

- **intersect**. Returns resources that are in both expressions. For example:

```
orgunit(name="Drivers").position(name="Developer") intersect group(name="JavaDevelopers")
```

returns any resource who is both a Developer in the Drivers organization unit and a member of the Java Developers Group.

Use an intersect to specify multiple qualifier values for a capability or a privilege. (If the data type of the qualifier is EnumSet it is possible for the capability or privilege to have multiple qualifier values.) For example:

capability(name='Language' qualifier='French') intersect capability(name='Language' qualifier='Spanish')

- **subtract.** Returns resources that are in the first expression but not the second. For example `orgunit(name="Drivers").position(name="Developer") subtract group(name="Managers")` returns any resource who is a Developer in the Drivers organization unit but is not a member of the Managers Group.
- **not.** Returns resources that are in the first expression but not the second. For example `orgunit(name="Drivers").position(name="Developer") intersect not group(name="Managers")` returns any resource who is a Developer in the Drivers organization unit but is not a member of the Managers Group.



While similar in the resources they return, **subtract** will return the *relative complement* and should be used when performance is a consideration.

Another example might be:

```
(
orgunit(type="Support").position(type="SupportEngineer")
intersect
not group(name="Backdesk")
)
union
orgunit(name="Support").position(name="Manager")
```

This selects support engineers in the Support department who are not part of the back desk group, together with the support department's managers.

Combining expressions in this way can sometimes mean that a resource is included in the results for more than one reason. If this happens, the resource will only be included in the result set once; that is, there is no duplication of results.

RQL Cleanup Configuration

When resource queries are no longer referenced, they should be deleted from the system.

The `de.properties` file contains the following resource query cleanup properties that can be used to configure when and how often resource queries are evaluated to determine which can be deleted:

- `ResourceQueryCleanerEnable` - Enables, or disables, the evaluation of resource queries to identify those that are no longer referenced. If this is disabled (false), automatic deletion of un-referenced resource queries does not take place.
- `ResourceQueryCleanerStart` - The time of day at which resource queries, that are no longer being used, are scheduled for deletion.

Cleanup of unused resource queries is invoked at this time each day that cleanup is scheduled according to the value of `ResourceQueryCleanerInterval`.

- `ResourceQueryCleanerInterval` - The interval between evaluations of resource queries to identify those that are no longer referenced, and can be deleted. This defaults to once per day (at the time specified in `ResourceQueryCleanerStart`).
- `ResourceQueryCleanerEnd` - The time of day at which the last resource query evaluation of the queries that can be deleted will be accepted.
- `ResourceQueryCleanerLimit` - The number of resource queries checked for deletion in a single database transaction.

- `ResourceQueryCleanerPause` - The number of seconds between deleting batches of resource queries.

For additional information about these properties, see "Configuration of the TIBCO ActiveMatrix BPM Directory Engine" in the *TIBCO ActiveMatrix BPM Administration*.