



TIBCO ActiveMatrix BusinessWorks™

Application Development

*Software Release 6.6
November 2019*

Document Updated: January 2020, February 2020, March 2020

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, the TIBCO logo, the TIBCO O logo, TIBCO ActiveMatrix BusinessWorks, TIBCO Rendezvous, TIBCO Enterprise Message Service, TIBCO Business Studio for BusinessWorks, TIBCO Enterprise Administrator, TIBCO ActiveSpaces, TIBCO Runtime Agent, and TIBCO Designer are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

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

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 © 2001-2020. TIBCO Software Inc. All Rights Reserved.

Contents

Figures	8
TIBCO Documentation and Support Services	9
Changing Help Preferences	10
Application Development Overview	11
Application Design Considerations	13
Process Design Considerations	16
Service Design Considerations	19
Memory Saving Considerations	20
TIBCO Business Studio™ for BusinessWorks™ Essentials	22
Outline	24
Module	24
File Explorer	25
API Explorer	25
Process Editor	28
Palette Library	28
Entity Naming Conventions	29
Importing an Existing Project into Workspace	30
Developing a Basic Process	32
Creating an Application Module	32
Creating a Shared Module	33
Reconfiguring Deployment Target	34
Generating the manifest.json File Using the bwdesign Utility	35
Generating the manifest.yml file	36
Exporting a Shared Module as a Binary Shared Module	36
TIBCO Business Studio for BusinessWorks	36
CLI	37
Using Binary Shared Modules in your Project	38
Referencing Shared Modules	41
Creating a Process	42
Working with Process Properties	43
Creating a Subprocess	49
Creating an Activator Process	51
Adding Activities	54
Working with Transitions	55
Working with Standard Activity Features	56
Input and Output	57

Creating a Module Property	61
Editing a Module Property	62
Promoting Module Properties for Visibility at the Application Level	62
Deleting a Promoted Property	63
Importing WSDLs	63
Using Additional Features	64
Using Scopes	64
Adding Scope Variables	64
Defining and Using Shared Variables	67
Retrieving and Assigning a Value of a Shared Variable	69
Working with Critical Section Groups	69
Using Fault Handlers	70
Using Conversations	71
Using Checkpoints	72
Using Coercions	72
Adding a Single Coercion	73
Adding Multiple Coercions	74
Coercing a Specific Data Type	74
Editing Coercions	75
Removing Coercions	75
Configuring Database for the Engine	76
Configuring the Engine for Group Persistence Mode	78
Configuring EMS as the Group Provider for Engine	78
Configuring TIBCO FTL® as the Group Provider for Engine	80
Creating Process Diagrams Explicitly	82
Displaying Individual Element Mappings	83
Removing Groups	84
Configuring the Ungroup Preferences	84
Ungrouping a Local Transaction Group	84
Ungrouping Groups with Scopes	85
Overview of Policies	87
Managing Policy Resources	88
Creating a Folder for Policies	88
Creating an Authentication Resource	88
Associating Policies	89
Removing a Policy	89
HTTP Security	91
Enforcing Basic Authentication	91
Enforcing Basic Credential Mapping	93

SOAP Security	97
Enforcing WSS Consumer	97
Enforcing WSS Provider	100
Building Projects Automatically	104
XPath	105
XPath Basics	105
XPath Expression	107
XPath Builder	110
Developing a SOAP Service	113
Consuming SOAP Services	116
Developing a RESTful Service	118
Implementing a REST Service Provider	118
Discovering API Models from TIBCO Business Studio™ for BusinessWorks™	120
Importing an API Model into your Workspace	121
Creating an XML Schema for a Swagger 2.0 File Imported in TIBCO Business Studio™ for BusinessWorks™	123
Synchronizing the Imported REST API Models in TIBCO Business Studio™ for BusinessWorks™	124
Developing Java Applications	125
Using a Simple Java Invoke Activity	125
Accessing Module Properties from Java Global Instance	126
Accessing Module Properties from Java Invoke Activity	126
Accessing Module Properties in User-Defined Java Code Referenced in JavaProcessStarter	127
Creating an Application	128
Working with Application Properties	129
Creating an Application Property	129
Exporting an Application Profile	130
Tokenizing Application Properties for exporting in the Properties file	130
Importing an Application Profile	133
Generating Deployment Artifacts	135
Deploying an Application	138
Refactoring a Shared Resource or Policy Package	141
Renaming a Resource or a Policy Package	141
Changing the Location of a Resource or a Policy	141
Working with Multiple Component Processes	142
Adding Multiple Component Processes	142
Deleting Multiple Component Processes	142
Enabling Auto Start of Component Process	143
Analyzing Dependencies and References	144
Unused Resources	146

Repairing TIBCO ActiveMatrix BusinessWorks™ Projects	150
Using the Debugger	152
Configuring the Debugger	153
Testing an Application in TIBCO Business Studio™ for BusinessWorks™	155
Remote Debugging	155
Unit Testing	158
Running Test Assertions	158
Using Demo Projects	158
Adding Unit Test Assertions	160
Running Maven from Command Line	169
Unit Test Reports and Test Coverage Reports	170
Limitations for Unit Test Assertions	171
Running Activity Assertions	171
Adding Mocking Support for Activities	173
Running Unit Tests in TIBCO Business Studio™ for BusinessWorks™	175
Generating Mock Output File	176
Limitations for Mock Support	178
Collaborative Application Development	179
Configuring TIBCO Business Studio™ for BusinessWorks™ with Git	179
Generating gitignore Files	179
Generating gitignore Files at Application Module Level	181
Synchronizing Module Properties	182
Using the bwdesign Utility	184
Best Practices	190
Troubleshooting	193
Mapping and Transforming Data	193

Figures

Approaches to Application Development	12
TIBCO Business Studio for BusinessWorks Workbench	23
Parent Process	51
Sub Process	51
Drag-and-Drop a Resource	54
Drag-and-Drop a Resource	55
Input Tab	58
Input Editor Tab	58
Output Tab	61
Output Editor Tab	61
Fault Handler Attached to an Inner Scope	70
Schema Elements in Data Source	106
XPath Builder	110

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_BW_<version>_docinfo.html
```

Access the following *TIBCO ActiveMatrix BusinessWorks*[™] guides on the TIBCO Documentation site:

- Concepts
- Installation
- Getting Started
- Application Development
- Administration
- Bindings and Palettes Reference
- Business Works Samples
- Error Codes
- Migration
- Performance Benchmarking and Tuning
- REST Reference Guide

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>.

Changing Help Preferences

By default, documentation access from TIBCO Business Studio™ for BusinessWorks™ is online, through the TIBCO Product Documentation site that contains the latest version of the documentation. Check the website frequently for updates. To access the product documentation offline, download the documentation to a local directory or an internal web server and then change the help preferences in TIBCO Business Studio for BusinessWorks.

Prerequisites

Before changing the help preferences to access documentation locally or from an internal web server, download the documentation from <https://docs.tibco.com/>.

1. Go to <https://docs.tibco.com/>
2. In the **Search** field, enter and press **Enter**.
3. Select the TIBCO ActiveMatrix BusinessWorks™ product from the list. This opens the product documentation page for the latest version.
4. Click **Download All**.
5. A zip file containing the latest documentation downloads to your web browser's default download location. Copy the zip file to a local directory or to an internal web server and unzip the file.

To point to a custom location:

Procedure

1. In TIBCO Business Studio for BusinessWorks, click **Window > Preferences**. On Mac OS X, click **TIBCO Business Studio > Preferences**.
2. In the Preferences dialog, click **BusinessWorks > Help**.
3. Click **Custom Location**. Then, **Browse** and select the `html` directory in the folder where you extracted the documentation, or provide the URL to the `html` directory on your internal web server.
4. Click **Apply**, and then click **OK**.

Application Development Overview

TIBCO ActiveMatrix BusinessWorks™ applications can be developed using either the traditional phases of waterfall development, or using an incremental and iterative approach such as Scrum.

The *TIBCO ActiveMatrix BusinessWorks™ Application Development* guide explains the following:

- Approaches to application development.
- Considerations to be made when building an application.
- Information on how to work with various software components and how to generate the deployment artifact.

Application development consists of the following phases:

- **Analysis** - Analyze the business problem and identify the applications, modules, services, and processes that need to be created to solve the problem.
- **Application Creation/Design** - Create one or more applications identified during the analysis phase. TIBCO Business Studio™ for BusinessWorks™ provides the design-time environment to design an application and its components that implement the business logic.
- **Service Design** - Create the services identified in the analysis phase. The services can be accessed by processes that are used to implement the business logic.
- **Process Design** - Create the processes that implement the business logic. These processes can access the services configured.
- **Generating Deployment Artifacts** - Create a deployment artifact — an archive file, after creating and configuring the processes and services.



If any changes to the design or configurations are made, the archive file must be regenerated.

There are two main approaches to application development: top-down and bottom-up.

Top-down is a design approach that begins with a holistic view of the application, specifying the major functions or interfaces it will need before the next level of details. This process is repeated until the most granular pieces are designed and implemented. The application is then ready for testing. Top-level services and processes can be designed and developed first before moving to the lower levels.

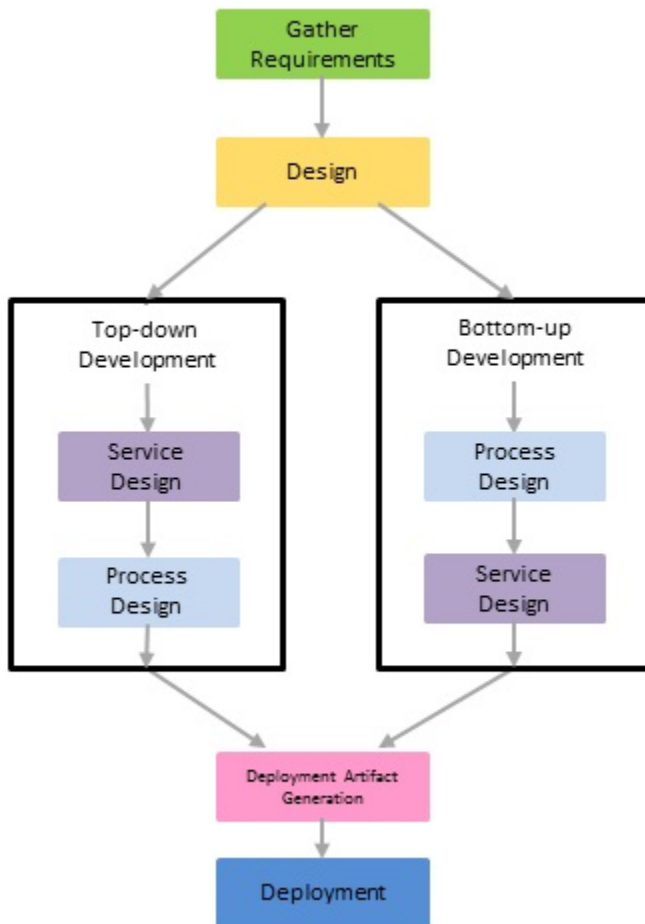
In the bottom-up approach, the basic elements of the application are first specified and developed as building blocks. These reusable parts are then connected to form functional units that serve a higher purpose. This process is repeated until the design grows in complexity and completeness to form an application. The building blocks can be created as layers of services, subprocesses, and shared resources. These building blocks are assembled together to form application modules or shared modules. These modules are then assembled together to form an application.

In practice, even a new application can have existing services to leverage from. As a result, a problem can be approached from both top and bottom, resulting in a hybrid approach. The bottom part can start creating reusable shared modules to encapsulate existing system services that are well defined. The top part can start with the business requirements and break it down to intermediate layers, until the reusable modules are reached.

Either top-down or bottom-up approaches can be used with service-driven or process-driven design patterns. Service-driven means the service contract or interface of each functional component is formalized first. The processing logic behind the service simply becomes an implementation detail that is encapsulated. This is where these SOA design principles can be followed: standardized service contract, loose coupling, service abstraction, service reusability, service statelessness, and service composability.

Process-driven means the business processes or integration flows are first realized and captured. Service contracts may or may not be applicable in a process-centric application, especially for batch or EAI-type automation scenarios.

Approaches to Application Development



Each of these approaches can be followed in conjunction with the waterfall or Scrum development methods.

The generation of the deployment artifact indicates that the application can be deployed to the run time. Any further changes to the design-time configurations require the deployment artifact to be regenerated. For deployment and administration details, see *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

Application Design Considerations

Applications help solve integration problems of varying complexity. This section describes some important factors to consider when designing an application.

Choosing Between Integration Styles

The table, [Salient features of integration styles](#), provides guidelines to choose a high-level integration style for your applications.

Salient features of integration styles

	Speed of Integration	Data Abstraction	Richness of Orchestration Primitives	Typical Endpoints
Batch-oriented	Non-real time	Record	Low	Databases, files, and so on
Application-oriented	Real-time	Message	Medium	Application APIs, Adapters, and so on
Service-oriented	Real-time	Service, Operation	High	Web services and APIs
Resource-oriented	Real-time	Resource	Medium	Mobile/Web Applications and APIs

In an application-oriented integration style, each operation in a process can be invoked by a call to the process. Invoking multiple operations requires multiple calls to the process, that are then executed sequentially.

A service-oriented style exposes multiple operations available in a process and each of the operations can be called directly. These operations are not related and can be executed independently. However, you can use conversations to correlate the related messages between two or more parties.

Choosing the Modularity

An application module is the smallest unit of resources that is named, versioned, and packaged as part of an application, and then executed in the ActiveMatrix BusinessWorks runtime environment. It cannot provide capabilities to other modules.

A shared module is the smallest unit of resources that is named, versioned, and packaged as part of an application and can be used by other modules that are part of the same application. Shared modules export their functionality (process, shared resources, and schema namespaces) to application modules or to other shared modules. When creating a new module, select a shared module if the business logic needs to be shared across multiple applications. Shared modules can also be used if XML collisions exist.

Differences between Application and Shared Modules

	Runtime	Reusability	Encapsulation	XML Namespace Restrictions
Application Modules	Can be executed by the run time when packaged as part of an application.	Can be used by one or more applications.	Processes within an application module are visible to each other. However, the processes are not visible outside of the module.	Namespace can be provided by multiple documents.
Shared Modules	Cannot be executed by the run time unless utilized by an application module.	Can be used by one or more application modules or shared modules.	Processes within a shared module are visible to each other. However, only the processes defined as public are visible outside of the shared module.	Only one document can provide the namespace; that is two documents cannot have the same namespace. All schemas and WSDL files are visible to other modules that depend on the shared module.

Choosing Implementation Technologies for the Modules

When implementing the business logic, ActiveMatrix BusinessWorks provides flexibility ranging from developing applications graphically without coding, to using existing Java classes (or libraries), to writing custom code. Application modules or shared modules typically consist of one or more business processes that define the business logic. Create an application or shared module using the GUI to leverage the rich orchestration capabilities provided by ActiveMatrix BusinessWorks.

Choose to create (or use) a Java module (or a Java OSGi bundle), if multiple calls from a process to other Java libraries are needed to compute the result. Java modules provide a high degree of customization. To use the enhanced Java development tooling such as source folders, JRE libraries, and so on, select the **Use Java Configuration** check box in TIBCO Business Studio for BusinessWorks when creating an application module. Alternatively, create a module that contains existing Java code or custom code.

Differences between Process Modules and Java Modules

	Orchestration Capabilities	Visibility	Granularity	Examples
Process Modules	High	High visibility of process flow logic, services, and bindings.	Better suited for coarse-grained functionality that consists of more discrete functionality and process constructs.	Account opening, mortgage loan, and so on.
Java Modules	Low	Low	Better suited for fine-grained functionality that has a very specific function, and often requires very little or no process constructs.	Query flight status, update product description, and so on.

Process Design Considerations

In process-driven design, the business processes or integration flows are first realized and captured. Service contracts might be applicable in a process-centric application, especially for batch or EAI-type automation scenarios. This topic describes some important factors to be considered when using a process-driven approach.

Choosing Between Stateful and Stateless Processes

Stateful processes maintain the state across multiple operations. They are better suited when you need the server to maintain the state across operations. For processes that involve related message exchanges between the same or different consumers, conversations can be used to maintain state across operations.

Stateless processes do not maintain state. They are better suited when you need to process higher loads of requests as each operation is executed independently. They do not require correlation or conversations between multiple operations in a process, thus allowing the server to process each operation without maintaining any state information. The client can choose to maintain the state information, if needed.

Process	Maintains State	Data Sharing	Conversations
Stateful Processes	Across multiple operations and interfaces.	Data can be shared by activities across operations that are executing as part of the same job.	Uses conversations to enable correlation.
Stateless Processes	Does not maintain state.	Data is not shared.	No conversations.

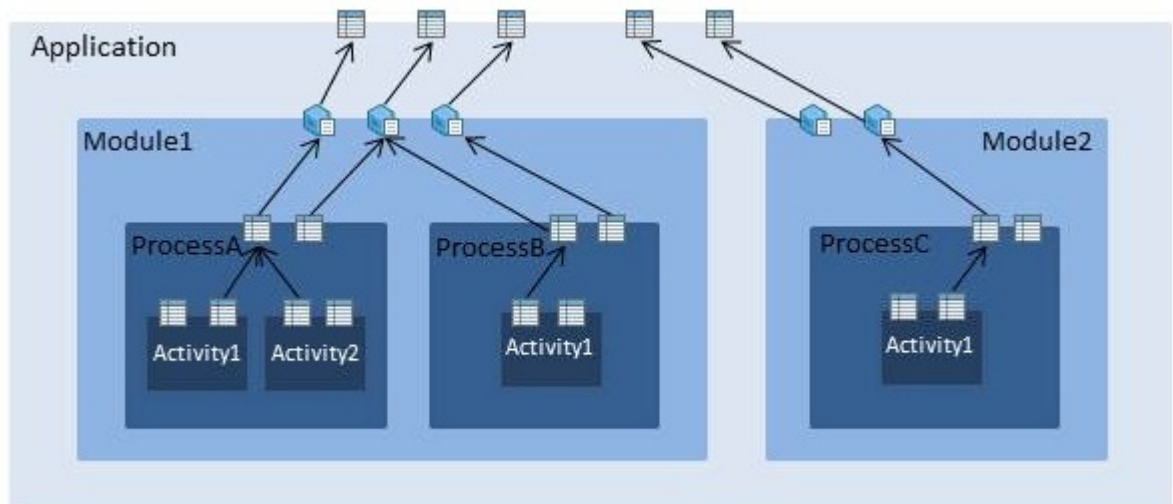
Choosing Between Properties and Variables

Properties are used to save configuration data at different levels. They can be classified into application properties, module properties, and process properties. For more information, see [Choosing Between Process Properties, Module Properties, Shared Module Properties, and Application Properties](#).

Variables are used to save the state at different levels. They can be classified into process variables, scope variables, and shared variables. For more information, see [Choosing Between Process Variables, Scope Variables, and Shared Variables](#).

Choosing Between Process Properties, Module Properties, Shared Module Properties, and Application Properties

Properties can be classified into application properties, module properties, shared module properties, and process properties. Properties follow the layered configuration model where configuration is pushed from top to the bottom as seen in the illustration:



Properties defined in the inner layer can reference a property defined at the parent layer. For example, a process property can reference a module property instead of providing a literal value. Public properties are visible to the encapsulating layers.

Choosing the right level ensures an easier to maintain list of properties in your application and keeps the number of properties at the application level to a minimum.

Comparing Process, Module, Shared Module and Application Properties

Property	Scope/Visibility	Datatype	Values	Additional Information
Process Properties	Visible within a process.	Literal or shared resource reference.	Literal, shared resource reference, or a module property reference.	Literal values cannot be modified at the module or application level.
Module Properties	<ul style="list-style-type: none"> Visible within the module. Private module properties cannot be viewed from the Admin UI. Not visible or changeable from Administrator. 	Literal or shared resource reference.	<ul style="list-style-type: none"> Literal or a shared resource reference. Private module property values cannot be edited from the Admin UI. 	Cannot be assigned to an activity directly. You need to reference a module property from a process property, and then reference the process property from the activity.

Property	Scope/Visibility	Datatype	Values	Additional Information
Shared Module Properties	<ul style="list-style-type: none"> • Visible within the module. • Visible within projects that contain dependencies to the Shared Module that the Shared Module Property came from. • Private module properties cannot be viewed from the Admin UI. • Not visible or changeable from the Admin UI. 	Literal or a shared resource reference.	<ul style="list-style-type: none"> • Literal or a shared resource reference. • Private module property values cannot be edited from the Admin UI. 	<ul style="list-style-type: none"> • Shared Module Properties are module properties that come from a Shared Module. • Cannot be assigned to an activity directly. You need to reference a module property from a process property, and then reference the process property from the activity. • Can be used for activities, process properties, shared resources, and SOAP Bindings.
Application Properties	Displays all the module properties in the application. These properties are visible in Administrator.	Literal.	<ul style="list-style-type: none"> • Literal. • Profiles can be used to provide a new set of values for the application. 	Overrides module properties, thus enabling you to use different values for the same module.

Choosing Between Process Variables, Scope Variables, and Shared Variables

A process variable saves the state at the process level and a scope variable saves the state within the scope.

Variables defined within a scope are visible only within the scope. If the scope variable name is the same as a process variable name, then the scope variable takes precedence over the process variable within the scope.

Shared variables are used to save the state. There are two types of shared variables:

- **Module shared variable** - saves the state at a module level.
- **Job shared variable** - saves the state for the duration of a job.

For more information on sharing variables, see [Using Shared Variables](#) topic and the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide.

Handling Exceptions

Errors can occur when executing a process. The potential runtime errors in your process can be handled in one of the following ways:

- **Catch Specific:** Used to catch a specific kind of fault at either activity, scope, or process levels.
- **Catch All:** Used to catch all error or faults thrown at the selected level.



You can add an error transition to an activity or a group to specify the transition to take in case of an error.

Service Design Considerations

In service-driven design, the service contract or interface of each functional component is formalized first. The processing logic behind the service simply becomes an implementation detail that is encapsulated. This section describes some important factors to consider when using the service-driven approach.

Choosing Between Abstract Process Starters, Services, and Service Subprocesses

Choose a process starter activity to start a process when an event occurs. There can be only one process starter in a process.



Do not create a process with a technology specific process starter such as an HTTP or JMS process starter.

Choose a service if you want to expose the operations available in a process outside the application module.

Choose a service subprocess to make your business process easier to understand and debug. A subprocess is invoked by a parent process and the output of the subprocess is used in the main process. A parent process calls a subprocess in two ways: in-line and non-in-line. At run time, an in-line subprocess executes as part of the parent process' job, while a non-in-line subprocess spawns a new job.

Choosing between REST and SOAP Bindings

A process service is exposed to external consumers by configuring bindings such as REST or SOAP.

Service	Data Abstraction	State Information	Overhead of Additional Parameters (Headers or other SOAP elements)
REST Services	Resources	Stateless	Less

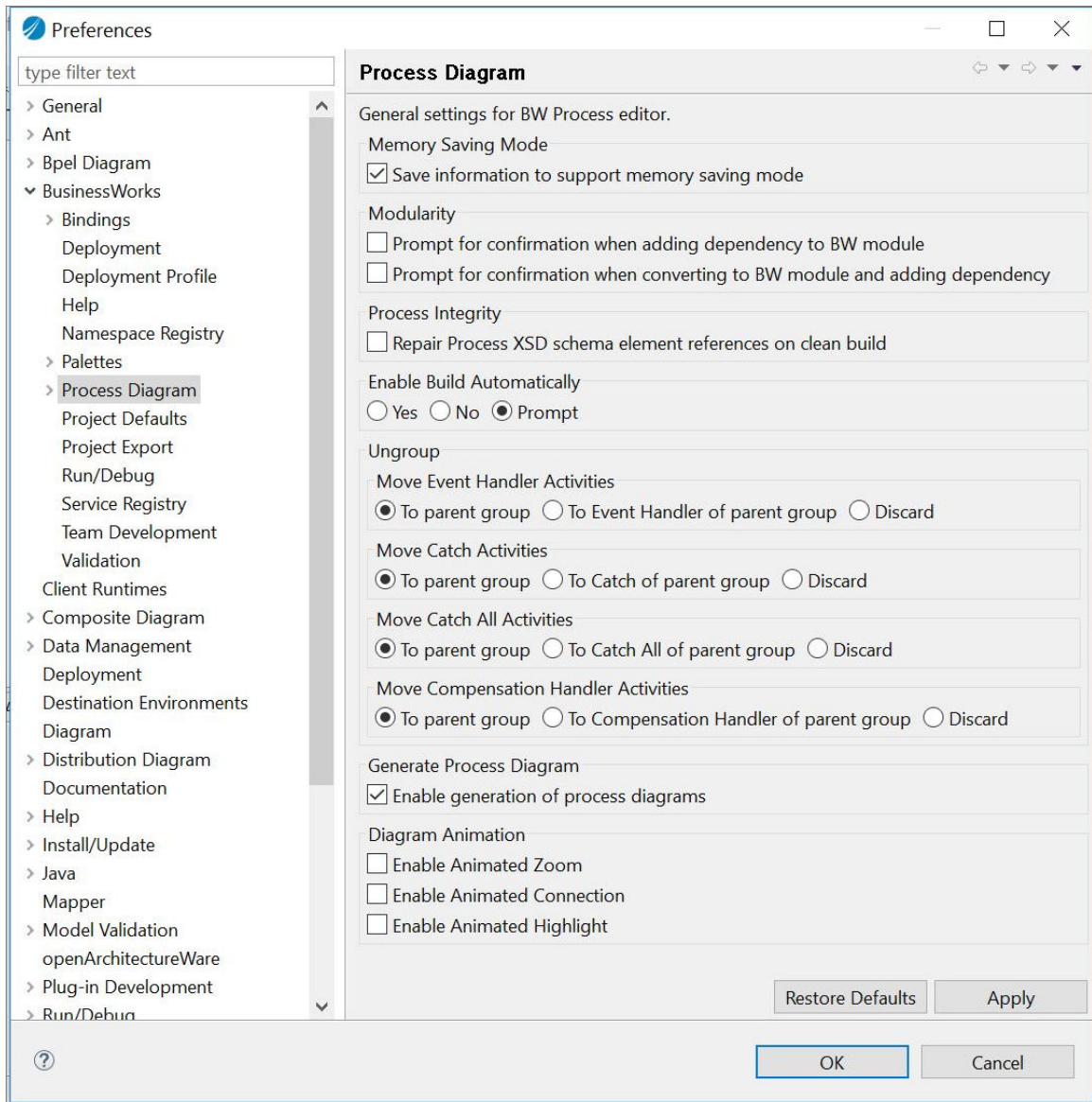
Service	Data Abstraction	State Information	Overhead of Additional Parameters (Headers or other SOAP elements)
SOAP Services	Operations	Stateful	High

You can use multiple Web Service Definition Language (WSDL) files with an identical target namespace in a shared module, and an application module.

Memory Saving Considerations

This is to outline the variables which are not used for a specific activity so that the items corresponding to the variables are freed (set to null) after an activity is executed at the run time. When Memory Saving Mode option is selected, the memory saving variables are calculated and an activity frees up the unused variables at the run time. In case of new projects, you can select the **Save information to support memory saving mode check box** available at **Window > Preferences > BusinessWorks > Process Diagram** in the Memory Saving Mode section.

For the process that already has the memory saving variables, the memory saving variables must be re-calculated when that process is saved to keep all the memory saving variables in sync with the usage of the variables in that process.



To know more about the memory saving feature for existing projects, see "Enabling Memory Saving Mode at Design Time" in the *TIBCO ActiveMatrix BusinessWorks™ Performance Benchmarking and Tuning* guide.

TIBCO Business Studio™ for BusinessWorks™ Essentials

TIBCO Business Studio for BusinessWorks is an Eclipse-based integration development environment that is used to design and test TIBCO ActiveMatrix BusinessWorks™ applications.



If you are familiar with the TIBCO Business Studio for BusinessWorks user interface, skip to the section [Developing a Basic Process](#).

Using TIBCO Business Studio for BusinessWorks, designers implement an executable application that can be deployed to ActiveMatrix BusinessWorks.

Starting TIBCO Business Studio for BusinessWorks

To start TIBCO Business Studio for BusinessWorks on Windows, select **Start > All Apps > TIBCO_HOME > Studio for Designers**. On Linux or Mac OS, select the TIBCO Business Studio for BusinessWorks executable located at `<TIBCO_HOME>/studio/<version>/eclipse/`.

On the Workspace Launcher dialog, accept the default workspace or browse to create a new workspace, and then click **OK**. TIBCO Business Studio for BusinessWorks starts and the default development environment, called a *workbench*, is displayed. A welcome screen is displayed in the window when a workspace is opened for the first time. For more information TIBCO Business Studio for BusinessWorks, see the section *Accessing Samples* in the *ActiveMatrix BusinessWorks™ Samples* guide.

On Mac OS, TIBCO Business Studio for BusinessWorks displays the Subversion Native Library Not Available dialog box if the SVN interface is set to JavaHL (default) and the JavaHL libraries are not available. To ensure that the dialog box is not displayed each time you start TIBCO Business Studio for BusinessWorks, perform one of the following:

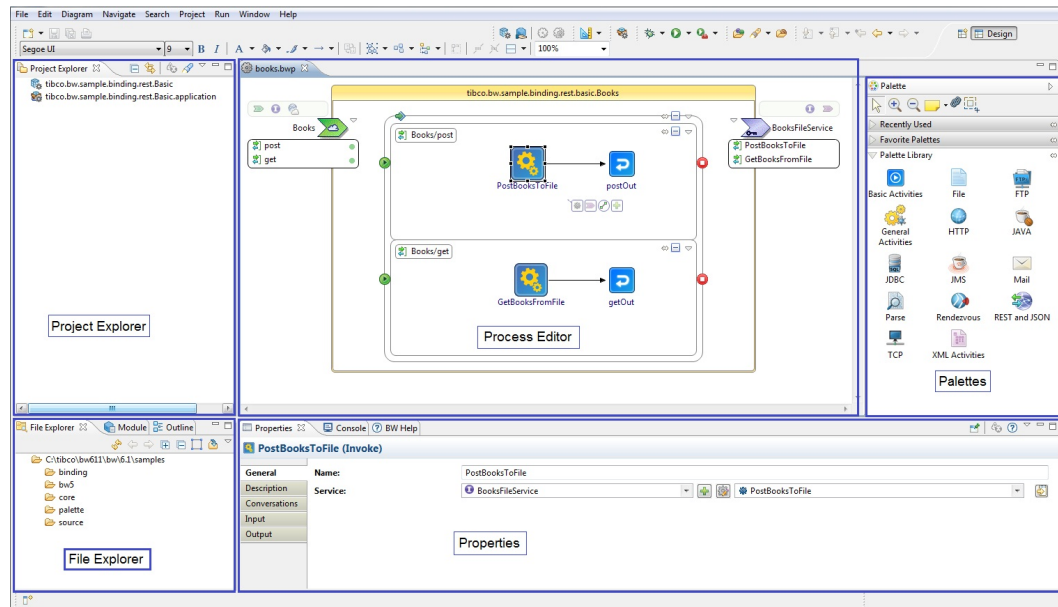


- Install the JavaHL libraries. See <http://subclipse.tigris.org/wiki/JavaHL> for instructions.
- Update the SVN interface to use SVNKit instead of JavaHL. Select **Window > Preferences** and in the Preferences dialog box, select **Team > SVN**. For the **SVN interface Client** field, select `SVNKit (Pure Java)` interface from the drop-down list.









TIBCO Business Studio for BusinessWorks Development Environment

TIBCO Business Studio for BusinessWorks provides a workbench that is used to create, manage, and navigate resources in your Eclipse workspace. A *workspace* is the location on your machine where the artifacts related to your ActiveMatrix BusinessWorks™ projects are stored.

TIBCO Business Studio for BusinessWorks Workbench



The TIBCO Business Studio for BusinessWorks workbench has features such as:

- **Menu:** Contains menu items such as File, Edit, Navigate, Search, Project, Run, Window, and Help.
- **Toolbar:** Contains buttons for the frequently used commands such as:
 - New 
 - Save 
 - Enable/Disable Business Studio Capabilities 
 - Create a new BusinessWorks Application Module 
 - Create a new BusinessWorks Shared Module 
 - Debug 
 - Run 
- **Perspectives:** Contain an initial set and layout of views that are needed to perform a certain task. TIBCO Business Studio for BusinessWorks launches the Design perspective by default. Use the Design perspective when designing a process and the Debug perspective when testing and debugging a process. To change the perspective, select **Window > Open Perspective > *perspective_name*** from the main menu. Or, you can click the icon  at the top right-hand side of the workbench and select the perspective to open.
- **Views:** Display resources and allow for navigation in the workbench. For example, the Project Explorer view displays the ActiveMatrix BusinessWorks applications, modules, and other resources in your workspace, and the Properties view displays the properties for the selected resource. To open a view, select **Window > Show View > *view_name*** from the main menu.
- **Editors:** Provide a canvas to configure, edit, or browse a resource. Double-click a resource in a view to open the appropriate editor for the selected resource. For example, double-click on a process (`MortgageAppConsumer.bwp`) in the Project Explorer view to open the process in the editor.

Explorers

The TIBCO Business Studio for BusinessWorks consists of the following tabs in the left pane:


- Project Explorer
- API Explorer
- File Explorer
- Outline tab
- Module tab
- Deployment Servers


Creating an Application and Designing a Process

Create an application and design one or more process(es) within the application in TIBCO Business Studio for BusinessWorks to implement the business logic. See [Developing a Basic Process](#).

Testing and Debugging an Application

Using TIBCO Business Studio for BusinessWorks you can test and debug your application from the design-time.

To run the selected application, select **Run > Run** from the main menu, or click  on the toolbar.

To execute and debug the application, select **Run > Debug** from the main menu, or click  on the toolbar.

By default, the project displayed in the Process Editor launches. You can run or debug an application using a specific configuration. Create one or more configurations for your application by selecting **Run > Run Configurations** from the main menu and specifying the following:

- Bundles to be executed.
- Program arguments such as the target operating system, target architecture, and VM arguments.
- Settings that define the Java Runtime Environment including the Java executable, runtime JRE, configuration area, and so on.
- Tracing criteria for the OSGi JAR file, if needed.
- Common options such as saving the results either as local files or as shared files, displaying them in the menus (Debug and/or Run), and defining encoding for the result files.

Outline

The Outline view displays the details of a currently selected process or artifact in a tree like structure. It shows a more in-depth view of the selected artifact as compared to the Project Explorer.

Use this view while you are actively editing a process to select an artifact and see its properties right away.

Module

The Module tab displays the module properties and shared variables for the selected module. It displays the variables in the module.


Click the module name in the Project Explorer to view the default values of the module properties defined in the module and/or the shared variables that exist in the module. This tab is useful as it saves to the additional step of having to open the Module Descriptor editor to view the default values of the

selected module. You can also drag and drop a shared variable from the Module tab into a process that is open in the Process Editor.

File Explorer

The File Explorer displays a view of selected folders in your local file system.

By default, the File Explorer displays the **samples** directory.

Click the **Open Directory to Browse** button () to open your file system and navigate to the directory that you want to view in the File Explorer. The File Explorer can display one directory at any given time.

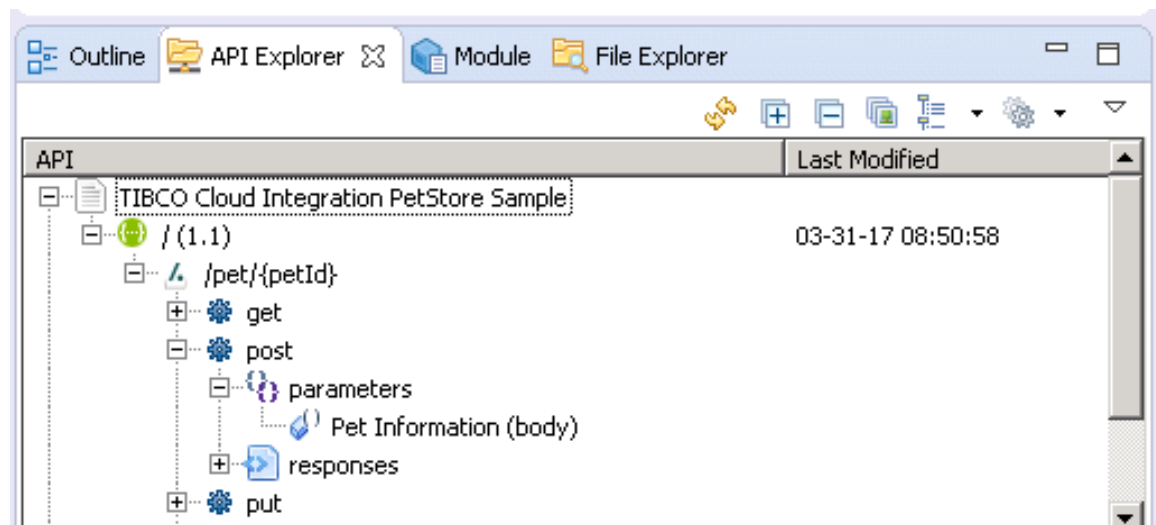
To revert to the samples directory, click the Go to Default Samples Directory button ().


Click the back arrow to go to a previous location or the forward arrow to go to the next location in case you had navigated to a previous location.

You can also open the directory in your Windows file system by right-clicking on the path in the File Explorer and selecting **Open Location** from the resulting menu. Select **Create Folder** to create a new folder under that directory. The **Import Selected Projects** option allows you to open the projects in the Project Explorer.


API Explorer

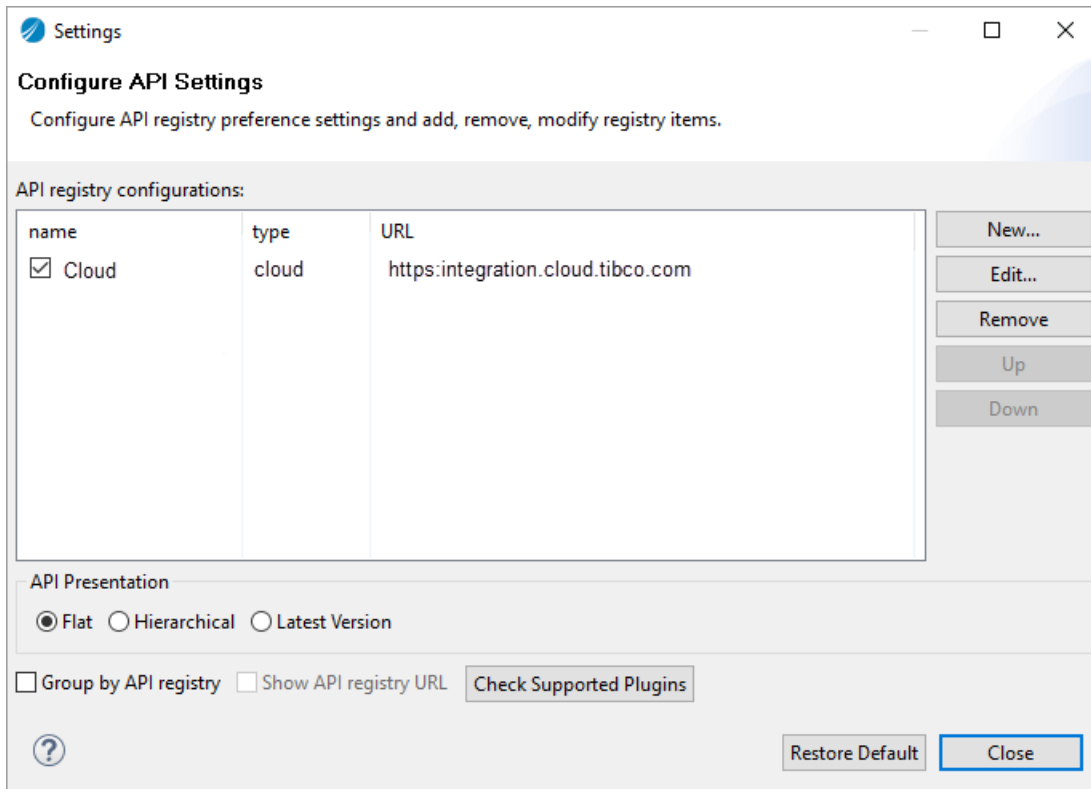
Displays a connected view of the TIBCO ActiveMatrix BusinessWorks™ API Modeler residing in the cloud. This view shows abstract APIs that were created in API Modeler. You can also view the APIs residing on your local machine from the **API Explorer**.



When you open TIBCO Business Studio™ for BusinessWorks™ for the very first time, enter your credentials for the registry site by opening the Settings dialog box and double-clicking on the registry name and entering your username and password for the site in the resulting dialog box. To open the Settings dialog, click the () button on the upper right corner of the **API Explorer** view and click **Settings**. This populates the **API Explorer** view with the APIs that are available in the registry.

Adding a new registry to the API Explorer view

Use the Settings dialog in the **API Explorer** to add a new registry (location) from where you want to view the APIs. To open the Settings dialog box, click the () button on the upper right corner of the **API Explorer** view, and click **Settings**.



By default, the Settings dialog box is configured with a Cloud registry which is set to the URL for the API Modeler.

To create a new registry:

1. Click the **New** button.
2. Enter a name for the registry **Name** field.
3. Select whether the registry is pointing to a local folder on your machine (**Local Folder**) or to a URL in the cloud (**Cloud**).
4. Provide the location of the registry in the **URL** field. If the registry points to a location on the cloud, you need to provide the authentication details for it in the **Username** and **Password** text boxes.
5. Click **Finish**.

To edit an existing registry entry:

1. Click the name of the registry and click **Edit**.
2. Make your edits to the entry. You can change the name of the registry, delete the registry configuration by clicking **Remove**, or changing the order in which the registries show up in the API Explorer by using the **Up** and **Down** button.
3. Click **Finish** when you are done with your edits.

Select a specific registry entry check box to display the registry in the **API Explorer** view. For more information, see [Filtering the APIs in the API Explorer View](#).

Setting the presentation of the APIs in the API Explorer view

In this dialog box, you can specify how the discovered APIs appear in the **API Explorer** view:

- **API Presentation** - specifies how the APIs appear in the **API Explorer** view

Flat - displays the APIs as a flat list with each API's version number displayed next to its name in parenthesis. If there are multiple versions of the same API, each version is shown as a separate API, hence multiple APIs with the same name but different version numbers.







Hierarchical - displays every API as a hierarchy of API name label with version number folder under it and the actual API under the version folder. If there are multiple versions for an API, each version is listed in its own separate folder under the API name label.

Latest Version - displays only the latest version of the API, even though there might be multiple versions available.

- **Group by API registry** - groups the APIs according to the registry from which they were discovered. You also have the option to display the URL of the APIs next to the registry name by selecting the **Show API Registry URL** check box.
- **Group apps by sandbox** - If you have multiple sandboxes that contain apps, the **Cloud Applications** view displays the sandboxes and groups the apps under their respective sandbox.
- **Check Supported Plugins** - If your application uses plug-ins, you must verify that the plug-ins are supported in ActiveMatrix BusinessWorks before you push the application to the cloud. You can do so by clicking this button.

You should now see the APIs displayed in the API Explorer in the format that you specified in the **Settings** dialog. Expanding an API shows you its version, the resource path, and the operations you can perform on that resource.

The **API Explorer** view has the following quick-access buttons that you can use to format the way the APIs are listed:

-  Refresh
-  Expand All
-  Collapse All
-  Group by API Registry
-  API Presentation
-  API Registries. Selecting a registry from this drop-down list toggles between displaying and hiding the registry in the API Explorer.

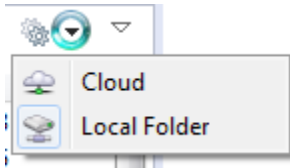
Searching for APIs in API Explorer

Use the search filter that appears at the bottom of the API Explorer view to search for API names that match the string that you enter in the **Filter** text box. You can search by typing in the version number, the full API name, or a full word within an API name. Wildcards is not supported. The search is case insensitive.

Filtering the APIs in the API Explorer view

If your APIs reside in multiple locations and you have set up the API registries in the Settings dialog of the API Modeler view, you can filter the APIs in API Modeler such that it shows you only the APIs available in a certain registry.

To do so, click the  button on the upper right corner of the API Modeler view and select the registry whose APIs you want to view.



Process Editor

Process Editor is the canvas in which you design and create your process.

You can click an activity in the activities palette located to the right of the Process Editor and drop it in the Process Editor by clicking anywhere within the process boundary or you can add an activity from the right-click menu accessible from within the Process Editor. Use [Transitions](#) to create a flow between the activities.

To open an existing process in the Process Editor, double-click the *<process>.bwp* in the Project Explorer. The process diagram opens in the Process Editor.

Palette Library

TIBCO Business Studio for BusinessWorks comes with a variety of Palettes each of which contain multiple activities relevant to the Palette.

Click the Palette name to see which activities are available for the palette. To use an activity in your process, click the activity, then move your cursor anywhere within the process boundary in the Process Editor and click again.

Entity Naming Conventions

Most of the ActiveMatrix BusinessWorks named entities are modeled as NCNames (or as a subset of an NCNames). These include activity names, service names, reference names, binding names, and component names.

Process names and shared resource names are represented as a subset of an NCName as they do not allow the use of a dot (.) character in their names. A small set of named entities are modeled as OSGi symbolic names. This set includes application names, module names, process package names, and shared resource package names.

NCName stands for XML "non-colonized" name. See <http://www.w3.org/TR/xmlschema-2/#NCName> for the W3C definition of NCName. NCName represents the set of characters that conforms to the following restrictions:

- Can include letters or numbers A-Z, a-z (lower case letters), 0-9, -, _ (underscore)
- Cannot include the following characters: @, :, \$, %, &, /, +, ,, ;,), A-Z (uppercase letters), - (hyphen) and white space characters.
- Cannot begin with a number, dot (.), or minus (-) character. However, these characters can appear later in an NCName.

The OSGi symbolic name is defined as part of the OSGi Specification, which is available at <http://www.osgi.org/download/r5/osgi.core-5.0.0.pdf>. OSGi symbolic names are represented using the following syntax:

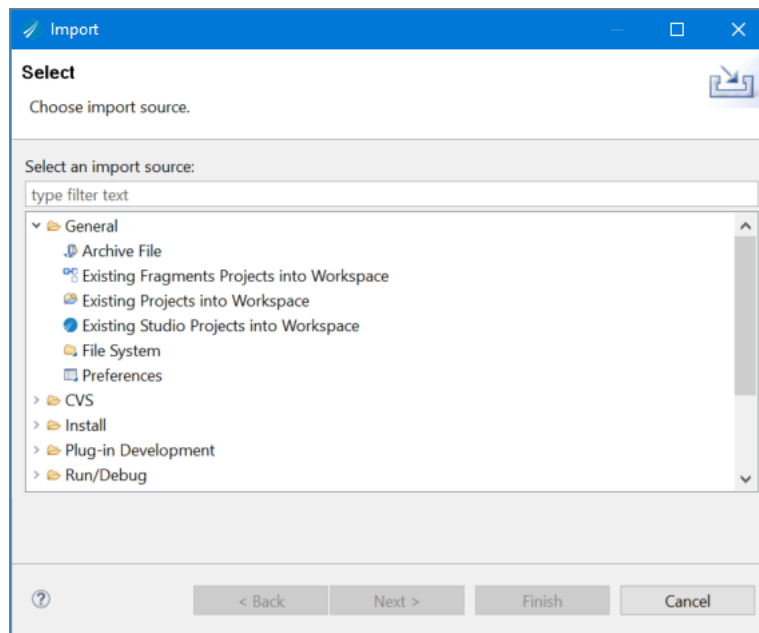
```
symbolic-name ::= token('.'token)*
token ::= ( alphanum | '_' | '-' )+
alphanum ::= alpha | digit
digit ::= [0..9]
alpha ::= [a..zA..Z]
```

Importing an Existing Project into Workspace

To import existing projects into workspace from TIBCO Business Studio™ for BusinessWorks™, follow these steps.

Procedure

1. Navigate to **File** and click **Import...**
The **Import** wizard is displayed with the **Select** page.

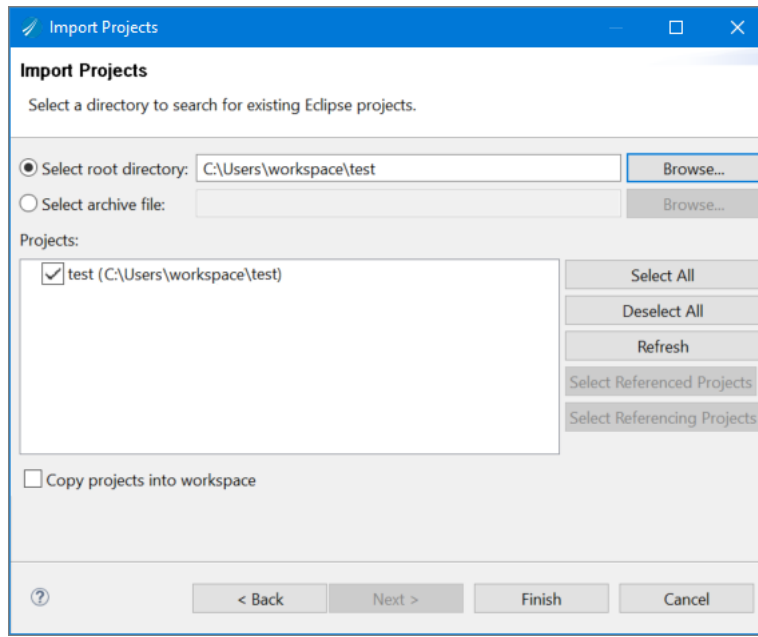


2. Select the **Existing Studio Projects into Workspace** option available under **General** category. Or type the source name text as **Existing Studio Projects into Workspace** in the **Select an import source:** input field.
3. Click the **Next** button.
The **Import** wizard displays the **Import Projects** page.
4. Select the **Select root directory:** option to select the path of the directory, where the required project is stored.
5. Click the **Browse** button next to the **Select root directory** input field. Or copy and paste the path of the required project directory in the **Select root directory:** input field.
Browse For Folder wizard is displayed.



If you want to import the projects in a .zip file, select the **Select archive file** option, and then click the **Browse** button next to the **Select archive file:** option. Or copy and paste the path of the required .zip file in the **Select archive file:** input field.

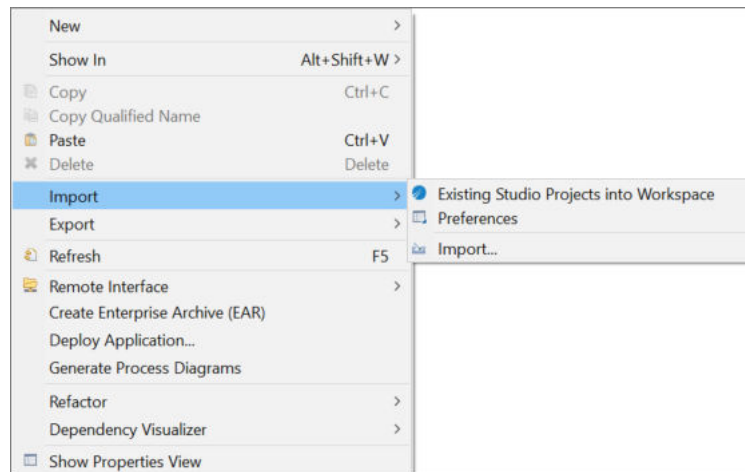
6. Navigate to the required directory, where the required project is stored.
7. Click the **Ok** button on the **Browse For Folder** wizard.
The projects available under the selected directory are displayed in the Projects area of the **Import Projects** wizard.
8. Select the required project(s) to import.
9. Select the **Copy projects into workspace** option.



10. Click **Finish**.

The green color status bar indicates the status of the import process and imported projects are displayed in the **Project Explorer** pane.

- You can also import the projects by right-clicking in the **Project Explorer** and navigating to **Import > Existing Studio Projects into Workspace**. In this way of importing a project, the **Import** wizard is displayed with the **Import Projects** page, and the **Select** page is skipped.



- The import functionality is also available from the command line interface. For more information, see [Using bwdesign](#).

Developing a Basic Process

Using processes you can implement business logic that obtains and manages the flow of information in an enterprise between a source and different destinations.

TIBCO Business Studio™ for BusinessWorks™ Workbench provides a design environment to develop and test a process. Developing a simple process consists of the following phases:

1. [Creating an Application Module](#) to contain the processes and shared resources.
2. [Creating a Shared Module](#) (optional).
3. [Creating a Process](#) that implements the business logic.
4. [Working with Process Properties](#) to define the runtime behavior of the process.
5. [Adding activities](#) to the process that describe the tasks in the business logic.
6. [Connecting Activities with Transitions](#) to describe the business process flow between activities in a process.
7. Configuring the input and output data for the activities. For more information, see [Working with Standard Activity Features](#).

At run time, the process engine executes the process definition and creates an instance of the process definition called a *job*. A job automates your business process by executing what is described in the process definition.




Conceptual information about processes and their use is provided in the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide.

Creating an Application Module

Application modules are packages containing one or more processes, shared resources, and metadata such as name, version, dependencies, and so on.

The New BusinessWorks Application Module wizard helps create an application module. There are multiple ways to launch the wizard:

- From the main menu, select **File > New > BusinessWorks Resources** and then select  **BusinessWorks Application Module**.
- Right-click in the Project Explorer view and choose **New > BusinessWorks Application Module**.

Specify the values for the following fields in the wizard:

1. **Project name:** Name of the application module.
2. **Use default location:** Specifies the location on disk to store the application module's data files. By default, this value is set to the workspace. To change, clear the check box and browse to select the location to be used.
3. **Version:** Version of the application module.
4. **Deployment Target:** Select the required deployment platform(s).



Optional. You can set the default deployment profile to create applications, and migrate the existing TIBCO ActiveMatrix BusinessWorks™ 5.x projects with the set preference. Navigate to **Window > Preferences > BusinessWorks > Deployment Profile**.

5. Depending on the deployment platform selected by user, the project name will be followed by the target platform names. For example, **tibco_bw_sample_palette_http_requestresponse [Container, Tibco Cloud, AppSpace]**.

6. **Create empty process:** Selected by default to create an empty process with the specified name (default: Process). Clear the check box if you do not want to create an empty process.
7. **Create Application:** Selected by default to create an application with the specified name. Clear the check box if you do not want to create an application.
8. **Use Java Configuration:** Select to provide the Java tooling capabilities in your module. Selecting this option creates a Java module.
9. Click **Finish**.



You can add identical package names in 2 different shared modules


Result

An application module with the specified name is then created and opens in the workbench. If the option to create an empty process and an application were selected, the process and application with the specified names are also created.

Creating a Shared Module

Shared modules are the smallest unit of resources that are named, versioned, and packaged as part of an application and can be used by other modules that are part of the same application.

The New BusinessWorks Shared Module wizard helps create a shared module. There are multiple ways to launch the wizard:

- From the main menu, select **File > New > BusinessWorks Resources** and then select  **BusinessWorks Shared Module**.
- Right-click in the Project Explorer view and select **New > BusinessWorks Shared Module**.

Specify the values for the following fields in the wizard:

1. **Project name:** Name of the shared module.
2. **Use default location:** Specifies the location on disk to store the shared module's data files. By default, this value is set to the workspace. To change, clear the check box and browse to select the location to be used.
3. **Version:** Version of the shared module.
4. **Deployment Target:** Select the required deployment platform(s).



Optional. You can set the default deployment profile to create applications, and migrate the existing TIBCO ActiveMatrix BusinessWorks™ 5.x projects with the set preference. Navigate to **Window > Preferences > BusinessWorks > Deployment Profile**.

Deployment target support for dependency modules when refactoring the platform support for dependent modules adds the target support instead of overwriting it. For example,



Application1 : Configured to AppSpace and uses SharedModule1

Application2 : Configured to AppSpace and uses SharedModule1

SharedModule1 : Configured to AppSpace

If the deployment target platform for Application2 is reconfigured to Container, then SharedModule1 will now be configured to both AppSpace and Container.

5. Depending on the deployment platform selected by user, the project name will be followed by the deployment target names. For example, **tibco_bw_sample_palette_http_requestresponse [Container, Tibco Cloud, AppSpace]**.

6. **Use Java Configuration:** Select to provide the Java tooling capabilities in your module. Selecting this option creates a Java module.
7. Click **Finish**.

Result

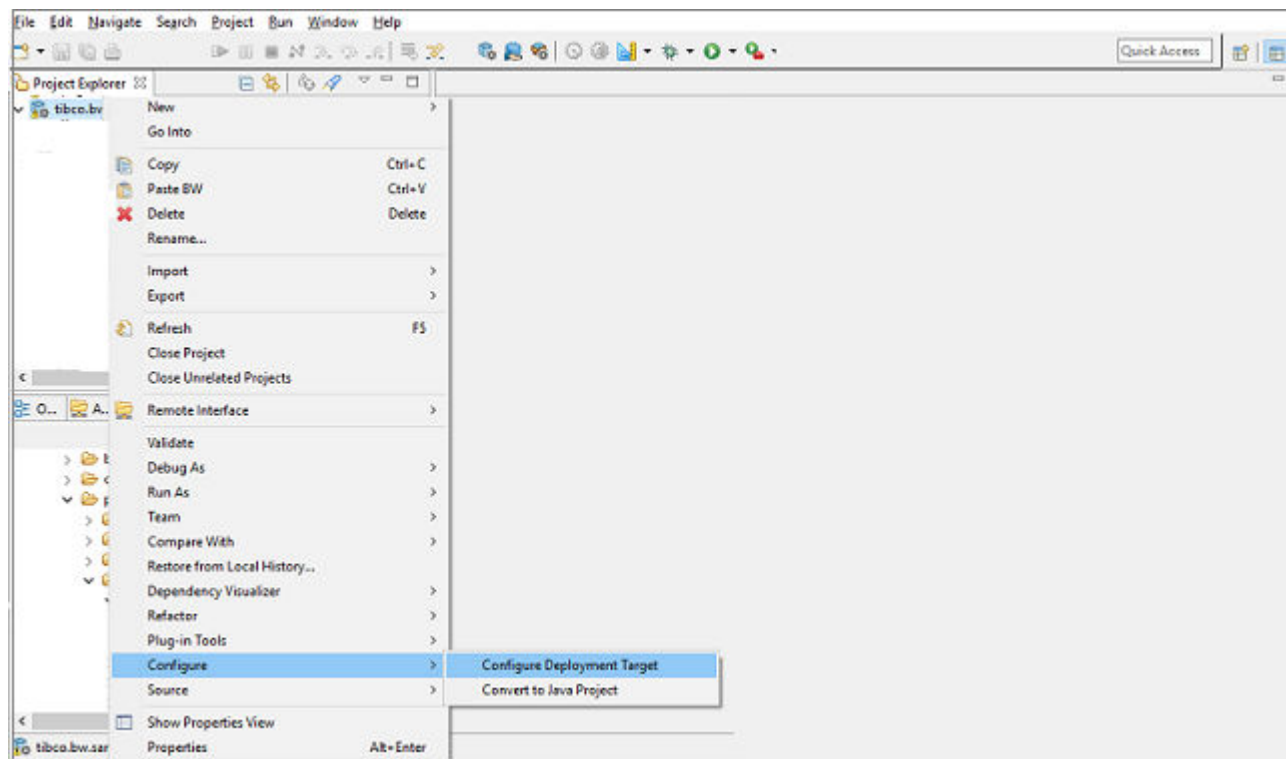
A shared module with the specified name is created and opened in the workbench.

Reconfiguring Deployment Target

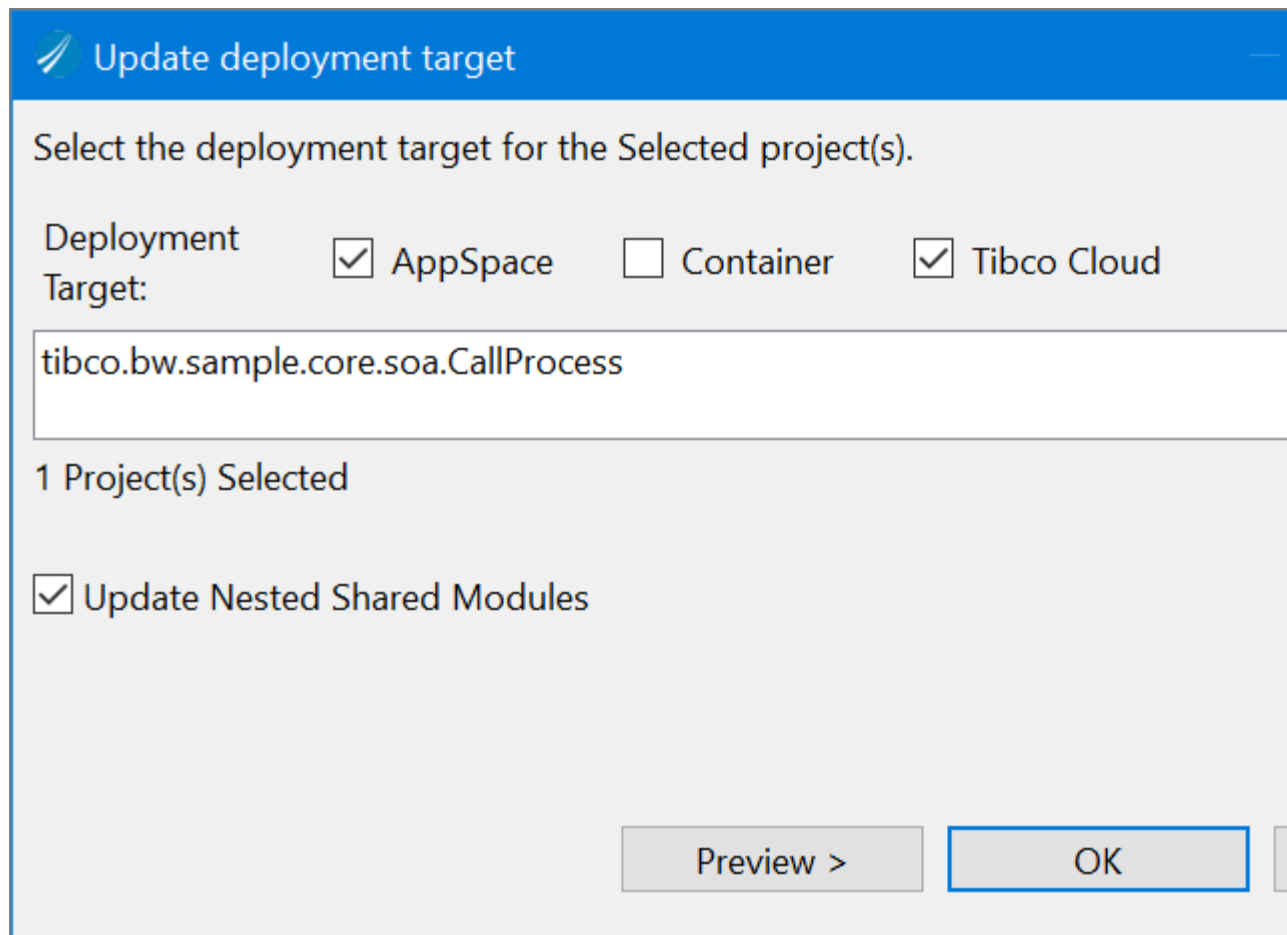
Applications can be reconfigured to be run on a different platform or can be configured to more than one platform at the same time. Using the **Configure Deployment Target** option, applications can be developed and run on the Enterprise edition (AppSpace), Container or the TIBCO Cloud edition.

Procedure

1. To reconfigure an application to a different deployment targets for an application, navigate to **Configure > Configure Deployment Target**.



2. In the **Update deployment target** window, select the target platform(s) to configure or reconfigure the application.



3. The **Update Nested Shared Modules** check box refactors all the nested shared modules in the application and is selected by default.
4. Once the deployment target is updated, export the EAR file and deploy it to the selected platform. Options such as **Push to Cloud** and **Deploy Application** are displayed.
5. On opening the projects, only the palettes and activities supported by the target platform(s) configured for a project will be displayed.



You can also set the default deployment profile to create applications, and migrate the existing BusinessWorks 5.x projects with the set preference. Navigate to **Window > Preferences > BusinessWorks > Deployment Profile**.

Generating the `manifest.json` File Using the `bwdesign` Utility

In order to push an application created in TIBCO ActiveMatrix BusinessWorks™ or to TIBCO Cloud Integration, there must be a `manifest.json` file that defines your application. Applications that were built with versions before TIBCO Business Studio for BusinessWorks 1.1.0 do not have the `manifest.json` file generated and bundled with their EAR file and hence are not enabled for the TIBCO Cloud Integration environment. If you would like to push such applications to TIBCO Cloud Integration, you must generate a `manifest.json` file for them.

The `manifest.json` file can be generated from the `bwdesign` utility as follows:

```
generate_manifest_json ear_location manifest_location
```

Procedure

1. Open a command prompt or terminal window.

2. Navigate to `<BW_HOME>\bin` directory.
3. Enter the following command:
`bwdesign`
4. Enter the following command:



Use a new fresh clean workspace for `manifest_location` when running the following command for generating the `manifest.json` file.

```
generate_manifest_json ear_location manifest_location
```

where `ear_location` is the path to the EAR file and `manifest_location` is the location where you would like to save the generated `manifest.json` file.

For more information on using the utility, see [Using the bwdesign Utility](#).

Generating the `manifest.yml` file

To push an application created in TIBCO ActiveMatrix BusinessWorks™ or TIBCO Cloud™ Integration to Pivotal Platform the application manifest (`manifest.yml`) file is required. In ActiveMatrix BusinessWorks™ the `manifest.yml` file can be created from the `Context` menu option, **Create Manifest YML** for an application in the **Project Explorer** view.

Exporting a Shared Module as a Binary Shared Module

You can create a binary shared module from a shared module. However, you cannot convert a binary shared module to a regular shared module.

To export a shared module as a binary shared module, begin by implementing the process you want to share. The process must have a descriptive name and a description. Next, test the process by calling it from a test application. Once satisfied, you create a zip archive file for the project which contains the process and distribute that zip using a mechanism such as email, FTP, or a web page, that is external to TIBCO Business Studio for BusinessWorks.



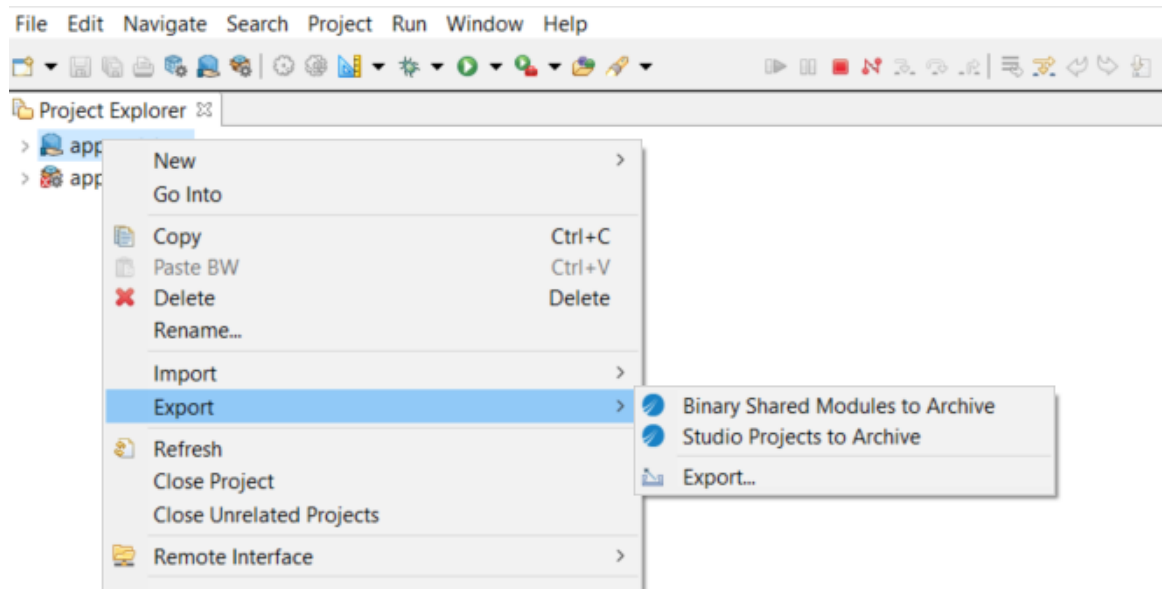
Back up the shared module by exporting the project as an archive file. To do this, select **Export > Studio Projects to Archives**.

TIBCO Business Studio for BusinessWorks

To export a shared module as a binary shared module from TIBCO Business Studio for BusinessWorks, follow these steps.

Procedure

1. In Project Explorer, right-click the shared module folder, and choose one of the following options to begin exporting the shared module as a binary shared module:
 - Select **Export > Export**. In the Export dialog, expand the **General** node, select **Binary Shared Modules to Archive**, and click **Next**.
 - Select **Export > Binary Shared Modules to Archive**.



2. Select the check box of the shared module to convert to a binary shared module.
3. In the **To Archive File** field, navigate to the folder where you want it created and enter a name for the binary shared module you want to create and click **Save**.
4. Click **Finish** in the Export Project dialog.

Result

The shared module is exported as a binary shared module.

To confirm the shared module was successfully exported as a binary shared module, import the binary shared module into a new workspace, and expand the project. All application folders and details, with the exception of the folders under the Module Descriptors folder, are hidden. Optionally, check the MANIFEST.MF file, and confirm the TIBCO-BW-SharedModuleType header is set as follows:

```
TIBCO-BW-SharedModuleType: binary
```

CLI

To export a shared module as a binary shared module from the command line, follow these steps:

Prerequisites

- Start the bwdesign utility. To do this, follow these steps:
 1. Open a terminal and navigate to BW_HOME\bin.
 2. Type `bwdesign -data <TIBCO_BusinessStudio_workspace_absolutePath>`. For example, `bwdesign -data C:\myWorkspace`.
- Back up the shared module by exporting the project as a zip or EAR file. To do this, type `-export [options] [projects] [outputfolder]`

Type `export -binary <shared_module>`. For example, `export -binary shared_petstore`. Optionally, type `export -bin <shared_module>`. For more details about the `-binary` and `-bin` commands, type `export --help`.

Result

The shared module is exported as a binary shared module.

To confirm the shared module was exported as binary shared module, import the binary shared module into a new workspace by typing `bwdesign -data`

<TIBCO_BusinessStudio_workspace_absolutePath>. After doing this, expand the project in the Project Explorer to verify that all application folders and details, with the exception of the folders under the Module Descriptors folder, are hidden. Optionally, check the MANIFEST.MF file, and confirm the TIBCO-BW-SharedModuleType header is set as follows:

```
TIBCO-BW-SharedModuleType: binary
```

Using Binary Shared Modules in your Project

To use a binary shared module, you begin by importing the archive into your workspace where it appears like any other shared module, except that the internal details of the shared module are not visible. You use a binary shared module in the same way as you would use any other shared module. You can see the processes in the Project Explorer but cannot view their diagrams in the Process Editor or open them with a text editor to decipher their models.

You can see the following artifacts associated with a binary shared module:

- Process and package name
- XML schema files associated with the module



Because the schema files are in plain text, you will be able to modify them. Keep in mind though that if and when you import a newer version of the module into your workspace, your modifications to the schema files will be overwritten.

- Shared resources - you can reference them, but cannot edit them
- Module Descriptor folder - only the Overview item is available under this folder
- Module Descriptor editor will be able to display the Overview page only. All other fields will be disabled

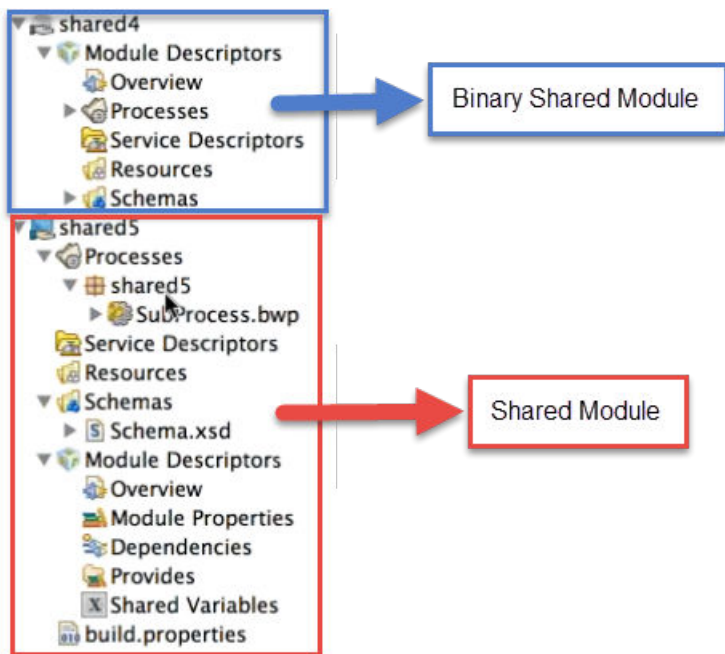
You can implement a Call Process activity that invokes the functionality in the binary shared module. When deploying your application, the binary shared modules are included in the application like any other shared module.

Difference between a Shared Module and a Binary Shared Module

This section describes the difference between a shared module and a binary shared module.

In Project Explorer

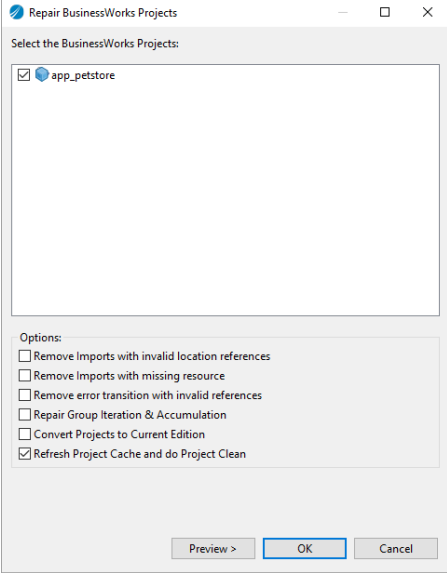
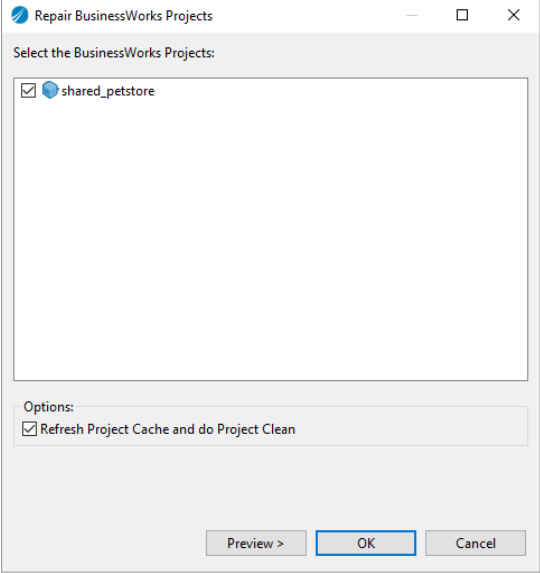
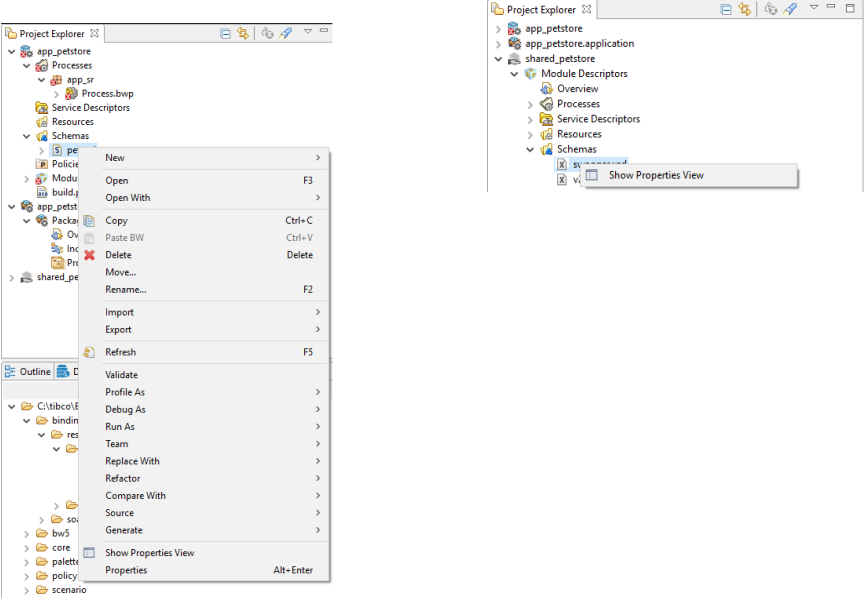
The image below shows you the difference between a shared module (shared5, in the image below) and a binary shared module (shared4). Notice that almost all the editable artifacts (such as Module Properties, Dependencies and Shared Variables) are missing from the binary shared module tree. This is one way to prevent the binary shared module from being edited.



Menu Items

At the project level some of the context menus items are disabled in the binary shared modules. At the resource level all the menu items except for Show Properties View are disabled.

Context Menu	Shared Module	Binary Shared Module
<p>At the project level: Right-click menu from process name</p>		

Context Menu	Shared Module	Binary Shared Module
Repair BusinessWorks Projects dialog		
Context menus at Processes, Service Descriptors, Resources, and Schemas level		

Public Processes and Internal Processes

A binary shared module can contain two types of processes - public processes and private (internal or inline) processes. While a public process in a binary shared module can be called by an application, a private process within the module is meant for consumption by the public processes within that binary shared module only. By default, the private processes are **not** visible in the Project Explorer.

To view the private processes in the Project Explorer, do the following:

1. In the Project Explorer, click the **View Menu** button (☰) and select **Customize View**.

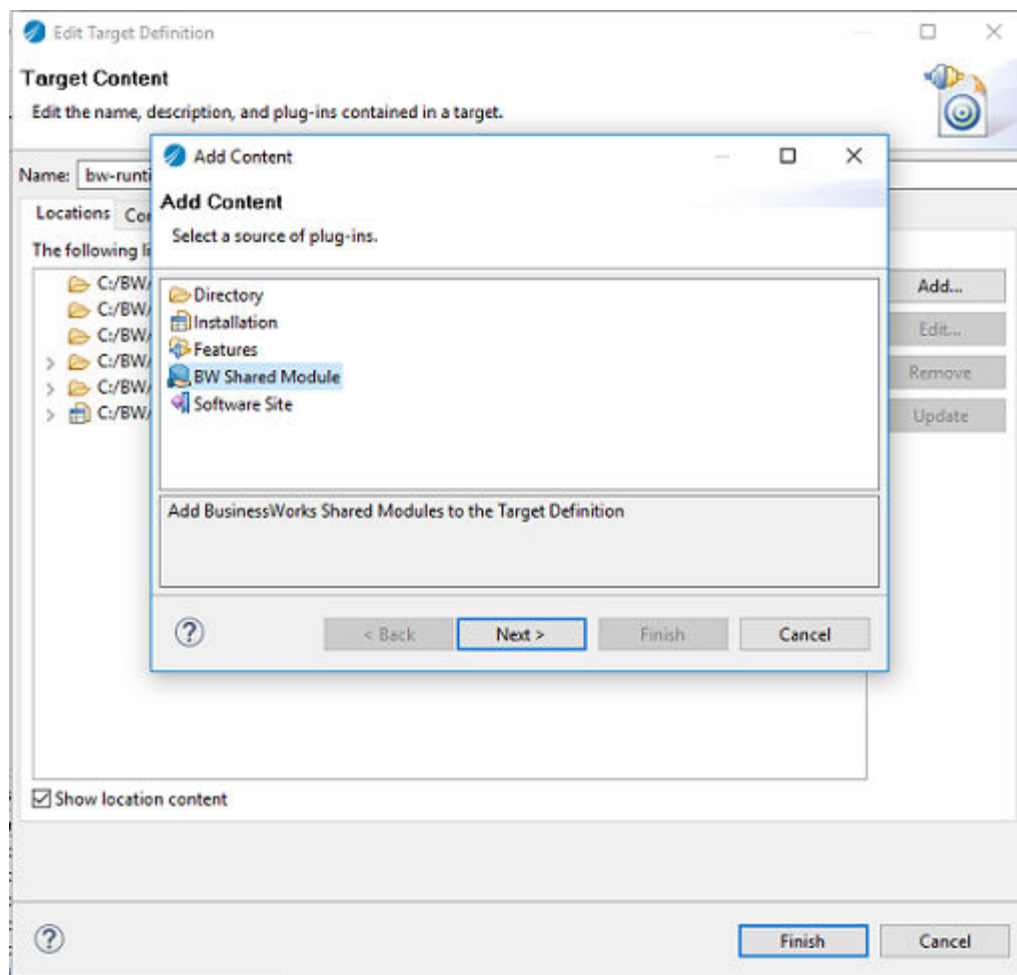
2. In the Available Customizations dialog, deselect the **BW binary private processes** check box and click OK.

Referencing Shared Modules

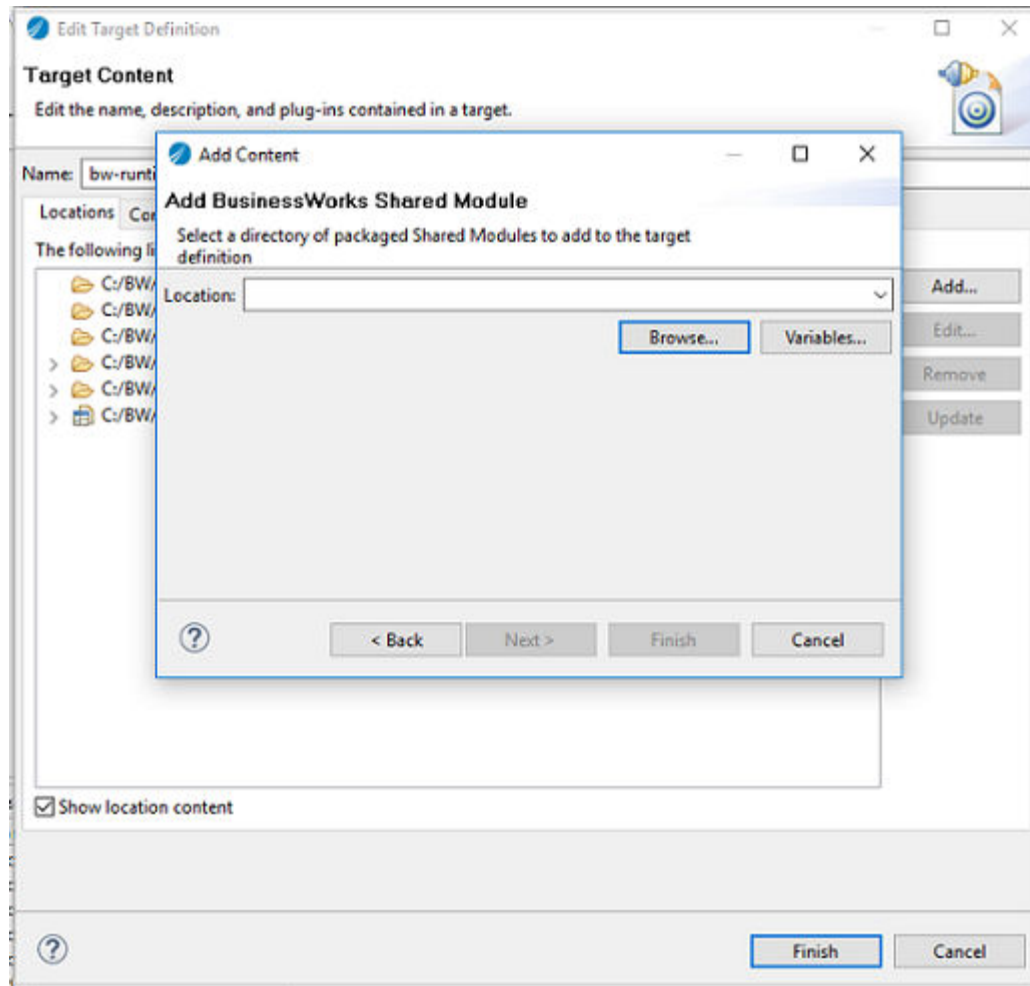
You can import shared modules from an external location to a shared location where other users can reference them. The shared modules that you import are read-only.

Procedure

1. To reference external shared modules, navigate to **Window > Preferences > Plug-in Development > Target Platform**. Using the target platform option you can add, delete, or edit target definitions. The exported definitions are stored locally and can be moved to a project and shared with other users.
2. On the **Target Content** dialog box, click **Add**.
3. On the **Add Content** dialog box, select the option **BW Shared Module** and click the **Next**.



4. On the **Add BusinessWorks Shared Module** dialog box, browse to the location of the external shared module ZIP folder to add the shared module to the target definition.



5. In the Application Module, navigate to **Module Descriptors > Dependencies** and click the **Add** button to view all the available external shared modules. To add the required external shared module, select the shared module and click **Ok**.
The external shared module can then be opened in the read-only mode.



Creating a Process

Processes are always contained in a process package. When creating a process, either create a new process package or select an existing package in which the new process is to be created.

Prerequisites


A module must exist to which processes can be added. If a module does not exist, create a new module before creating a process.

The BusinessWorks Process Creation wizard helps create a generic business process. By default, it is configured to create a process with name `Process`. There are multiple ways to launch the wizard:

- From the main menu, select **File > New > BusinessWorks Resources** and then select  **BusinessWorks Process**.
- From the **Module Descriptors > Overview** getting started area, click  **Create a BusinessWorks Process**.

- Right-click the **Processes** folder in the **Project Explorer** view, and then select **New > BusinessWorks Process**.

Specify the values for the following fields in the New BWProcess Diagram wizard:

Field	Description
Process Folder	Name of the module and the Process special folder where the process will be located. You can add multiple folders in Project Explorer and then update this field to select the new folder. For example: <code>bw.test.app/Processes</code> .
Package	Name of the package in the module where the new process is added. Accept the default package, or browse to select a different package name. For example: <code>bw.test.app.main</code> .
Process Name	Name of the new process. For example: <code>MainProcess</code>
Modifiers	Designate whether the process will be public or private. This can be changed later.
Patterns	Choose the pattern Empty Process when creating a process. <div style="display: flex; align-items: center;">  <div> <p>To create a subprocess, choose the pattern Subprocess. For more information, see Creating Sub-Processes on details for creating a subprocess.</p> </div> </div>

Click **Finish** to create a new empty process. A process with the specified name is created and opened in the Process Editor.

What to do next

After creating the process proceed to:

- Configure the process as described in [Working with Process Properties](#)
- Add activities to the process as described in [Adding Activities](#)

Working with Process Properties

Process configuration defines the behavior of a process at runtime. You can specify, or edit, the modifiers, mode, and activation type for a process. You can also define process properties and process variables, add or remove services and references, and configure the process dependencies. Open a process in TIBCO Business Studio™ for BusinessWorks™ if it is not already open and go to the **Properties** view. Configure the properties for a process by selecting the appropriate tab in the **Properties** view.

General


Property Name	Description
Package	Displays the name of the package containing the process. This field is not editable. To rename the package name, select the bulb icon on the right side. It open a Rename Package dialog box. Change the package name using the Rename Package dialog box.
Name	Name of the process. This field is not editable. To rename the process name, select the bulb icon on the right side. It open a Rename Process dialog box. Change the process name using the Rename Process dialog box.

Description

Property Name	Description
Description	Description of the process.

Advanced

Property Name	Description
Target Namespace	Target namespace for the process. You can specify a different target namespace.
Modifiers	Modifiers define the visibility of the process outside its package: <ul style="list-style-type: none"> • Public: can be invoked by processes that are defined either inside or outside the package. • Private: can be invoked only by processes that are part of the same package.

Property Name	Description
Mode	<p>Mode defines whether the process depends on the engine to maintain its state:</p> <ul style="list-style-type: none"> • Stateful: Stateful processes maintain the state across multiple operations. They are better suited when you need the server to maintain the state across operations. For processes that involve related message exchanges between the same or different consumers, conversations can be used to maintain state across operations. • Stateless: Stateless processes do not maintain state. They are better suited when you need to process higher loads of requests as each operation is executed independently. They do not require correlation or conversations between multiple operations in a process, thus allowing the server to process each operation without maintaining any state information. The client can choose to maintain the state information, if needed.
Activation	<p>Activation mode for a process defines the way in which processes are activated at runtime.</p> <ul style="list-style-type: none"> • Multiple AppNodes: At runtime, the application is distributed and activated on all the AppNodes in the AppSpace. In the event of a failure on one of the AppNodes, the application continues to run with fewer AppNodes. • Single AppNode: At runtime, the application is activated on only one AppNode in the AppSpace. In the event of a failure, another AppNode will be activated and any check pointed data can be recovered. <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-top: 10px;">  <p>This feature requires the engine persistence mode to be set to group and the database and group provider to be configured. See Configuring Database for the Engine and Configuring Group Provider for the Engine for details.</p> </div>

Property Name	Description
Activity Error Variable	<p>By default, this check box is selected for migrated processes. During migration, activity error variables are created for activities in the process that contain error transitions. Additional activity error variables are also created for activities with fault types, and or, if new activities with fault types are added to the process.</p> <p>If you are configuring a process created in ActiveMatrix BusinessWorks 6.x, and you select this check box, activity error variables are created for activities in the process that have fault types. If new activities with fault types are added to process, additional activity error variables are created.</p> <p>Clear the check box removes activity error variables created for activities with fault types. Global error variables and activity error variables for activities with error transitions are not affected.</p>

Property Name	Description
Namespace Registry	<p>Namespaces and prefixes can be configured at the Process level. Click the Configure namespace registry link field from the Advance tab of the Process configuration to view, add, change or delete prefixes for namespaces used in the input bindings of the activities in the process definition. Process namespace registry applies to the current process.</p> <p>Namespaces and prefixes can also be configured at the Module level. To add a new prefix for a namespace or to change the current namespaces and prefix configurations, from the Module Descriptors > Overview getting started area, click the Configure namespace registry link. Module namespace registry applies to all the processes in the module.</p> <p>If you have defined both, Process level and Module level configurations for a namespace, the Process level registry takes precedence over the Module namespace registry.</p> <p>When namespace registry is applied, prefixes in the activity input bindings are updated using the prefixes defined in the namespace registry where the namespaces are referred to.</p> <p>A list of namespaces and their prefixes is automatically populated when an input or output binding is created or modified. This list is populated at the Process level or at the Module level, depending on the preference set at Windows > Preferences > BusinessWorks > Namespace Registry.</p>

Process Properties

Add or delete process properties variables in the following format:

Property Name	Description
Property Name	Provide a property name.

Property Name	Description
Data Type	<p data-bbox="893 220 1212 262">Supported Data Types are:</p> <ul data-bbox="893 283 1276 1375" style="list-style-type: none"> <li data-bbox="893 283 1029 315">• Boolean <li data-bbox="893 325 1053 357">• DateTime <li data-bbox="893 367 1021 399">• Integer <li data-bbox="893 409 997 441">• Long <li data-bbox="893 451 1053 483">• PassWord <li data-bbox="893 493 1005 525">• String <li data-bbox="893 535 1085 567">• Data Format <li data-bbox="893 577 1101 609">• FTP Resource <li data-bbox="893 619 1085 651">• HTTP Client <li data-bbox="893 661 1141 693">• HTTP Connector <li data-bbox="893 703 1260 735">• Identity Provider Resource <li data-bbox="893 745 1149 777">• JDBC Connection <li data-bbox="893 787 1133 819">• JMS Connection <li data-bbox="893 829 1276 861">• JavaGlobalInstanceResource <li data-bbox="893 871 1268 903">• KeyStoreprovider Resource <li data-bbox="893 913 1197 945">• LDAP Authentication <li data-bbox="893 955 1189 987">• Notify Configuration <li data-bbox="893 997 1181 1029">• Proxy Configuration <li data-bbox="893 1039 1204 1071">• Rendezvous Transport <li data-bbox="893 1081 1125 1113">• SMTP Resource <li data-bbox="893 1123 1061 1155">• SSL Client <li data-bbox="893 1165 1252 1197">• Subject Provider Resource <li data-bbox="893 1207 1101 1239">• TCP Resource <li data-bbox="893 1249 1189 1281">• ThreadPool Resource <p data-bbox="893 1396 1364 1459">Data types may vary depending on the additional plug ins installed.</p>
Default Value	Provide the Default value based on a data type.

Process Variables

Process variables are used to store temporary data that are used by the process to store values other than simple output from an activity. Simple type or Complex Type variables can be created.

Services

Use the **Services** tab to create additional services.

References

Use the **References** tab to create additional references that are consumed by the process.



Dependencies

The **Dependency** tab can be used for troubleshooting any unresolved element-namespace issues in your process. This tab provides a view of what WSDL & XSD namespaces are currently being imported, and it also provides a way to add a new namespace import that will resolve a specific Element. If you choose the Element, the appropriate namespace import is then added to make sure that the element resolves.

Creating a Subprocess

Subprocesses are designed for complex business processes to make the main process easier to understand and debug. Subprocesses are called inside the main process and their output is used in the main process.

The BusinessWorks Process Creation wizard helps create a subprocess. There are multiple ways to launch the wizard:

- From the main menu, select **File > New > BusinessWorks Resources** and then select  **BusinessWorks Sub Process**.
- From the **Module Descriptors > Overview** getting started area, click  **Create a BusinessWorks Sub Process**.
- Right-click the **Processes** folder in the **Project Explorer** view, and then select **New > BusinessWorks Sub Process**.

Specify the values for the following fields in the New Subprocess wizard:

Field	Description
Process Folder	Name of the module and the special folder where the subprocess will be located.
Package	Name of the package in the module where the new subprocess is to be added. Accept the default package, or browse to select a different package name.
Process Name	Name of the subprocess.
Modifiers	Designate whether the process will be public or private. This can be changed later.

Field	Description
Interface Mechanism	<p>Select either Direct or Service</p> <ul style="list-style-type: none"> • Direct: Select this option to create a non-WSDL-based subprocess. When you select this option, a new subprocess, containing a Start and End activity, is created. • Service: Select this option to create a WSDL-based subprocess. Next, choose one of the following options: <ul style="list-style-type: none"> – Default: Select Inline to create an inline subprocess. Select Standalone to create a standalone subprocess. – Custom: Select this option and click Next to create a new WSDL interface or use an existing WSDL interface for the subprocess.

- Right-click the **Processes** folder in the **Project Explorer** view, and then select **New > BusinessWorks Process**.

Specify the values for the following fields in the New BWProcess Diagram wizard:

Field	Description
Process Folder	Name of the module and the special folder where the subprocess will be located.
Package	Name of the package in the module where the new subprocess is to be added. Accept the default package, or browse to select a different package name.
Process Name	Name of the subprocess.
Modifiers	Designate whether the process will be public or private. This can be changed later.
Patterns	<p>Select Standard Patterns > Process and then select one of the following options:</p> <ul style="list-style-type: none"> • Direct Subprocess • Service Subprocess <p>See the preview of the selected subprocess.</p>

Click **Finish** to create a subprocess.

Result

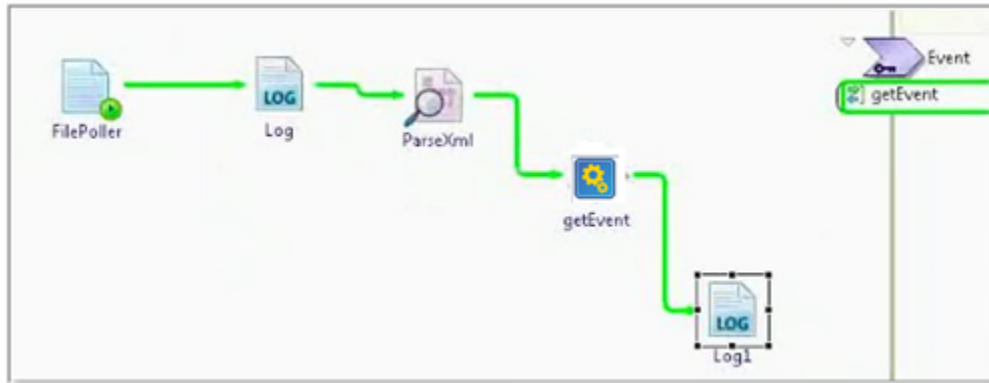
A subprocess with the specified name, and containing a **Start** and **End** activity, is created and opened in the Process Editor.

Parent Process and a SubProcess Example

Consider an example that illustrates how a parent process is designed to call a subprocess and collect data from that subprocess.

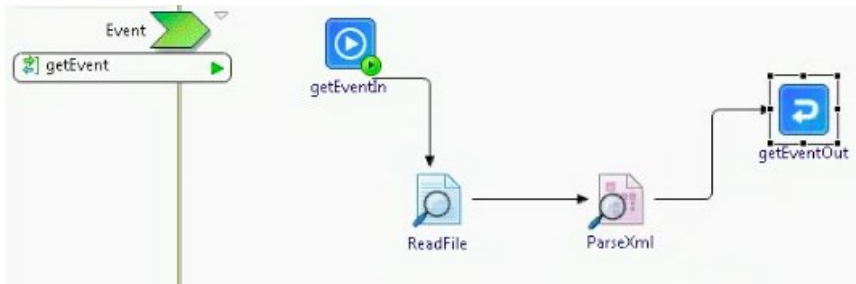
The parent process consists of a `getEvent` activity that calls the subprocess.

Parent Process



The subprocess implements the interface `getEvent` and returns the output back to the parent process. The parent process then logs the output received from the subprocess in a log file.

Sub Process




Creating an Activator Process

An activator process consists of two service operations, `On StartUp` and `On ShutDown`, which can be used to perform tasks when an application starts or after an application stops.

An application module can contain only one activator process. The following steps describe how to create an activator process for an application module.

Procedure

1. From the Module Descriptors > Overview > General Information area, click the  icon in front of the Activator Process input field.

Overview

General Information
This section describes general information about this BusinessWorks Module.

ID:	AppModule1
Version:	1.0.0.qualifier
Name:	AppModule1 Module
Vendor:	TIBCO Software Inc.
Module Type:	Application Module
Description:	
Activator Process:	

- Review the fields in the Create Activator Process wizard and click **Finish** to create an activator process. The **New BWProcess Diagram** wizard is displayed as follows.

New BWProcess Diagram

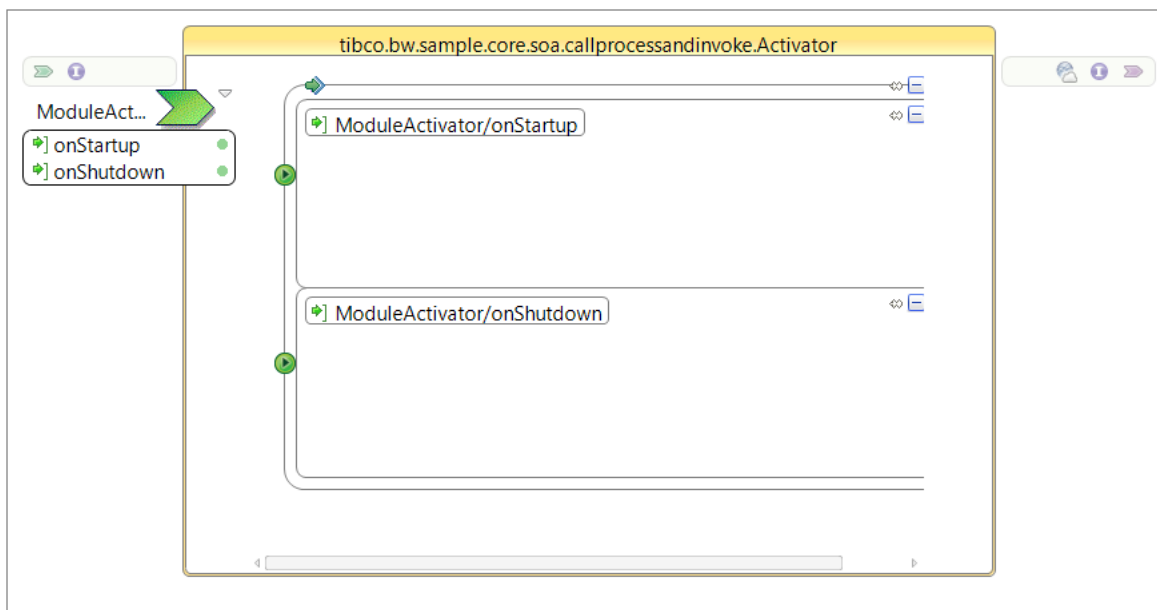
BusinessWorks Process Creation
Create Activator Process.


Select a folder for the process and enter a unique name.

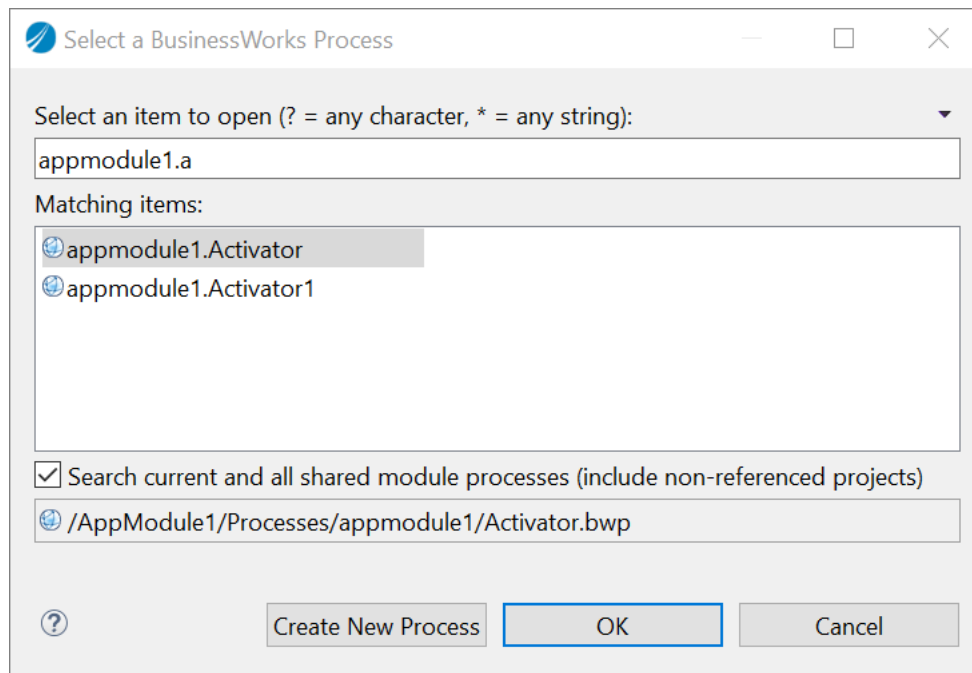
Process Folder:	tibco.bw.sample.core.soa.CallProcessAndInvoke/Processes	Browse...
Package:	tibco.bw.sample.core.soa.callprocessandinvoke	Browse...
Process Name:	Activator	

Result

An activator process with the service operations On StartUp and On ShutDown is created.



- You can change the activator process any time after its selection.
- You can choose the activator only from the module for which the **Overview** editor is opened, and locate anywhere within the module.
- To open the list of existing activator processes, click the  icon in front of the Activator Process input field. As you start typing the name of the existing process in the **Select an item to open (?=any character, *=any string):** input field, the matching results are displayed. The **Select a Business works Process** window is displayed as follows.



Adding Activities

Activities are the individual units of work in a process.

There are multiple ways to add activities in a process: from the right-click menu on the Process Editor, from the palettes, and from the File Explorer or Project Explorer.

Adding Activities from the Palettes

To add an activity to a process using the palette:

1. In the Palette view, select a palette from the library. All the activities available in the palette are displayed.
2. Select the activity that you want to add and drop it onto the process in Process Editor.
3. Configure the activity by specifying the values for the properties in the Properties view. The configuration properties are grouped under different tabs such as **General**, **Description**, **Input**, **Output**, **Conversation**, and so on. For example, upon adding a **Log** activity, you can configure it by specifying the values for the properties under the tabs: **General**, **Description**, and **Input**. See *Working with Standard Activity Features* for details.



General and **Description** tabs are available for all activities to enter their name and a short description. Depending on the activity, these tabs may include additional fields such as variables, time, shared configurations, and other values that are required by the activity. Activities can also contain additional tabs such as **Input**, **Output**, **Conversation**, **Fault**, and so on.

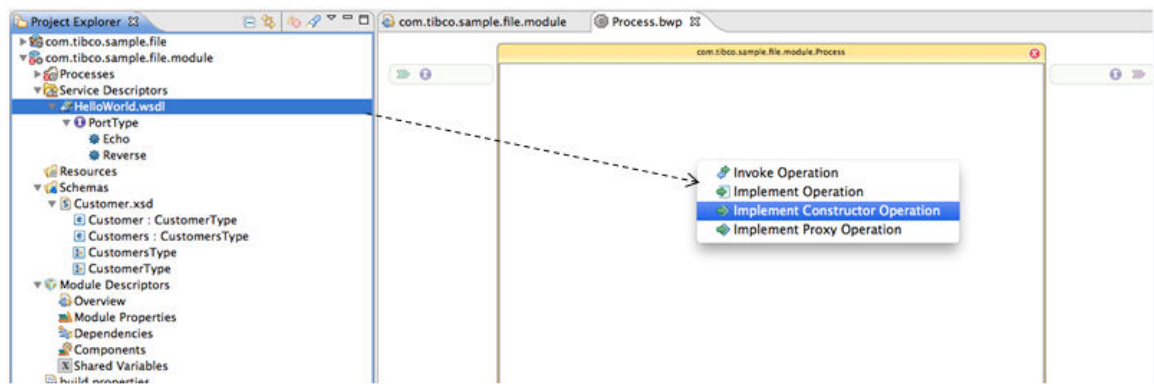
Adding Activities From the Project Explorer

You can add pre-configured activities to a process by dragging-and-dropping a selected resource such as a schema (XSD) or WSDL file from the Project Explorer. To do so, follow these steps:

1. In the Project Explorer, select a file such as a WSDL file that you want to use to create an activity.
2. Drag and drop the resource onto an existing process. The software parses the resource and provides a menu consisting of a list of pre-configured activities.
3. From the menu, select the activity you want to add to the process.

In the example, drag and drop the file `HelloWorld.wsdlEcho.wsdl` from the Project Explorer onto the process. A menu with a list of activities is presented. Select an activity to be added to the process.

Drag-and-Drop a Resource



An activity is connected to another activity by dragging the [+] symbol, positioning and dropping it, and then selecting the next activity from the menu selection. See [Working with Transitions](#) for details.

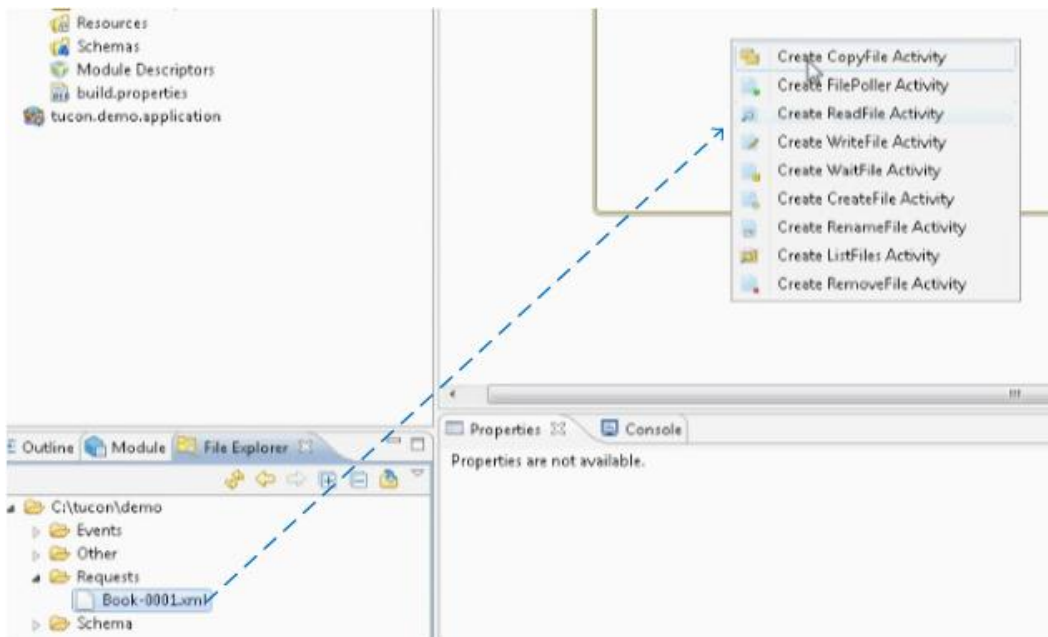
Adding Activities From the File Explorer

You can add pre-configured activities to a process by dragging-and-dropping a selected file such as an XML file from the File Explorer. To do so, follow these steps:

1. In the File Explorer, select a file you want to use to create an activity.
2. Drag and drop the resource onto an existing process. The software parses the resource and provides a menu consisting of a list of pre-configured activities from the File palette.
3. From the menu, select the activity you want to add to the process.

In the example, drag and drop the file `Book-0001.xml` from the File Explorer onto the process. A menu with a list of activities is presented. Select an activity to be added to the process.

Drag-and-Drop a Resource



An activity is connected to another activity by dragging the [+] symbol, positioning and dropping it, and then selecting the next activity from the menu selection.



Working with Transitions

Transitions are used to connect two activities to represent the flow of process execution from one activity to the other.

Transitions are added between activities in a process and are configured to fit the process goal.

Adding a Transition

You can choose to add a transition in one of the following ways:

- Click the **Create a Transition** icon  in the Palette view's toolbar and draw a line between two activities that are to be connected.
- Select the beginning activity of the transition, click the icon  and drag and drop it on to the ending activity of the transition.

Configuring a Transition

After creating a transition specify the configuration information on the **General** tab of the Properties view:

1. **Label:** Add a label for the transition that will be available in the diagram. This label can be changed later.
2. **Fill Color:** Select **Color** for the transition from the basic colors or define a custom color. Color coding helps you distinguish among different transitions based on the conditions that are defined for them. The default color for Error is red, while the default color for other transition types is black.
3. **Condition Type:** Select the type of the condition for the selected transition: Success, Success with condition, Success with no matching condition, and Error.

You can define several types of conditions for a transition:

Success

Take this transition unconditionally. If the activity completes successfully, always transition to the activity the transition points to. This is the default condition for transitions.

Success with Condition

Specify a custom condition using XPath. If the activity completes successfully, and the condition evaluates to true, take the transition to the pointed-to activity.

Success with no Matching Condition

Take this transition when the activity completes successfully but *only* if no other transitions are taken. This is useful when multiple transitions with conditions are drawn to other activities. This condition type can be used to handle any cases not handled by the conditions on the other transitions.

Error

Take this transition if there is an error during the activity processing.

Error Transitions

Error transitions are taken if there is an error during the processing of an activity or group. When an activity or group throws an error or fault, none of the success conditions are taken; only the error transition is executed. An error transition can be added to process starter activities, signal-in activities, regular activities, and groups.



Activities and groups only support one error transition at a time.

Working with Standard Activity Features

Specify the required configuration elements to make the activity work. These configuration elements are available in the Properties view.

Each activity usually has two or more of the following tabs for specifying the characteristics of the activity:

General

This tab is available for all activities. In addition to the name of the activity, it also sets other parameters such as questions about directories and overwriting for file activities, class name for Java activities, host name, and port number for mail activities, modifiers, mode, and activation settings.

Description

This tab is available for all activities. You can write down any information you need to preserve for the activity.

Statement

This tab is available for query activities; used to define, validate, and execute a query.

Advanced

You can specify any advanced configuration parameters here.

Event

For activities that wait for incoming events, such as incoming TIBCO Rendezvous™ messages, this tab specifies the timeout for the incoming event.

Conversations

Used to add new conversations. For more information about conversations, see [Using Conversations](#).

Input Editor

Used to edit an output element by adding a complex anonymous type, complex element, primitive element, and so on. Not all activities have this option enabled. For more details see [Input and Output](#).

Input

Using the tab you can map and transform output data from the previous activities in the process (including the event that starts the process) to input data for **Input** the activity. For more details see [Input and Output](#).

Output Editor

This tab is used to choose or configure the output header element. Not all activities have this option enabled. For more details see [Input and Output](#).

Output

This tab displays the output of the activity's data to the activities that follow in the process definition. For more details see [Input and Output](#).

Fault

Lists the activity faults or various exceptions that might occur with this activity, such as `FileNotFoundException` or `IllegalCopyException`.

Input and Output

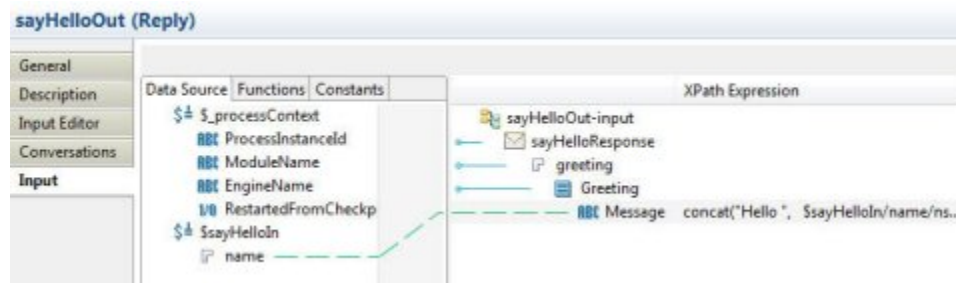
The **Input** tab is used to enter the data for an activity and the **Output** tab displays the output schema of an activity.

Configuring the Input Tab

The **Input** tab is available in the Properties view and is used to enter data for an activity. Input data for an activity can be any of the following:

- **Constant/Literal** specified using numbers or strings enclosed in quotes.
- **Regular Expression** specified using an existing schema item or by keying in a constant expression in the field.
- **Mapping** the output from previous activities to the current activity's input. Using the mapper, you can choose functions or constants from the **Functions** and **Constants** tabs with the mapped data.

Input Tab



To create a mapping:

1. Click on the desired item in the available schema in the Data Source panel. Drag the item to the desired item in the Activity Input panel.
2. To type in a constant or expression, click on the schema item in the Activity Input panel and type the constant or expression into the field.

Right-Click Menu

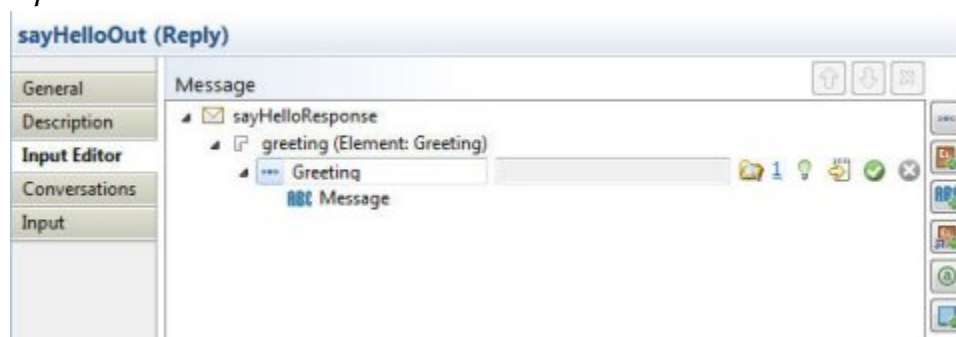
When you select an element in the Activity Input schema and right-click, a popup menu appears. The **Surround With** menu item contains several sub-items that are useful shortcuts for creating XSLT statements.

- **Surround with Choose** — a shortcut for adding a choice statement and its associated conditions or otherwise statements around the currently selected element.
- **Surround with If** — a shortcut for adding an if statement and placing the currently select element as the sub-element of the if.
- **Surround with ForEach** — a shortcut for moving the current element into a For-Each statement.
- **Surround with ForEach Group** — a shortcut for moving the current element into a For-Each-Group statement and adding a Group-By grouping statement. The `current-group()` is not provided on source side. When you create for-each binding under a for-each-group, it adds `current-group()` by default. The Grouping statement creates the list grouped by the desired element, and the `current-group()` function allows you to access the items in the requests repeating element that correspond to the group that is currently being processed.

Configuring the Input Editor Tab

Using the **Input Editor** tab you can configure the input data for an activity.

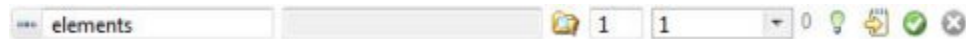
Input Editor Tab







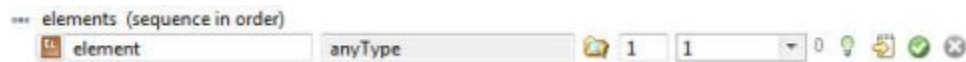
Instead of specifying a constant or an expression for the schema item, you can first configure the sequence in which this message will appear by setting up the element it is contained in.





You can define the sequence of an element using the icons on the right:

1.  **Add Complex Anonymous Type:** Adds an element sequence that is defined by the following:






- Schema type definition or creating a new type definition.
 - Number of Minimum Occurs (default is 1).
 - Number of Maximum Occurs (1 or unbounded).
 - Number of references to this resource (generated, in this case it is 0).
 - Initiate Rename Schema Element: rename the schema element by entering the New Name and choosing the option whether to update the references to this element.
 - The remaining icons are Go To , Accept Changes , and Delete , which invoke the general editing tools.
2.  **Add Complex Element:** This option adds a complex element that you can further define by the following:




- The schema type definition or a new type definition (default is anyType)
 - Number of Minimum Occurs (default is 1).
 - Number of Maximum Occurs (1 or unbounded).
 - Number of references to this resource (generated, in this case it is 0).
 - Initiate Rename Schema Element: rename the schema element by entering the New Name and choosing the option whether to update the references to this element.
 - The remaining icons are Go To , Accept Changes , and Delete , which invoke the general editing tools.
3.  **Add Primitive Element:** This option adds a primitive element that you can further define by the following:







- Choosing by the Primitive Types: String, Integer, Decimal, Boolean, Date&Time, Binary, URI or Any.
- Choosing by the Primitive Sub Types: String, Normalized String, Token, Language, Name, NC-Name, Q-Name, Name Token, Name Tokens, ID, ID ref, ID refs, Entity, and Entities.
- Number of Minimum Occurs (default is 1).

- d. Number of Maximum Occurs (1 or unbounded).
- e. Number of references to this resource (generated, in this case it is 0).
- f. **Initiate Rename Schema Element:** rename the schema element by entering the New Name and choosing the option whether to update the references to this element.
- g. The remaining icons are **Go To** , **Accept Changes** , and **Delete** , which invoke the general editing tools.




4.  **Add Reference Element:** This option adds a reference element that you can further define by the following:



- a. The schema type definition or a new type definition.
- b. Specifying the Minimum Occurs number (default is 0).
- c. Selecting from the drop-down list the Maximum Occurs number (1 or unbounded.)
- d. The remaining icons are **Go To** , **Accept Changes** , and **Delete** , which invoke the general editing tools.

5.  **Add Attribute:** This option adds an attribute that you can further define by the following:



- a. Choosing by the Primitive Types: String, Integer, Decimal, Boolean, Date&Time, Binary, URI or Any.
- b. Choosing by the Primitive Sub Types: String, Normalized String, Token, Language, Name, NC-Name, Q-Name, Name Token, Name Tokens, ID, ID ref, ID refs, Entity, and Entities.
- c. Use Optional/Required (default is Optional).
- d. The remaining icons are **Go To** , **Accept Changes** , and **Delete** , which invoke the general editing tools.

6.  **Add Any Element:** This option adds an element that you can further define by the following:



- a. Wildcard Namespace (a space-delimited list of the namespaces can be entered).
- b. Entering the Minimum Occurs number (default is 0).
- c. Selecting from the drop-down list the Maximum Occurs number (1 or unbounded.)

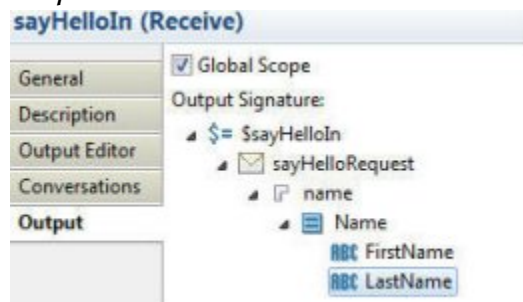
- d. The remaining icons are **Go To** , **Accept Changes** , and **Delete** , which invoke the general editing tools.

Viewing the Output Tab

The **Output** tab is available in the Properties view and is used to display the activity output schema. The output of an activity is displayed for informational purposes only and cannot be modified or altered.

The output tab displays the activity output schema. This name appears in subsequent activities input tabs. The activity output data is displayed for informational purposes only and cannot be modified or altered.

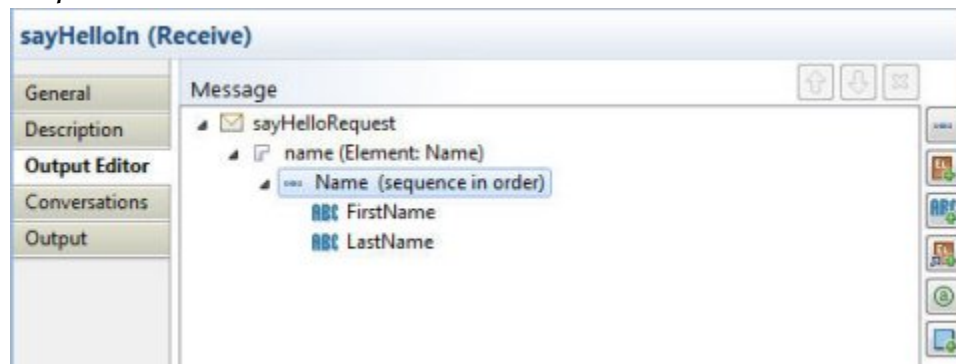
Output Tab



Configuring the Output Editor Tab

Input Editor allows for GUI based approach in configuring the output data.

Output Editor Tab



Using the icons on the right, additionally define the Name in element. The icons have same meaning as when used for the Input Editor.

Creating a Module Property

Module properties can be used to define configuration for shared resources, policy resources, and activities. Activities can use process properties or module properties. When a module property is referenced directly in an activity, a new process property is created automatically and mapped to module property with same name as the module property.

To create a module property, follow these steps:

Procedure

1. Expand the module in **Project Explorer**.

2. Expand **Module Descriptors**.
3. Double-click **Module Properties**.
This opens the Module Properties page in the right pane.
4. Click **New Property** to create a new module property.
5. Edit the name of the property by clicking its default name in the **Groups/Properties** column.
6. Optional. Change the property type by clicking in its **Type** column and selecting a type from the drop-down list.
7. Enter a value for the property by clicking in its **default** column.
You can organize the related properties into various groups. To create a group, click **New Group** and then move the property under the group using the **Move Up** or **Move Down** buttons.

Editing a Module Property

You can edit a module property from the **Module Properties** page or override its value from the **Properties** page that is accessed from the application. Before you edit a module property, keep in mind that the change will be propagated to any activity, binding, or shared resource that uses or references the property.

To edit the value of a module property, do the follow these steps:

Procedure


1. In **Project Explorer**, expand the application module completely and double-click **Module Properties** to open the Module Properties page in the right pane. Alternatively, to open the Properties page, expand the application completely and double-click **Properties**.
2. Double-click in its **[default]** column and enter a new value or edit the existing value.
3. Save the application.
The value of the property changes color from black to blue. If you edit the module property value in the Properties page accessed from the application, it will override the original value of the property that was defined in the Module Properties page.


Promoting Module Properties for Visibility at the Application Level

Module properties in an application module are visible and applicable only to the module in which they were created. They are internal to the application module and are not available at the application level.

To promote a module property to the application level, do the following:

Procedure

1. In **Project Explorer**, fully expand the application and double-click **Properties** under **Package Unit**. This opens the Properties editor in the right pane which displays both application properties as well as the module properties.
2. Expand the application module name, select the module property, and double-click its value column (**[default]**) to reveal the edit buttons.
3. Click the  button to promote the module property to the application level and save the application.
The property should appear under **Application**. Also, the value of the original module property will change color to blue and enclosed in %% indicating that the property has been promoted. The property will now be visible at the application level.

To revert the promotion, double-click the **[default]** column and click the  button. This removes the mapping of the promoted application property from its source module property but will not delete the promoted property appearing under **Application**. Make sure to manually delete the promoted property under **Application** in order to clean up unwanted properties.

If you edit a module property on this page, its color changes to blue indicating that the value of this property was overridden.

Deleting a Promoted Property


To delete a promoted property, select it under **Application** and click **Delete**.

Importing WSDLs

Follow these steps to import WSDL files from the internet into TIBCO Business Studio for BusinessWorks.

Procedure

1. Right-click the **Service Descriptors** folder, and select **Import > Import WSDL from URL**
2. Enter the URL of the WSDL in the **Resource URL** field.
The **Import Location** field is automatically populated with the import location of the WSDL and XSD files being imported. By default, WSDL file are imported to the **Service Descriptors** folder and XSD files are imported to the **Schemas** folder. You can update these import locations in the wizard.

 If you enter a remote WSDL URL, and the WSDL contains dependencies, these dependencies will be listed in the **Dependencies** section of the Import WSDL from URL wizard.
3. Optional. Unselect the **Update import location attribute and include location attribute to reference WSDL and XSD files imported locally** check box if you do not the import location to be automatically updated with the relative locations of corresponding and already imported WSDL and XSD files.

Using Additional Features

Complex business processes make use of additional features such as process scopes, fault handlers, conversations, checkpoints, and so on.

The sections that follow describe how to use the specified feature when developing a process.


Using Scopes

A scope is a group without any conditions that is used to encapsulate activities and variables from the outer scope.

Prerequisites

Select the activities you want to add to a Scope.

Procedure

1. Right-click on the selection and select **Create Group > Scope** .

The selected activities will be encapsulated in a new scope.

2. Configure the new scope from the Properties view.

- a) **General** tab

- **Name:** Specify a name for the scope.
- **Group Type:** Default is set to **Scope**, which is a group of activities without any conditions. Change the group type to create a group with conditions.

- b) **Description** tab

- **Description:** Enter a description for the new scope.

- c) **Variables** tab

You can add local variables to the group from the **Variables** tab. See [Adding Scope Variables](#) for details on adding variables.


A scope variable can override a process variable if they have the same name. Use the **Assign** activity to override a process variable with the scope variable.

Adding Scope Variables

A scope variable saves the state within the scope.

To add scope variables, select the scope in the Process Editor and then select the **Variables** tab on the Properties view.

Adding a Complex Type Variable

Click the icon  **Add complex type Variable** and select an existing schema or create a new schema to be added from the Select Schema Element Declaration dialog box.



Select Schema Element Declaration





Field/Action	Description
Workspace	When selected, the variable is valid only during the design-time.

Field/Action	Description
Current and Dependent Modules	When selected, the variable is valid for the current module and the modules that are dependent on it.
Current Module	When selected, the variable is restricted to the current module.
Display all XSD Elements	Select the check box to display all the XSD elements in the module. This check box is selected by default.
Include Process Inline Schemas	Select the check box to display the process inline schemas in the module.
Include WSDL Inline Schemas	Select the check box to display the WSDL inline schemas in the module.

If you chose an existing schema, click **OK** to select it. If you choose to create a new schema, click **Create New Schema** to create a new XML schema.


Create XML Schema

Field/Action	Description
Resource Name	Specify a name for the new schema.
Workspace Location	Specify a location to store the new schema. The wizard displays the default location for the particular module. You can choose to keep the default or browse to select a different location.
Choose a Root Element	<p>Add a primitive element to the new schema using the icon Add Primitive Element .</p> <p>The new primitive element will appear listed under the root element. Double-click the element to configure it.</p> 
Primitive Types	<p>Select the primitive type for the element from the drop-down list:</p> <ul style="list-style-type: none"> • String • Integer • Decimal • Boolean • Date & Time • Binary • URI • Any

Field/Action	Description
Subtypes	Select the subtypes for the element from the drop-down list: <ul style="list-style-type: none"> • String • Normalized String • Token • Language • Name • NC-Name • Q-Name • Name Token • Name Tokens • ID • ID ref • ID refs • Entity • Entities
Number of references to this resource	Displays the number of references to this resource.
 Initiate Element Rename Refactoring	Use to rename the schema element. You can choose to preview and update all references to the element.
 Accept Changes	Accept the changes entered for the new schema element.
 Cancel Changes	Cancel the changes accepted for the new schema element.
Remove Selected Element 	Any of the elements added to the schema can be deleted using this option.

Click **OK** when you are done editing the XML schema.

Adding a Simple Type Variable

Add a simple variable by clicking the icon  **Add simple type Variable**. Select the variable type from the drop-down list and specify a default value.

Variable Type	Default Value
String	None.
Integer	1
Decimal	1

Variable Type	Default Value
Boolean	true (You can select false from the drop-down list.)
Date & Time	None. Enter a date and time.
XSD Element	To select an XSD element, follow the instructions provided in Adding Scope Variables

Defining and Using Shared Variables

Shared variables are defined at a module level.

Defining a Shared Variable

Procedure




1. In the Project Explorer view, double-click **Shared Variables** under the **Module Descriptors** to open the **Shared Variables** tab.





Two panes are displayed:

- Module Shared Variables
- Job Shared Variables

For more information, see "Shared Variables" section in the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide.


2. Click one of the following icons in the respective sections to define a module shared variable or a job shared variable:
 -  - Add a complex element. You can choose from an existing schema declaration or create a new schema.
 -  - Add a simple element.
 -  - Select the Shared Module where the reference to the shared variable needs to be updated.
3. In the Properties view, provide the information as described in the following table.

Tab	Field Name	Description
General	Variable Name	Name of the shared variable
	Type	<p>After adding a complex or simple element, specify the Module Data type for the shared variable to use by selecting one of the following options from the drop-down menu:</p> <ul style="list-style-type: none"> • String • Integer • Boolean • Date&Time • Complex Element..
	Persistent	<p>By default, the value of a module shared variable is stored in memory and the current state of the module shared variable would be lost in case the engine (or the AppNode) crashes.</p> <p>Select the check box to persist the current value of the module shared variable. The current state of the variable in the engine's persistent storage is only updated when the value of the variable changes. Also, a persistent module shared variable can be made visible across AppNodes in an AppSpace when the engine persistent mode is set to "group".</p> <div style="display: flex; align-items: flex-start;"> <div style="margin-right: 10px;"></div> <div style="border-left: 1px solid #ccc; padding-left: 10px;"> <p>The engine persistence must be configured for the current value of the module shared variable to persist.</p> </div> </div> <div style="display: flex; align-items: flex-start; margin-top: 10px;"> <div style="margin-right: 10px;"></div> <div style="border-left: 1px solid #ccc; padding-left: 10px;"> <p>This check box only displays when configuring module shared variables. Job shared variables cannot be configured to be persistent.</p> </div> </div>
Description	Description	Description for the shared variable.

Tab	Field Name	Description
Initial Value	Initial Value	<p>Enter an initial value for the shared variable. Select one from the following options:</p> <ul style="list-style-type: none"> • None: Specifies that no initial value is set for the shared variable. Ensure that you set the value using the Set Shared Variable activity in the business process before you retrieve the value of the variable using the Get Shared Variable activity. • Select Value: Select this option to browse and select a file containing the initial value for the shared variable. • Build Value: Select this option to enter an initial value for the shared variable.

Retrieving and Assigning a Value of a Shared Variable

To retrieve the value of a shared variable, use the  **Get Shared Variable** activity in the General

Activities palette. To assign a value to a shared variable, use the  **Set Shared Variable** activity in the General Activities palette.

Working with Critical Section Groups

Critical Section groups and shared locks can be used to synchronize access to shared variables.

A Critical Section group allows only one process instance to execute the Critical Section group and its contents at a time. Use a Critical Section group to contain the activities that access the shared variables, Set Shared Variable and Get Shared Variable. Once a process instance begins executing a Critical Section group, other concurrently running process instances that are associated with that Critical Section group wait at the start of the group until the currently running process instance exits the Critical Section group. This ensures that the value of the shared variable is not modified while another process instance is accessing it. See *Bindings and Palettes Reference > Basic Activities Palette > Critical Section* for more information about using Critical Section groups and shared locks.

Best Practices

Critical section groups cause multiple process instances to wait for one process instance to execute the activities in the group. As a result, there may be performance implications when using these groups. When creating critical section groups, use the following guidelines to avoid potential performance issues:

- Keep the duration of a Critical Section group as short as possible. That is, put only a very few activities in a Critical Section group, and only use activities that execute very quickly.
- Avoid nesting Critical Section groups. If you must use nesting, ensure that Lock shared configuration resources are used in the same order in all process definitions. Deadlocks can occur if you do not specify the Lock resources in the same order in nested Critical Section groups for all process definitions.

- Do not include any activities that wait for incoming events or have long durations, such as Request/Reply activities, Wait For, Sleep, or other activities that require a long time to execute.

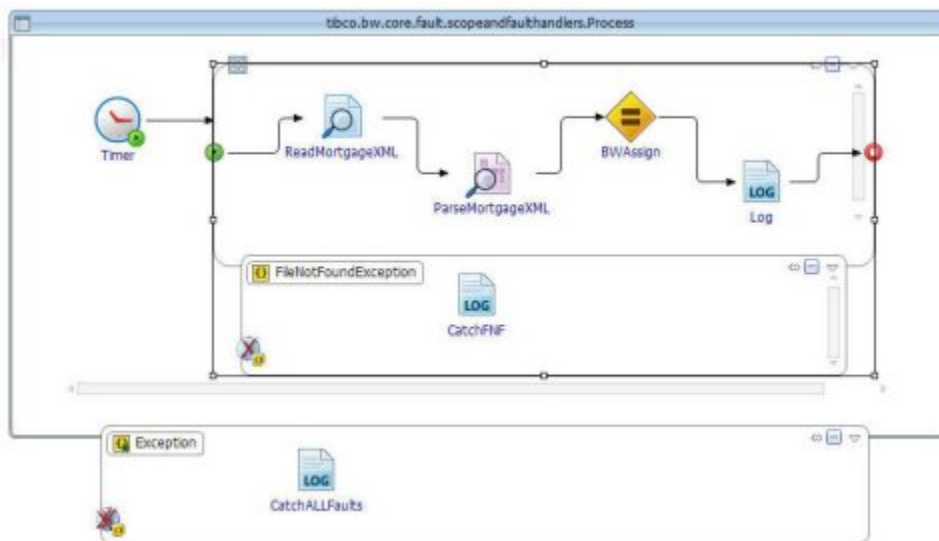
Using Fault Handlers

Fault handlers are used to catch faults or exceptions and create fault-handling procedures to deal with potential errors.

Fault handlers are defined at the scope level allowing you to catch faults or exceptions thrown by activities within a scope. There are two types of fault handlers: **Catch Specific Fault** and **Catch All Faults**.

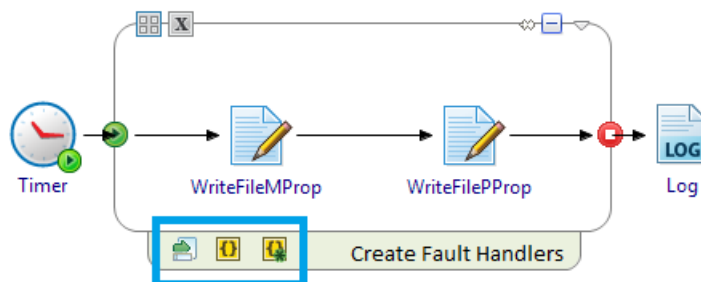
Fault handlers can be defined at the process level, or at a scope level within a process. The diagram below shows two fault handlers - one defined at the process level and the other defined at an inner scope level.

Fault Handler Attached to an Inner Scope





Procedure

- Select the activities inside the process where the exception is expected to occur and select **Create Scope > Scope** from the right-click menu.
- Move the cursor right underneath the scope's lower border to view the icons to create fault handlers.



- Click one the following:

- Create Catch  to create a fault handler for a specific exception.
- Create Catch All  to create a fault handler to catch all exceptions.

A new fault handler is added under the scope.

4. Add activities and configure the fault handling procedure inside the fault handler area. For example, add a **Log** activity inside the fault handler area to record messages from the exception.

Using Conversations

Conversations are used for stateful business processes, which means that for completion, processes require correlation of message exchanges. Such processes can be reentrant and so the previous process context is maintained for continuity.

Conversations are always initiated by one activity and joined by other activities. All operations that are part of the stateful process must generate a conversation ID and reply to the original client that contains the conversation ID.

For example, an operation Submit Purchase Order in a stateful process gets the Purchase Order ID in response. If the client wishes to cancel the purchase order, the client must use this correlation ID (Purchase Order ID) to invoke the Cancel Purchase Order operation.

Building a Conversation

Procedure

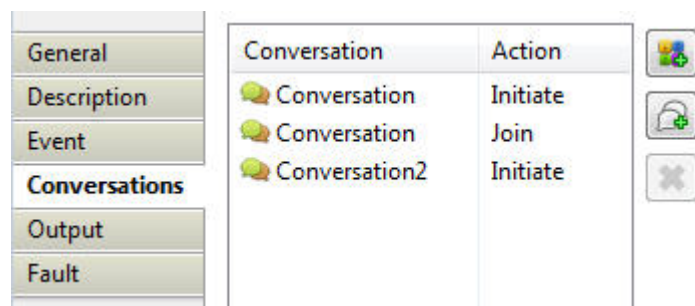
1. Right-click the activity that needs to initiate the conversation and select **Conversation > Create New Conversation**.

The **Conversations** tab in the Properties view displays the conversation name and action 'Initiate'.

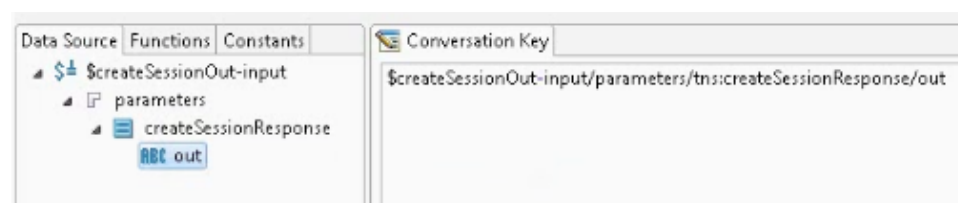
2. Right-click the activity that needs to join the conversation and select **Conversation > Join Conversation > Conversation_Name**.

The **Conversations** tab in the Properties view displays the conversation name and action 'Join'.

The **Conversations** tab of any activity that participates in conversations lists all the conversations it is participating in.



3. Click on the conversation name to specify the key data.



The initiating key is returned as a part of the response, and the client must provide the same key when calling a related operation the next time. This ensures that the first and second operations are called by the same client and the two operations are part of the same conversation.

Using Checkpoints

A **Checkpoint** activity saves the state and data of a process instance so that it can be recovered at a later time, in case of a failure.

If an ActiveMatrix BusinessWorks engine fails, all processes that have a **Checkpoint** activity can be recovered and resume execution from the last checkpoint executed in the process instance.

Only the most recent state is saved by a checkpoint. If you have multiple checkpoints in a process, only the state from the last executed checkpoint is available for recovering the process.

Checkpoints save the state of the entire process instance. A process (parent process) can call another process (sub-process) in two ways: in-line and non-inline. An in-line sub-process is executed as part of the parent process job, while the non-inline sub-process spawns a new job. When a **Checkpoint** activity is executed in an in-line sub-process, the checkpoint saves the state of the current process instance, including the state of the parent processes. However, when a checkpoint occurs in a non-in-line sub-process, the checkpoint saves the state of the spawned process instance only.

Checkpoints require the engine persistence mode to be either **datastore** or **group**. To configure the persistence modes, see:

- For Datastore : [Configuring Database for the Engine](#)
- For Group : [Configuring Database for the Engine](#) and [Configuring the Engine for Group Persistence Mode](#)

If the engine persistence mode is not configured with the correct value, an application with **Checkpoint** activity encounters an error at deployment.

Recovering After a Failure

Following a crash, when the engine is restarted, the process instances are recovered from the last checkpoint automatically. That is, all process instances that were check pointed will continue processing from the last executed **Checkpoint** activity.

Ensure that the process has all of the data required to continue processing following a recovery. When placing your checkpoint in a process, be careful with certain types of process starters or incoming events, so that a recovered process instance does not attempt to access resources that no longer exist. For example, consider a process with an HTTP process starter that takes a checkpoint after receiving a request but before sending a response. In this case, when the engine restarts after a crash, the recovered process instance cannot respond to the request since the HTTP socket is already closed. As a best practice, place the response activity before the checkpoint so that any response is sent before a checkpoint is taken. Also, do not place the **Checkpoint** activity in a critical section or an event handler.



Test checkpoints in your applications through the Admin UI or with bwadmin.



Using Coercions


In some scenarios, the datatype of a Data Source element might be undefined. If you know the datatype of an element, you can coerce the element into a specific type using the **Add/Edit Coercion...** option in TIBCO Business Studio for BusinessWorks. Additionally, you can use the **Add/Edit Coercion...** option to create, modify, or delete coercions for any element in the Data Source schema.

Adding a Single Coercion

To add a single coercion to an element, follow these steps.


Procedure

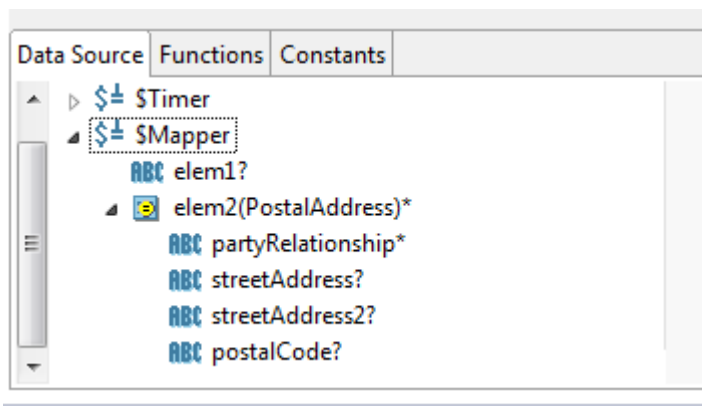
1. From the **Data Source** tab, select the element type of an element, right-click on the element and select **Add/Edit Coercion...**
2. In the Coercion window, click the  icon to add a coercion for the selected element.
3. Accept the default option for the **Component Type** field.
4. Select a schema for the **Namespace** field by choosing an option from the drop-down menu, or click the browse icon  to view a list of available schemas in the application module.
5. Click the **Type** field, to select an element type.

 Ensure that the **Type** you select is an extension of the base type.
6. Optional. Select the **Cardinality** check box, and choose one of the following options from the drop-down menu:
 - **Optional (?)**: Selecting this option sets the cardinality to zero to 1.
 - **Exactly one**: Selecting this option sets the cardinality to 1.
 - **Repeating (*)**: Selecting this option sets the cardinality to 0 to infinity.
 - **At least one (+)**: Selecting this option sets the cardinality to 1 to infinity.
7. Click **OK** to coerce the element type to be the datatype of the selected schema element.

Result

In the **Data Source** tab, the element of the selected datatype is replaced with the schema you specified. The coerced element can be mapped to any element in the Activity Input panel.



In the image below, the coerced element displays with the  icon in front of the element name, and the type you selected in parenthesis.




Adding Multiple Coercions

To add a single coercion to an element, follow these steps.

Procedure

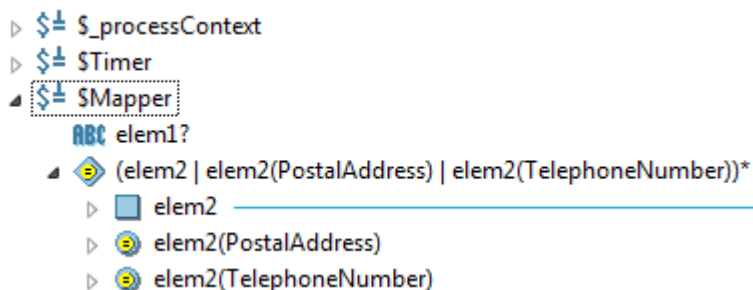
1. From the **Data Source** tab, select the element type of an element, right-click on the element and select **Add/Edit Coercion...**
2. In the Coercion window, click the  icon to add a coercion.
3. Accept the default option for the **Component Type** field.
4. Select a schema for the **Namespace** field by choosing an option from the drop-down menu, or click the browse icon  to view a list of available schemas in the application module.
5. Click the **Type** field, to select an element type.



Ensure that the **Type** you select is an extension of the base type.
6. Optional. Select the **Cardinality** check box, and choose one of the following options from the drop-down menu:
 - **Optional (?)**: Selecting this option sets the cardinality to zero to 1.
 - **Exactly one**: Selecting this option sets the cardinality to 1.
 - **Repeating (*)**: Selecting this option sets the cardinality to 0 to infinity.
 - **At least one (+)**: Selecting this option sets the cardinality to 1 to infinity.
7. Click **OK**.

Result

In the **Data Source** tab, the coerced element displays two types. Either type can be mapped to an element in the Activity Input panel.




Coercing a Specific Data Type

Use the **Substitution...** option to coerce an element type. This is useful if you want to specify that the input data use a specific datatype. Element, Type, Model group, and Attribute can be substituted.

Procedure

1. Select an element on the right side of the mapper, and select the **Substitution...** option.
2. Configure the **Component Type** field by selecting one of the following options:

- **Element:** The element, if not an AnyElement, can only be substituted by other members in its substitution group.
 - **Type:** An AnyType or abstract type can also be substituted by other types.
 - **Model Group:** Select this option to insert the contents of a selected model group into the mapper tree. The selected element in the Activity Input Schema is replaced by the contents of the model group you select.
 - **Attribute:** Select this option to coerce an attribute to the anyAttribute type. This option is useful if you are using attributes not specified in the schema.
3. Select a schema for the **Namespace** field by choosing an option from the drop-down menu, or click the browse icon  to view a list of available schemas in the application module.
 4. Click the **Type** field, to select an element type.



Ensure that the **Type** you select is an extension of the base type or within the same substitution group.

Result

After the substitution, the corresponding data type becomes the coerced one.

Editing Coercions

To edit a coerced element, follow these steps.

Procedure

1. From the **Data Source** tab, select the element type of an element, right-click on the element and select **Add/Edit Coercion...**
2. In the Coercion window, select the Type to coerce the element to use.
3. Modify the **ComponentType**, **Namespace**, **Type** and **Cardinality** fields as needed.
4. Click **OK**.


Result

In the **Data Source** tab, the coerced element displays the new type you selected.

Removing Coercions


You can remove individual coercions from an element, or you can remove all coercions and return the element to its original state.

Removing Individual Coercions

1. From the **Data Source** tab, select the element type of an element, right-click on the element and select **Add/Edit Coercion...**
2. In the Coercion window, select the Type to remove, and click the  icon.

In the **Data Source** tab, the element no longer displays the type you removed.

Removing All Coercions

To remove all coercions from an element, select the **Add/Edit Coercion...** option, and click the  icon in the Coercion window. Optionally, you can right-click on the coerced element, and select **Remove**

Coercion to remove all coercions. Once all coercions are removed from the selected element, and the element returns to its original state.

Configuring Database for the Engine

Checkpoint activity and other persistence features require the engine persistence mode (`bw.engine.persistenceMode`) to be configured for a **datastore** or **group** mode. When the engine persistence mode property is configured for **datastore** or **group** mode, the engine requires a database configuration.

Procedure

- Scripts for creating the engine database for various database types are located at `BW_HOME/config/dbscripts/engine`. Based on whether the engine persistence mode property is configured for **datastore** mode or **group** mode, complete one of the following steps:
 - If the engine persistence mode property is set to **datastore** mode, run the bundled scripts `create.sql` and `create-scp.sql` to create the engine database.
 - If the engine persistence mode property is set to **group** mode, run the bundled scripts `create.sql` and `create-dcp.sql` to create the engine database.
- To change the engine persistence mode, run the utility to set the persistence mode property `bw.engine.persistenceMode` to `datastore` or `group`, and then configure the engine database connection details.

```
bw.engine.persistenceMode=[datastore | group]
```



Before updating the AppSpace configuration, you must stop the AppSpace if it is running.

The database connection configuration can be specified at the AppSpace or the AppNode level. The database connection details specified at the AppSpace level will apply to all AppNodes within the AppSpace. The configuration specified at the AppNode level takes precedence over the configuration specified at the AppSpace level.

When the engine persistence mode property is set to `group`, the database connection configuration must be specified only at the AppSpace level.

When the engine persistence mode property is set to `datastore`, the database connection configuration cannot be shared by two or more AppNodes in the same AppSpace. As a result, the database connection configuration can be specified at the AppSpace level only if the AppSpace contains a single AppNode. For an AppSpace that contains two or more AppNodes, each AppNode requires a unique database and the database connection configuration must be specified at the AppNode level.

- To set database configuration properties at the AppSpace level, follow these steps:



Ensure you are using a different database instance for each AppSpace. To do this with a single database, create a tablespace or schema for each AppSpace.

- Copy the existing AppSpace `config.ini` file (located in the root of the AppSpace folder), or the AppSpace `config.ini` template file `appspace_config.ini_template` (located in `BW_HOME/config/`) to a temporary location.
 - Edit the engine persistence mode property, `bw.engine.persistenceMode`, and set it to `datastore` or `group`.
- ```
bw.engine.persistenceMode=[datastore | group]
```
- Configure the following database connection properties in the **BW Engine datastore configuration** section of the `config.ini` file:

```
#-----
Section: BW Engine Database Configuration.
#
The properties in this section are applicable to the BW Engine database.
All properties in this section are required when the BW Engine
```

```
property "bw.engine.persistenceMode" set to "datastore" or "group".
#

BW Engine Database Driver.
bw.engine.db.jdbcDriver=org.postgresql.Driver

BW Engine Database URL.
bw.engine.db.url=jdbc:postgresql://<servername>:<portnumber>/<dbname>

BW Engine Database User Name.
bw.engine.db.userName=user1

BW Engine Database User Password.
bw.engine.db.password=

BW Engine Database Connection Pool Size.
bw.engine.db.maxConnections=15
```

When setting the password property (**bw.engine.db.password**), the default format is plain text. Execute the command **bwadmin obfuscate**, or the command **bwobfuscator**, from the command line to encrypt the password; use the generated encrypted text as the password.




The **bwadmin bwenginedb** command will display BW engine datastore configuration settings.

4. To set the database for datastore mode at the AppNode level, follow these steps:
  - a) Copy the existing AppNode `config.ini` file (located in the root of the AppNode folder) to a temporary location.
  - b) Set engine persistence mode property **bw.engine.persistenceMode** to `datastore` and configure engine database connection details.
 

```
bw.engine.persistenceMode=[datastore]
```
  - c) Configure the engine database connection properties in the **BW Engine datastore configuration** section of the `config.ini` file. By default, the AppNode `config.ini` file does not contain these properties. Copy these properties from the AppSpace `config.ini` template file, `appspace_config.ini_template`, located in `BW_HOME/config` to the AppNode `config.ini` file and provide the database connection details.
5. Use one of the following **config** admin commands to push the configuration to the AppSpace or the AppNode:
  - AppSpace:
 

```
bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini
```
  - AppNode:
 

```
bwadmin[admin]> config -d myDomain -a myAppSpace -n myAppnode -cf <temporaryLocation>/config.ini
```
6. Restart the AppSpace.
7.  Before you clean the engine database, ensure that you have backed up all important data.

To clean the engine database that is configured for **datastore mode**, run the `drop.sql` and `drop-scp.sql` scripts. If the engine database is configured for **group mode**, run the `drop.sql` and `drop-dcp.sql` scripts.

## Result

You used the **bwadmin** command line to set the database configuration property. You can also use the Admin UI to set this property. See the following topics from the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

- Editing an AppSpace Configuration

- Editing an AppNode Configuration

## Configuring the Engine for Group Persistence Mode

The managed fault tolerance feature requires the engine persistence mode to be configured for the group mode. The group mode also supports the Checkpoint activity and other persistence features. When configured for the group persistence mode, the engine requires both a database and a group provider, such as TIBCO Enterprise Message Service™ (EMS) or TIBCO FTL®, to be configured.

Refer to the following topics for instructions about setting TIBCO EMS or TIBCO FTL as the group provider technology for the engine:

- [Configuring EMS as the Group Provider for Engine](#)
- [Configuring FTL as the Group Provider for Engine](#)

## Configuring EMS as the Group Provider for Engine

Follow these steps to configure the engine for group persistence mode, and to set TIBCO EMS as the group provider technology.

### Procedure

1. Create the engine database by executing the bundled scripts `create.sql`, `create-scp.sql` and `create-dcp.sql`. Scripts for creating the engine database for various database types are located in `BW/Home/config/dbscripts/engine`. The engine directory contains folders for the supported database types, and scripts for each database can be found in the respective folders.
2. Set engine persistence mode property (`bw.engine.persistenceMode`) to `group` and configure the engine group configuration.
  - a) Copy the existing AppSpace `config.ini` template file `appspace_config.ini_template` (located in `BW_HOME/config`) to the root of the AppSpace folder, or a temporary location, and rename the file as `config.ini`.
  - b) Edit the ActiveMatrix BusinessWorks engine persistence mode property, `bw.engine.persistenceMode`, and set it to `group`.

Follow these steps to configure the engine for group persistence mode, and to set TIBCO Enterprise Message Service™ (EMS) as the group provider technology.

```
bw.engine.persistenceMode=group
```

- c) Specify the group name and group provider technology in the `config.ini` file. The group name is optional and it defaults to domain and AppSpace names separated by an underscore (`_`). Only TIBCO Enterprise Message Service (EMS) is supported by the group provider technology.



You can use a different database instance for each AppSpace. Alternatively, you can use a single database instance for multiple AppSpaces if you create a tablespace or schema for each one.

```
#

Section: BW Engine Group Configuration.
#
The properties in this section are applicable to the BW Engine group.
Some of the properties in this section are required when the BW Engine
property "bw.engine.persistenceMode" is set to "group".
#

BW Engine Group Name. This is an optional property and it specifies name of
the BW engine group. If this property is not specified, then the group name
defaults to "Group_<DomainName>_<AppSpaceName>".
#bw.engine.groupName=mytestgroup

BW Engine Group Connection Provider Technology. This is a required
property
```

```
when the persistenceMode is set to "group"
(bw.engine.persistenceMode=group)
and it specifies the BW Engine group communication technology. The only
supported values are "ems" and "ftl". The group connection provider
technology property
requires additional configuration. See section "Configuring the Engine for
Group Persistence Mode"
for additional configuration.
bw.engine.groupProvider.technology=ems
```

d) Specify the group provider configuration:

```
#

Section: BW Engine Group Connection Provider EMS Configuration.
#
Some of the properties in this section are required when the BW Engine Group
Connection Provider Technology property
"bw.engine.groupProvider.technology"
value is set to "ems".
#

BW Engine Group Connection Provider EMS URL. This property is required if
the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSServerUrl=tcp://localhost:7222

BW Engine Group Connection Provider EMS User Name. This property is
required
if the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSUserName=admin

BW Engine Group Connection Provider EMS User Password. This property is
required if the group provider technology is "ems".
bw.engine.groupProvider.qin.EMSPassword=

BW Engine Group Connection Provider EMS Member Prefix. This property is
optional and the default value is "EMSGMS".
#bw.engine.groupProvider.qin.EMSPrefix=EMSGMS

BW Engine Group Connection Provider EMS Recovery Timeout in ms. This
property is optional and the default value is "5000" ms.
#bw.engine.groupProvider.qin.EMSRecoveryTimeout=5000

BW Engine Group Connection Provider EMS Recovery Attempt Delay in ms. This
property is optional and the default value is "500" ms.
#bw.engine.groupProvider.qin.EMSRecoveryAttemptDelay=500

BW Engine Group Connection Provider EMS Recovery AttemptCount. This
property is optional.
#bw.engine.groupProvider.qin.EMSRecoveryAttemptCount=

BW Engine Group Connection Provider EMS Connect Attempt Count. This property
is optional.
#bw.engine.groupProvider.qin.EMSConnectAttemptCount=

BW Engine Group Connection Provider EMS Connect Attempt Delay in ms. This
property is optional.
#bw.engine.groupProvider.qin.EMSConnectAttemptDelay=
```

When setting the password property (**bw.engine.groupProvider.qin.EMSPassword**), the default format is plain text. Execute the command **bwadmin obfuscate**, or the command **bwobfuscator**, from the command line to encrypt the password; use the generated encrypted text as the password.

3. **Optional.** The following properties are available for EMS SSL configuration.

```
EMS SSL Configuration
#client identity consisting of the certificate,
#private key and optionally extra issuer certificates can be included into a
single data block using PKCS12.
#KeyStore or Entrust Store encodings
#bw.engine.groupProvider.ems.ssl.trust.identity=
```

```
#The set of Trusted Certificates represents all trusted issuers of the server
certificate.
#It must be specified by the client application unless the host certificate
verification is completely disabled.
#bw.engine.groupProvider.ems.ssl.trust.certlocation=

#EMS SSL connection trust password. This
#property is required if the JMS server protocol is ssl. The password may
#be clear text or supplied as an obfuscated string.
#bw.engine.groupProvider.ems.ssl.trust.password=

#trusted certificate commonname must match the ems server hostname if set to
false
#bw.engine.groupProvider.ems.ssl.disable.verifyHostName=

#client and server certificates must match if set to false
#bw.engine.groupProvider.ems.ssl.trust.disable.verifyHost=
```

4. **Optional.** If you have saved the `config.ini` file to a temporary location, ensure you copy it to the AppSpace root folder located in `BW_HOME/domains/defaultdomain/appspaces/defaultappspace`.
5. Use the `config` admin command to push the configuration to the AppSpace: `bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini`.

## Configuring TIBCO FTL® as the Group Provider for Engine

Follow these steps to configure the engine for group persistence mode, and to set TIBCO FTL as the group provider technology.



Use of TIBCO FTL with TIBCO ActiveMatrix BusinessWorks™ for configuring `bwagent` and for configuring group provider for engine does not require TIBCO FTL licenses.

### Prerequisites

- See the ActiveMatrix BusinessWorks™ readme for the version of TIBCO FTL that is supported with the version of ActiveMatrix BusinessWorks 6.x you are using.
- Ensure you have installed FTL client libraries. For more information, see Integrating with TIBCO FTL in the *TIBCO ActiveMatrix BusinessWorks™ Installation* guide.
- These steps are only applicable if you are not using TIBCO FTL as the `bwagent` transport.
- If you are installing TIBCO FTL after you have already installed ActiveMatrix BusinessWorks, set the `tibco.env.FTL_HOME` variable in the `bwcommon.tra` file. You can find this file in the `bin` folder at `BW_HOME\bin` for Windows, or `/${BW_HOME}/bin` for Unix.

1. Install TIBCO FTL. For instructions, see the *TIBCO FTL® Installation* guide.
2. Start the FTL Realm server by executing the `./tibrealmserver -ht <hostIP>:<port>` FTL command.

```
./tibrealmserver -ht <hostIP>:<port>
```

3. Execute the following FTL command to populate data in the `bwadmin_ftlrealmserver.json` template file, located in the `config` folder at `BW_HOME/config`:

```
./tibrealmadmin -rs <realmserverurl> -ur <PATH of bwadmin_ftl_realmserver.json>
```



For instructions about how to configure an FTL backup server for high availability, see "Configuring Backup Realm Servers for Fault Tolerance" in the *TIBCO FTL® Administration* guide.

### Procedure

1. Create the engine database by executing the bundled scripts `create.sql`, `create-scp.sql` and `create-dcp.sql`. Scripts for creating the engine database for various database types are located at



*BW\_HOME/config/dbscripts/engine*. The engine directory contains folders for the supported database types, and scripts for each database can be found in the respective folders.

2. Set engine persistence mode property (**bw.engine.persistenceMode**) to group and configure the engine group configuration.
  - a) Copy the existing AppSpace *config.ini* template file *appspace\_config.ini\_template* (located in *BW\_HOME/config*) to the root of the AppSpace folder, or a temporary location, and rename the file as *config.ini*.
  - b) Edit the ActiveMatrix BusinessWorks engine persistence mode property, **bw.engine.persistenceMode**, and set it to *group*.
 

```
bw.engine.persistenceMode=group
```
  - c) Specify the group name and group provider technology as *ftl* in the *config.ini* file. The group name is optional and it defaults to domain and AppSpace names separated by an underscore (*\_*).



Ensure you are using a different database instance for each AppSpace.

```
#

Section: BW Engine Group Configuration.
#
The properties in this section are applicable to the BW Engine group.
Some of the properties in this section are required when the BW Engine
property "bw.engine.persistenceMode" is set to "group".
#

BW Engine Group Name. This is an optional property and it specifies name of
the BW engine group. If this property is not specified, then the group name
defaults to "Group_<DomainName>_<AppSpaceName>".
#bw.engine.groupName=mytestgroup

BW Engine Group Connection Provider Technology. This is a required
property
when the persistenceMode is set to "group"
(bw.engine.persistenceMode=group)
and it specifies the BW Engine group communication technology. The only
supported values are "ems" and "ftl". The group connection provider
technology property
requires additional configuration. See section "Configuring the Engine for
Group Persistence Mode"
for additional configuration.
bw.engine.groupProvider.technology=ftl
```

- d) Specify the group provider configuration:

```
#

Section: BW Engine Group Connection Provider FTL Configuration.
#
Some of the properties in this section are required when the BW Engine Group
Connection Provider Technology property
"bw.engine.groupProvider.technology"
value is set to "ftl"
#

BW Engine Group Connection Provider FTL Realm Server. This property is
required if
the group provider technology is "ftl".
bw.engine.groupProvider.ftl.realmserver=http://localhost:8080

BW Engine Group Connection Provider FTL Realm client user name
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.username=

BW Engine Group Connection Provider FTL Realm client password
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.password=

BW Engine Group Connection Provider FTL application identifier
```

```
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.appinstance.id=bwadmin-endpoint

BW Engine Group Connection Provider FTL secondary realm server
This property is optional.
#bw.engine.groupProvider.ftl.secondaryserver=

BW Engine Group Connection Provider FTL group name
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.groupname=

BW Engine Group Connection Provider FTL application name
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.appname=bwadmin

BW Engine Group Connection Provider FTL publish endpoint
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.publish.endpoint=bwadmin-endpoint

BW Engine Group Connection Provider FTL application name
This property is required if the group provider technology is "ftl".
bw.engine.groupProvider.ftl.subscribe.endpoint=bwadmin-endpoint
```

When setting the password property (`bw.engine.groupProvider.ftl.password`), the default format is plain text. Execute the command `bwadmin obfuscate`, or the command `bwobfuscator`, from the command line to encrypt the password; use the generated encrypted text as the password.

3. **Optional.** If you have saved the `config.ini` file to a temporary location, ensure you copy it to the AppSpace root folder located in `BW_HOME/domains/defaultdomain/appspaces/defaultappspace`.
4. Use the `config` admin command to push the configuration to the AppSpace: `bwadmin[admin] > config -d myDomain -a myAppSpace -cf <temporaryLocation>/config.ini`.

## Creating Process Diagrams Explicitly

Process design diagrams are not created in EAR files generated from third-party tools. In such cases, process design diagrams can be created from TIBCO Business Studio™ for BusinessWorks™ or from the command-line interface.

### TIBCO Business Studio™ for BusinessWorks™

1. Navigate to **Windows > Preferences > BusinessWorks > Process Diagram** and select the **Enable generation of process diagrams** check box.
2. Navigate to **Project Explorer**, right-click the application name, and select the **Generate Process Diagram** option.
3. Expand your application and navigate to the **Resources** folder.

The **Resources** folder contains the **Diagrams** folder which contains the process diagrams for all the processes in the application module and all the related shared modules.

When the application is deployed, the design diagrams that are generated are included in the EAR file and can be viewed from the Admin UI.

### Command line

1. Navigate to the bin folder and open the command prompt application.
2. At the command prompt, run the following command

```
bwdesign.exe -data pathOFWorkspace For example bwdesign.exe -data
D:\BW_Temp_Wrkspce\BW6.x\V.x
```

- Run the command, `gen_diagrams` where the first argument is the name of the application and the second argument is the path where the diagram is to be exported. The second argument is optional. For example, `gen_diagrams TestingProcessDiagram.application`



If the second argument is not provided, the process design diagrams are generated in the workspace. If the argument is provided, the process diagram is created in the provided path.

- Deploy the application after the process diagram is generated.

## Displaying Individual Element Mappings

Select the **Show mapping for selected element only** check box under **Preferences > Mapper** to only display mappings for elements you select in the mapper.

In this example, the **Show mapping for selected element only** option has not been enabled. As a result, all of the input schema mappings for the **Mapper-input** Mapper activity can be seen.

| Data Source                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Functions | Constants | XPath Expression                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>▶ <code>\$_processContext</code></li> <li>▶ <code>\$Timer</code></li> <li>▶ <code>\$PreviousActivity</code> <ul style="list-style-type: none"> <li>▶ <code>NewElement</code> <ul style="list-style-type: none"> <li><code>ABC NewElement</code></li> <li><code>ABC NewElement1</code></li> <li><code>ABC NewElement2</code></li> <li>▶ <code>ComplexElement?</code> <ul style="list-style-type: none"> <li><code>0.0 Element1?</code></li> <li><code>ABC Element2</code></li> </ul> </li> </ul> </li> </ul> </li> </ul> |           |           | <ul style="list-style-type: none"> <li>▼ Mapper-input <ul style="list-style-type: none"> <li>kv <ul style="list-style-type: none"> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement1</code></li> </ul> </li> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:ComplexElement/tns:Element1</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:ComplexElement/tns:Element2</code></li> </ul> </li> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement2</code></li> </ul> </li> </ul> </li> </ul> </li></ul> |

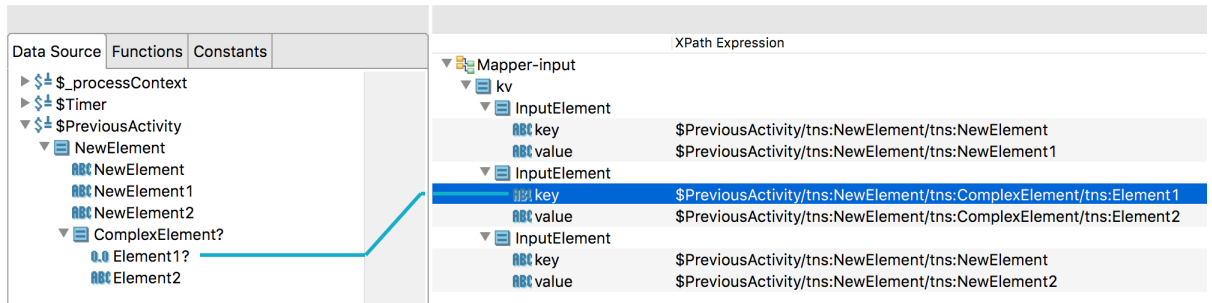
After enabling the **Show mapping for selected element only** check box, no mappings display.

| Data Source                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Functions | Constants | XPath Expression                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>▶ <code>\$_processContext</code></li> <li>▶ <code>\$Timer</code></li> <li>▶ <code>\$PreviousActivity</code> <ul style="list-style-type: none"> <li>▶ <code>NewElement</code> <ul style="list-style-type: none"> <li><code>ABC NewElement</code></li> <li><code>ABC NewElement1</code></li> <li><code>ABC NewElement2</code></li> <li>▶ <code>ComplexElement?</code> <ul style="list-style-type: none"> <li><code>0.0 Element1?</code></li> <li><code>ABC Element2</code></li> </ul> </li> </ul> </li> </ul> </li> </ul> |           |           | <ul style="list-style-type: none"> <li>▼ Mapper-input <ul style="list-style-type: none"> <li>kv <ul style="list-style-type: none"> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement1</code></li> </ul> </li> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:ComplexElement/tns:Element1</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:ComplexElement/tns:Element2</code></li> </ul> </li> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement2</code></li> </ul> </li> </ul> </li> </ul> </li></ul> |

Clicking on an element from the **Data Source** tab displays the input schema mapping for the element you selected.

| Data Source                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Functions | Constants | XPath Expression                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>▶ <code>\$_processContext</code></li> <li>▶ <code>\$Timer</code></li> <li>▶ <code>\$PreviousActivity</code> <ul style="list-style-type: none"> <li>▶ <code>NewElement</code> <ul style="list-style-type: none"> <li><code>ABC NewElement</code> (Selected)</li> <li><code>ABC NewElement1</code></li> <li><code>ABC NewElement2</code></li> <li>▶ <code>ComplexElement?</code> <ul style="list-style-type: none"> <li><code>0.0 Element1?</code></li> <li><code>ABC Element2</code></li> </ul> </li> </ul> </li> </ul> </li> </ul> |           |           | <ul style="list-style-type: none"> <li>▼ Mapper-input <ul style="list-style-type: none"> <li>kv <ul style="list-style-type: none"> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement1</code></li> </ul> </li> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:ComplexElement/tns:Element1</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:ComplexElement/tns:Element2</code></li> </ul> </li> <li>InputElement <ul style="list-style-type: none"> <li><code>ABC key</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement</code></li> <li><code>ABC value</code> → <code>\$PreviousActivity/tns:NewElement/tns:NewElement2</code></li> </ul> </li> </ul> </li> </ul> </li></ul> |

Clicking on an element on the target side shows what element it was mapped from on the source side.



## Removing Groups

Use the **Ungroup** option to remove a group. You can use this option to ungroup Local Transaction groups and groups with scopes.

## Configuring the Ungroup Preferences


Follow these steps to update the preferences for the **Ungroup** option.

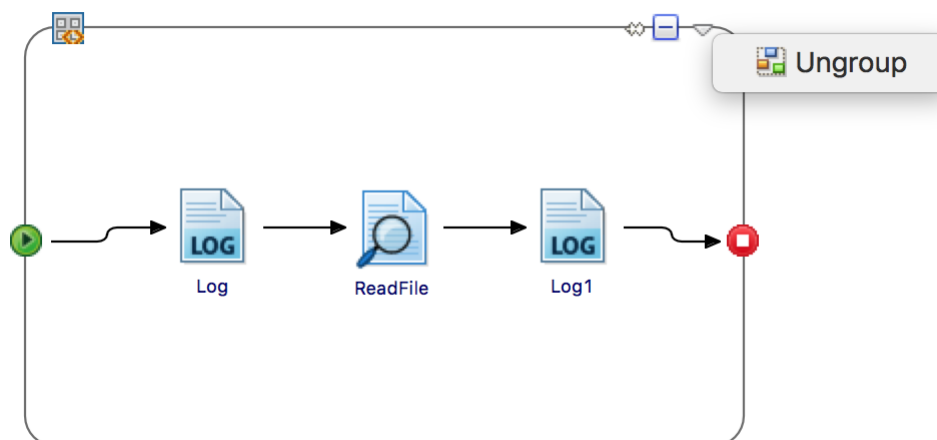
### Procedure

1. In TIBCO Business Studio™ for BusinessWorks™, click **Window > Preferences**. On Mac OS X, click **TIBCO Business Studio > Preferences**.
2. In the Preferences dialog box, click **BusinessWorks > Process Diagram**.
3. Under **Ungroup**, configure the settings for how to move activities after ungrouping groups with scopes.
4. Click **Apply**, and then click **OK**.



## Ungrouping a Local Transaction Group

Use the **Ungroup** option to remove a **Local Transaction** group.


To ungroup a **Local Transaction** group, click the  icon, and select **Ungroup**.





## Result

When the group is removed, the **GroupStart**  and **GroupEnd**  elements are deleted, and the activities move to the space that formerly contained the **Local Transaction** group. Activity transitions in the process flow remain intact, and the activities become part of the flow in the container group, or the process, it moved to.

## Ungrouping Groups with Scopes

Groups with scopes are groups that contain group variables, event handlers, fault handlers, and compensation handlers. To ungroup groups with scopes, click the  icon and select **Ungroup** option.

When the group is removed, the **GroupStart**  and **GroupEnd**  elements are deleted, and the activities move to the space that formerly contained the group. The contents of the group are re-located based on the type of container that held the group. A group with a scope can be contained within a local transaction group, a group with a scope, or a process.

### Groups with Group Variables

Group variables, which can consist of activity input variables, activity output variables, or user-defined variables, are moved out of the group to the nearest container that can be a group with a scope, or a process. Global and local variables, including group counter variables, index variables, or other variables that are part of the group, are deleted during the ungrouping process.

### Groups with Event Handlers

If a group with event handlers is contained in a group with a scope, a **Local Transaction** group, or a process, the activities and activity transitions are moved to the process flow of the container.

To ensure the activities are moved to an event handler, set the **Ungroup** preferences to **Move Event Handlers > To Event Handler of parent group**. See [Configuring the Ungroup Preferences](#) for instructions on how to do this. When this preference is selected, the following actions will occur after ungrouping a group with event handlers:

- If the container is a group with a scope, an event handler with the same configurations is created for the container, and activities are moved to the newly created event handler.
- If the container is a process, an event handler with the same configurations is created for the process, and activities are moved to the newly created event handler.
- If the container is a local transaction group, an event handler with the same configurations is created for the nearest group with a scope. If there is no nearby group, or parent group, with a scope, an event handler is created for the process. In both cases, activities are moved to the newly created event handler.

### Groups with Fault Handlers

If a group with **Catch** fault handlers, or a **Catch All** fault handler, is contained in a group with a scope, a **Local Transaction** group, or a process, the activities and activity transitions are moved to the process flow of the container group or container process.



Only one **Catch All** fault handler can exist for a group or the process, so if a group or a process already contains a **Catch All** fault handler the activities are moved to the existing **Catch All** fault handler. In other words, a new **Catch**, or a **Catch All**, fault handler is only created if a similar fault handler does not currently exist in the group or the process.

To ensure the activities in the **Catch** fault handler, or a **Catch All** fault handlers are moved to new **Catch** fault handlers, or a new **Catch All** fault handler, set the **Ungroup** preferences to **Move Catch Activities > To Catch of parent group** or **Move Catch Activities > To Catch All of parent group**. For

more information, see [Configuring the Ungroup Preferences](#) . When this preference is selected, the following actions will occur:

- If the container is a group with a scope, a **Catch**, or a **Catch All**, fault handler is created for the container, and activities in the fault handlers are moved to the newly created fault handlers.
- If the container is a process, a **Catch**, or a **Catch All**, fault handler is created for the container, and activities in the fault handlers are moved to the newly created fault handlers.
- If the container is a local transaction group, a **Catch**, or a **Catch All**, fault handler is created for the nearest group with a scope, or is created for the process. Activities in the fault handlers are moved to the newly created fault handlers.

### Groups with Compensation Handlers

If activities in a group with compensation handlers is contained in a group with a scope, a **Local Transaction** group, or a process, the activities and activity transitions in the group moved to the process flow of the container.

To ensure the activities in compensation handlers are moved into new compensation handlers, set the **Ungroup** preferences to **Move Catch Activities > To Compensation Handler of parent group**. For more information, see [Configuring the Ungroup Preferences](#) . When this preference is selected, the following actions will occur:

- If the container is a group with a scope, and the group does not have a compensation handler, a compensation handler is created for the container, and activities are moved to the compensation handler.
- If the container is a process, a compensation handler is not created for the container, and the compensation handler activities are moved to the process flow.
- If the container is a local transaction group, a compensation handler is not created for the container. Instead, a compensation handler is created for the nearest group with a scope.

# Overview of Policies

---

Policies are categorized under the following policy types:

## HTTP Security

- Basic Authentication

The Basic Authentication policy secures the HTTP layer of REST, SOAP, and pure HTTP services by validating user name and password credentials stored in HTTP headers. User name and password credentials can be authenticated against an XML File Authentication provider or an LDAP Authentication provider.

- Basic Credential Mapping

The Basic Credential Mapping policy enables authentication for specified users by automatically attaching appropriate credentials to request messages before they reach services. You can choose to enforce Fixed or Conditional credential mapping.

## SOAP Security

- WSS Provider

Configure the WSS Provider policy to enforce and validate authentication, confidentiality, integrity, and time stamping of service-side messages.

- WSS Consumer

Configure the WSS Consumer policy to enforce and validate confidentiality, integrity, time stamping, and credential mapping of response messages.

# Managing Policy Resources

---

Manage policies and policy resources from the TIBCO Business Studio for BusinessWorks.

## Creating a Folder for Policies

Policies are always stored in the **Policies** folder. The folder might not exist in projects you have imported from previous versions of ActiveMatrix BusinessWorks™ 6.x. If you create a new policy to add to an activity or binding, the **Policies** folder is automatically created. You can also create a special folder to contain policies.

To create a special folder for policies, follow these steps:

### Procedure

1. In the **Project Explorer** pane, right-click the application module and select **New > Folder** to launch the BusinessWorks Application Folder wizard.  
The Folder wizard opens.
2. Specify the following values in the New Folder window:
  - **Enter or select the parent folder:** Type the name of the parent folder, or select an existing folder to be the parent folder.
  - **Folder name:** Type **Policies**.
3. Click **Finish** to create the **Policies** folder.  
The new folder displays in the Project Explorer pane.
4. Right-click the **Policies** folder, and select **Special Folders > Use as Policies Folder**.

### Result

The folder can now store policies.

## Creating an Authentication Resource

Policies use authentication resources to verify credentials and provide appropriate credentials for users. Follow these steps to create a policy authentication resource.

### Procedure

1. In the Project Explorer, right-click the **Resources Folder**, and select a new shared resource. For example, select **New > XML Authentication** .
2. Edit the following fields:
  - **Resource Folder:** Name of the folder where the resource will be located.
  - **Package:** Name of the package in the module where the new resource is added. Accept the default package, or browse to select a different package name.
  - **Resource Name:** Name of the resource. Accept the default name, or type a new name.
3. Click **Finish**.

### Result

The authentication resource displays under the **Resources** folder in the Project Explorer.




## Associating a Policy

Enforce security on your ActiveMatrix BusinessWorks™ application, by associating a policy with an existing activity or binding.

### Associating a Policy with an Activity

1. In the Process Editor, select the activity to associate the policy with. Activities that support policies display the **Policy** tab under the **Properties** tab.
2. From the **Properties** tab, select the **Policy** tab.

3. Click the **Add Policy to Activity**  icon.

4. From the Select Policy window, perform one of the following actions:

- Click **Create a New Policy** to set up a new policy with resources. Policies you can add to the activity are listed under **Select the type of policy**.

For more information about setting up policies and resources from the policy wizard, see appropriate sections under HTTP Security and SOAP Security.

Click **Finish** to create the new policy.

- Select an existing policy under **Matching Items** and click **OK**.

The policy is associated with the activity.

### Associating a Policy with a Binding

1. In the Process Editor, select the binding to associate the policy with.
2. From the **Properties** tab, select the **Bindings** tab.
3. Click the name of the binding under the **Binding** section.
4. Click the **Bindings** tab, and select the **Policy** field from the tree.

5. Click the **Add Policy**  icon.

6. From the Select Policy window, perform one of the following actions:

- Click **Create a New Policy** to set up a new policy with resources. Policies you can add to the activity are listed under **Select the type of policy**.

For more information about setting up policies and resources from the policy wizard, see the appropriate sections under HTTP Security and SOAP Security.

Click **Finish** to create the new policy.

- Select an existing policy under **Matching Items** and click **OK**.

The policy is associated with the binding.


## Removing a Policy

Follow these steps to remove a policy from an activity or a binding.

### Removing a Policy From an Activity

1. Select the activity associated with the policy.

2. From the **Properties** tab, select the **Policy** tab.

3. Select the policy to remove, and click the **Delete the selected policy**  icon.

The policy is no longer associated with the activity.

### **Removing a Policy From a Binding**

1. Select the binding associated with the policy.

2. From the **Properties** tab, select the **Binding** tab.

3. Under the **Policies** field, select the policy to remove, and click the **Delete the selected policy**  icon.

The policy is no longer associated with the binding.

# HTTP Security

Apply security to the HTTP layer of REST, SOAP, and pure HTTP services.

## Enforcing Basic Authentication

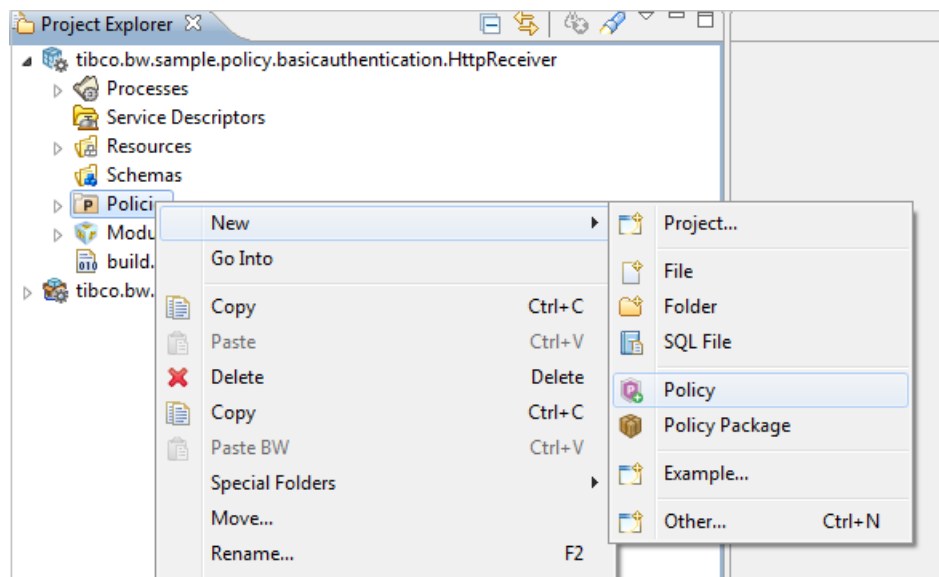
Implement the Basic Authentication policy to ensure user credentials in request messages are authenticated.

First, set up a new Basic Authentication policy by creating and configuring the policy and its resources. Next, associate the policy with an activity or binding in your application.

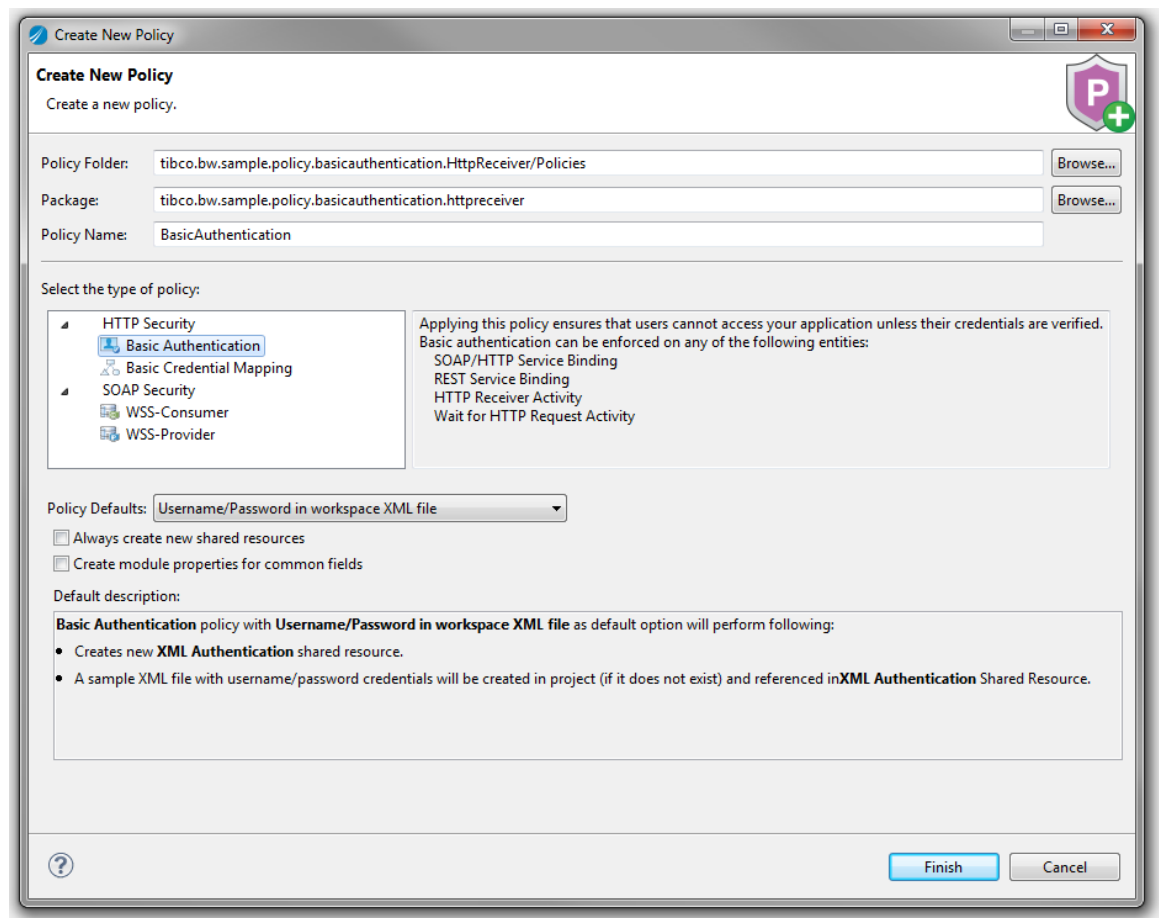
### Setting Up a Policy with Resources

Follow these steps to set a new Basic Authentication policy with resources:

1. In the **Project Explorer**, right-click the **Policies** folder and select **New > Policy**.



The Policy Wizard opens.



2. Specify the following values in the Create New Policy Window:

- **Policy Folder:** Name of the folder where policies will be located. default
- **Package:** Name of the package in the module where the new policy is added. Accept the default package, or browse to select a different package name.
- **Policy Name:** Name of the new policy. By default, the policy name is configured to match the security policy you choose. For example, if you select the Basic Authentication policy, the default name of the policy is Basic Authentication.

3. Under **Select the type of Policy**, click **Basic Authentication**.

4. From the **Policy Defaults** drop-down menu, select one of the following options:



The **Policy Defaults** menu offers a list of commonly used policy configurations to choose from. After you select a Policy Default, a policy with preconfigured settings and related resources is created. If resources already exist in the module, the newly created policy automatically refers them. However, if no resources exist, new resources with default settings are created and referred to by the policy. To view policy configurations and new resources that might be created, see the **Default description** at the bottom of the **Policy Wizard**.

- **Username/Password in workspace XML file:** Select this option to verify user credentials through an XML Authentication resource stored in your workspace. A new Basic Authentication policy configured for XML authentication and the following resources are produced in your workspace:
  - A sample XML File containing user name and password credentials with the default file name `XMLUsers.xml`

- A new XML Authentication resource with the default file name `BasicAuthentication_AuthenticationProvider.authxml`
  - **Username/Password in filesystem XML file:** Select this option to verify user credentials through an XML Authentication resource stored in your local file system. A new Basic Authentication policy configured for XML authentication is produced in your workspace:
    - A sample filesystem XML File the default file name `BasicAuthentication_AuthenticationProvider.authxml`
  - **Username/Password in LDAP:** Select this option to verify user credentials through an LDAP Authentication resource. A new Basic Authentication policy configured for LDAP authentication and the following resource is produced in your workspace:
    - A new **LDAP Authentication** resource with the default file name `BasicAuthentication_AuthenticationProvider.ldapResource`.
  - **Empty Policy (No Default) :** Select this option to create a new Basic Authentication policy with no preselected options and no resources.
5. **Optional.** Select **Always create new shared resources** to ensure new resources are generated for the policy and referred to by the policy.
  6. **Optional.** Select **Create module properties for common fields** to override default properties in newly created resources with module properties. Resources with module properties for common fields are generated after you select this option.
  7. Select **Finish** to create the policy.

### Configuring Resources and the Policy

For resource configurations, see the following topics under the "Shared Resources" topic in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide.

- XML Authentication
- LDAP Authentication

For policy configuration details, see the topic "Basic Authentication", under "Policy Resources" in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide.

### Associating the Policy with an Activity or a Binding

You can associate the Basic Authentication policy with the following activities and bindings:

- HTTP Receiver Activity
- Wait for HTTP Request Activity



Credentials authenticated on this activity are not used for propagation during credential mapping.

- SOAP Service Binding
- REST Service Binding

For instructions about how to enforce a policy on an activity, or a binding in your application, see [Associating Policies](#) topic.

## Enforcing Basic Credential Mapping

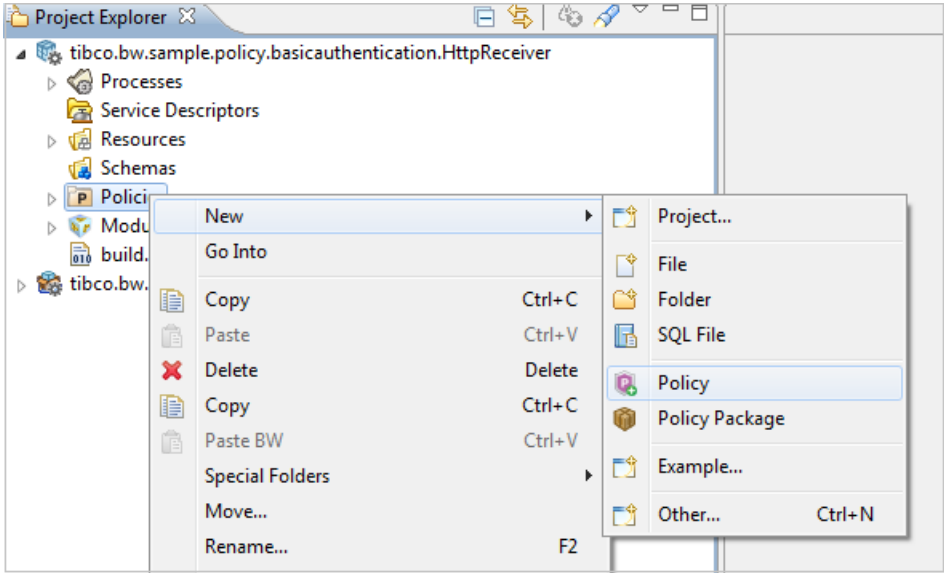
Map credentials for different types of users by enforcing the Basic Credential Mapping Policy.

First, create and configure new policy. Next, associate the policy, with an activity or a binding in your application.

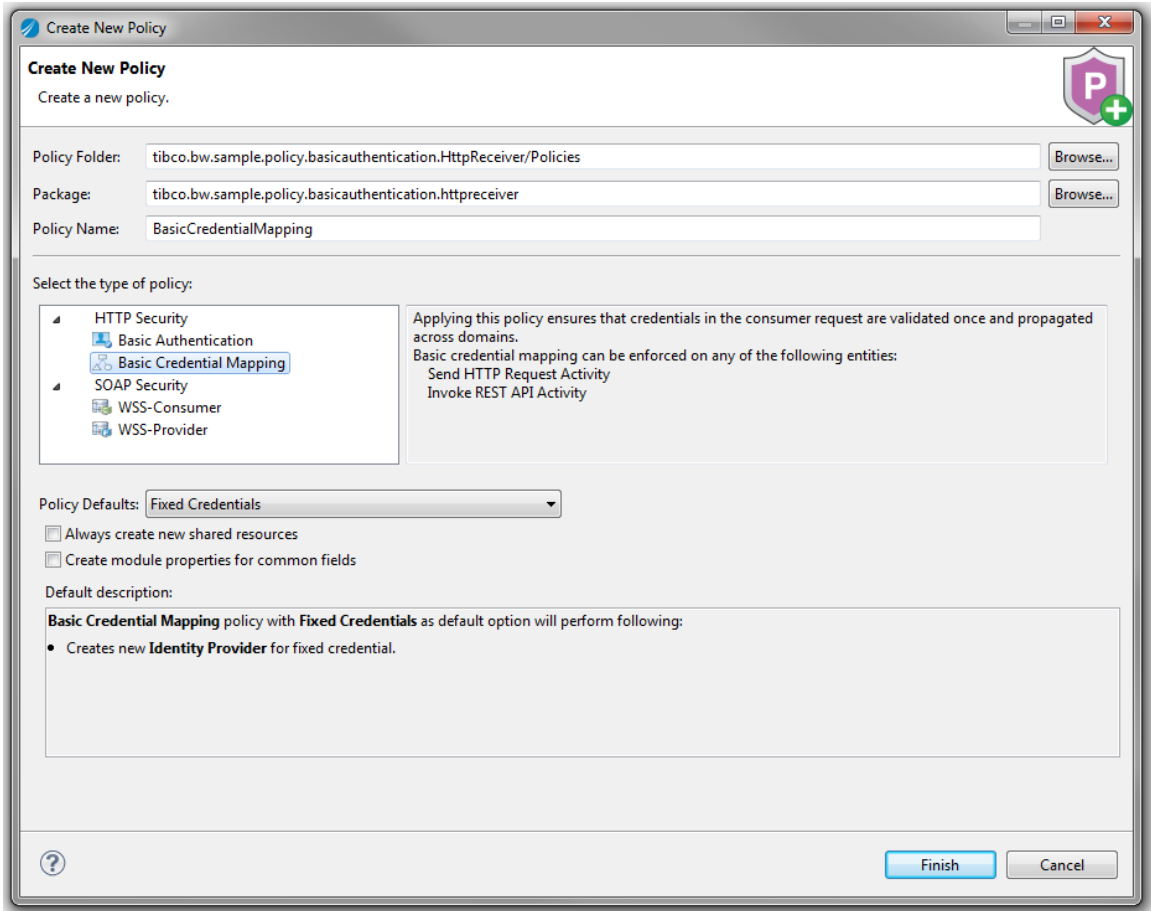
### Setting Up a Policy with Resources

Follow these steps to set up a new Basic Credential Mapping policy with resources:

1. In the **Project Explorer**, right-click the **Policies** folder and select **New > Policy**.



The Policy Wizard opens.



2. Specify the following values in the Create New Policy Window:
  - **Policy Folder:** Name of the folder where policies will be located.
  - **Package:** Name of the package in the module where the new policy is added. Accept the default package, or browse to select a different package name.
  - **Policy Name:** Name of the new policy. By default, the policy name is configured to match the security policy you choose.
3. Under **Select the type of Policy**, select **Basic Credential Mapping**.
4. From the **Policy Defaults** drop-down menu, select one of the following options:



The **Policy Defaults** menu offers a list of commonly used policy configurations to choose from. After you select a Policy Default, a policy with preconfigured settings and related resources is created. If resources already exist in the module, the newly created policy automatically refers them. However, if no resources exist, new resources with default settings are created and referred to by the policy. To view policy configurations and new resources that might be created, see the **Default description** at the bottom of the **Policy Wizard**.

- **Fixed Credentials:** Select this option to ensure a fixed set of credentials are mapped for all users. A new Basic Credential Mapping policy configured for Fixed Basic Credential Mapping and the following resource is produced in your workspace:
    - An **Identity Provider** resource with the default file name `BasicCredentialMapping_FixedIdentityProvider.userIdResource`
  - **Authenticated & Anonymous Users :** Select this option to enforce Basic Credential Mapping for authenticated users and anonymous users. A new Basic Credential Mapping policy configured for conditional basic credential mapping and the following resources are produced in your workspace:
    - An **Identity Provider** resource for authenticated users with the default file name `BasicCredentialMapping_AuthIdentityProvider.userIdResource`
    - An **Identity Provider** resource for anonymous users with the default file name `BasicCredentialMapping_AnonIdentityProvider.userIdResource`
  - **Role Based Credentials:** Select this option to enforce basic credential mapping for authenticated users with roles. A new Basic Credential Mapping policy configured for conditional basic credential mapping and the following resources are produced in your workspace:
    - An **Identity Provider** resource for authenticated users with the default file name `BasicCredentialMapping_AuthIdentityProvider.userIdResource`
    - Two separate **Identity Provider** resources for authenticated users with roles. The default file names of the resources are `BasicCredentialMapping_RoleIdentityProvider.userIdResource` and `BasicCredentialMapping_RoleIdentityProvider1.userIdResource`
  - **Empty Policy (No Default) :** Select this option to create a new Basic Authentication policy with no preselected options and no resources.
5. **Optional.** Select **Always create new shared resources** to ensure new resources are generated for the policy and referred to by the policy.
  6. **Optional.** Select **Create module properties for common fields** to override default properties in newly created resources with module properties. Resources with module properties for common fields are generated after you select this option.

7. Select **Finish** to create the policy.

### **Configuring Resources and the Policy**

For more information about resource configurations, see Identity Provider in the Shared Resources topics in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide.

For more information about policy configuration details, see Basic Credential Mapping, under Policy Resources in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide.

### **Associating the Policy with an Activity or a Binding**

You can associate the Basic Credential Mapping policy with the following activities and bindings:

- SEND HTTP Request Activity
- Invoke REST API Activity



To enforce credential mapping on a SOAP reference, apply the WSS Consumer policy and select either **SAML Token based Credential Mapping** or **Username Token based Credential Mapping**.

For instructions about enforcing a policy on an activity or binding in your application, see [Associating Policies](#).



# SOAP Security

Apply security to the SOAP layer of messages and services.

## Enforcing WSS Consumer

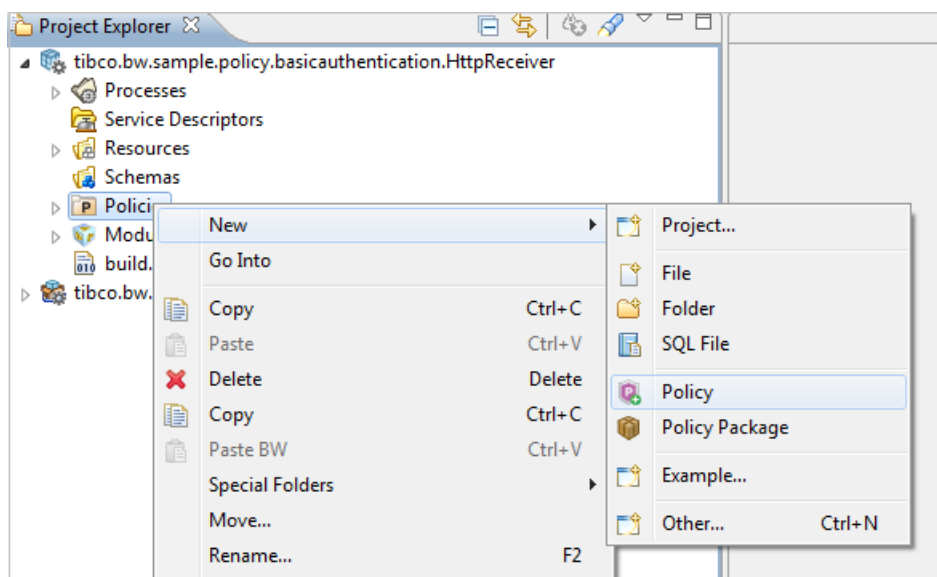
Enforce the WSS Consumer policy to ensure that the confidentiality, integrity, and the time stamp of a request remains secure.

First, create and configure the policy. Next, associate the policy with a binding in your application.

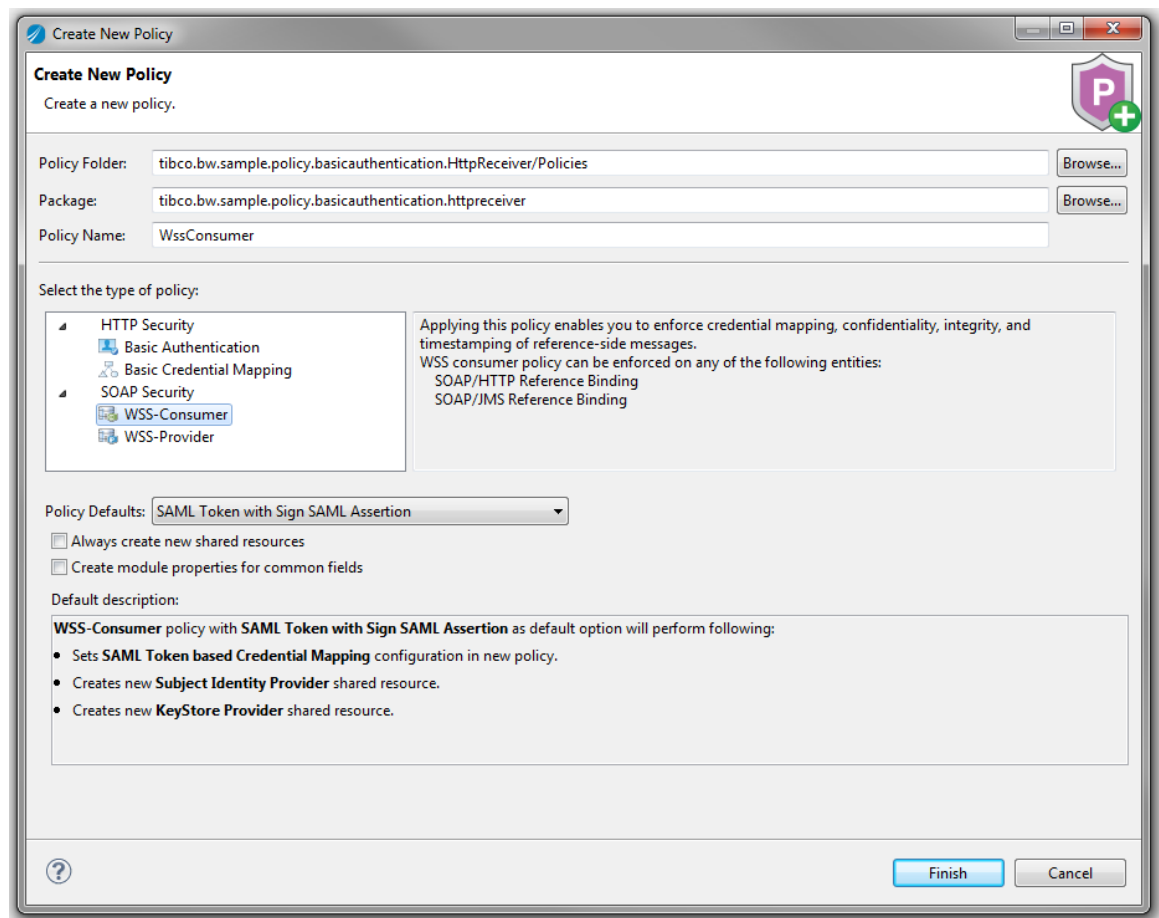
### Setting Up a Policy with Resources

Follow these steps to set up a new WSS Consumer policy with resources:

1. In the **Project Explorer**, right-click the **Policies** folder and select **New > Policy**.



The Policy Wizard opens.



2. Specify the following values in the Create New Policy Window:

- **Policy Folder:** Name of the folder where policies will be located.
- **Package:** Name of the package in the module where the new policy is added. Accept the default package, or browse to select a different package name.
- **Policy Name:** Name of the new policy. By default, the policy name is configured to match the security policy you choose.

3. Under **Select the type of Policy**, select **WSS Consumer**.

4. From the **Policy Defaults** drop-down menu, select one of the following options:



The **Policy Defaults** menu offers a list of commonly used policy configurations to choose from. After you select a Policy Default, a policy with preconfigured settings and related resources is created. If resources already exist in the module, the newly created policy automatically refers them. However, if no resources exist, new resources with default settings are created and referred to by the policy. See **default description** at the bottom of the **Policy Wizard** to view policy configurations and new resources that might be created.

- **SAML Token with Sign SAML Assertion:** Select this option to enforce SAML token-based credential mapping. A WSS Consumer policy configured for SAML token-based credential mapping and the following resources are produced in your workspace:
  - A keystore resource with the default file name `server.jks`
  - A **Keystore Provider** resource with the default file name `WssConsumer_IdentityStore.keystoreProviderResource`

- A **Subject Provider** resource with the default file name `WssConsumer_SAMLIdentityProvider.sipResource`.
  - **UserName Token with Fixed Credentials:** Select this option to enforce fixed user name token-based credential mapping. A WSS Consumer policy configured for fixed credential mapping with a user name token and the following resources are produced in your workspace:
    - A **Subject Provider** resource, with the default file name `WssConsumer_FixedIdentityProvider.userIdResource`
  - **UserName Token with Authenticated and Anonymous Credentials:** Select this option to enforce conditional user name token-based credential mapping. A WSS Consumer policy configured for conditional credential mapping with user name tokens and the following resources are produced in your workspace:
    - An **Identity Provider** resource for authenticated users, with the default file name `WssConsumer_AuthIdentityProvider.userIdResource`
    - An **Identity Provider** shared resource for anonymous users, with the default file name `WssConsumer_AnonIdentityProvider.userIdResource`
  - **UserName Token with Roles and Authenticated Credentials:** Select this option to enforce conditional user name token-based credential mapping. A WSS Consumer policy configured for conditional credential mapping with user name tokens and the following resources are produced in your workspace:
    - Two **Identity Provider** resources for authenticated users with roles, with the default file names `WssConsumer_RoleIdentityProvider.userIdResource` and `WssConsumer_RoleIdentityProvider1.userIdResource`
    - An **Identity Provider** resource for authenticated users with the default file name `WssConsumer_AuthIdentityProvider.userIdResource`
  - **Empty Policy (No Default) :** Select this option to create a new WSS Provider policy with no preselected options and no resources.
5. **Optional.** Select **Always create new shared resources** to ensure new resources are generated for the policy and referred to by the policy.
  6. **Optional.** Select **Create module properties for common fields** to override default properties in newly created resources with module properties. Resources with module properties for common fields are generated after you select this option.
  7. Select **Finish** to create the policy.

### Configuring Resources and the Policy

For more information on resource configurations, see to the following topics under Shared Resources in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide:

- Identity Provider
- Keystore Provider
- Subject Provider

For more information on policy configuration, see WSS Consumer in the Policy Resources section of the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide.

### Associating the Policy with a Binding

You can associate the WSS Consumer policy with the following bindings:

- SOAP-HTTP Reference Binding

- SOAP-JMS Reference Binding

For instructions about how to enforce a policy on a binding in your application, see [Associating Policies](#).

## Enforcing WSS Provider

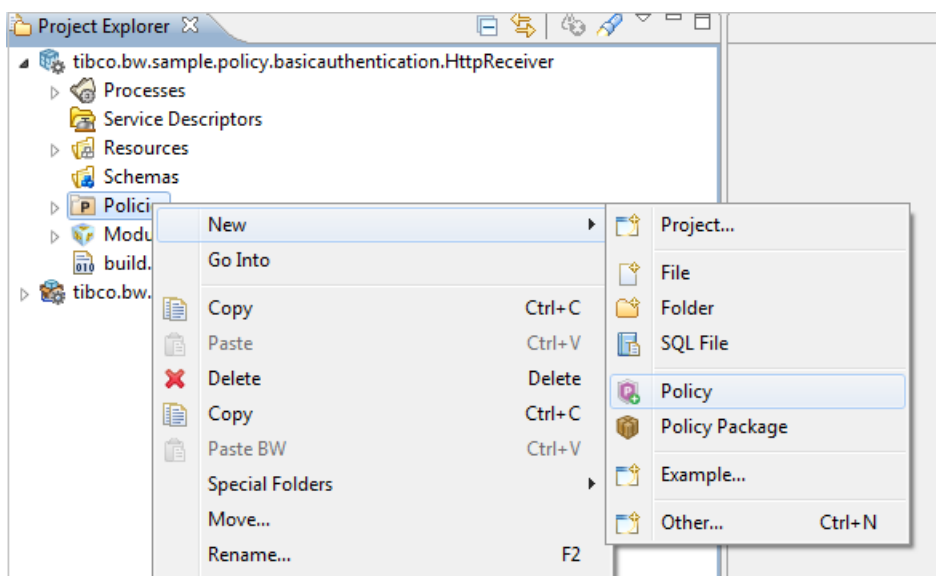
Use the WSS Provider policy to enforce authentication, confidentiality, integrity, and the time stamping of service-side messages.

First, create and configure the policy. Next, associate the policy with a binding in your application.

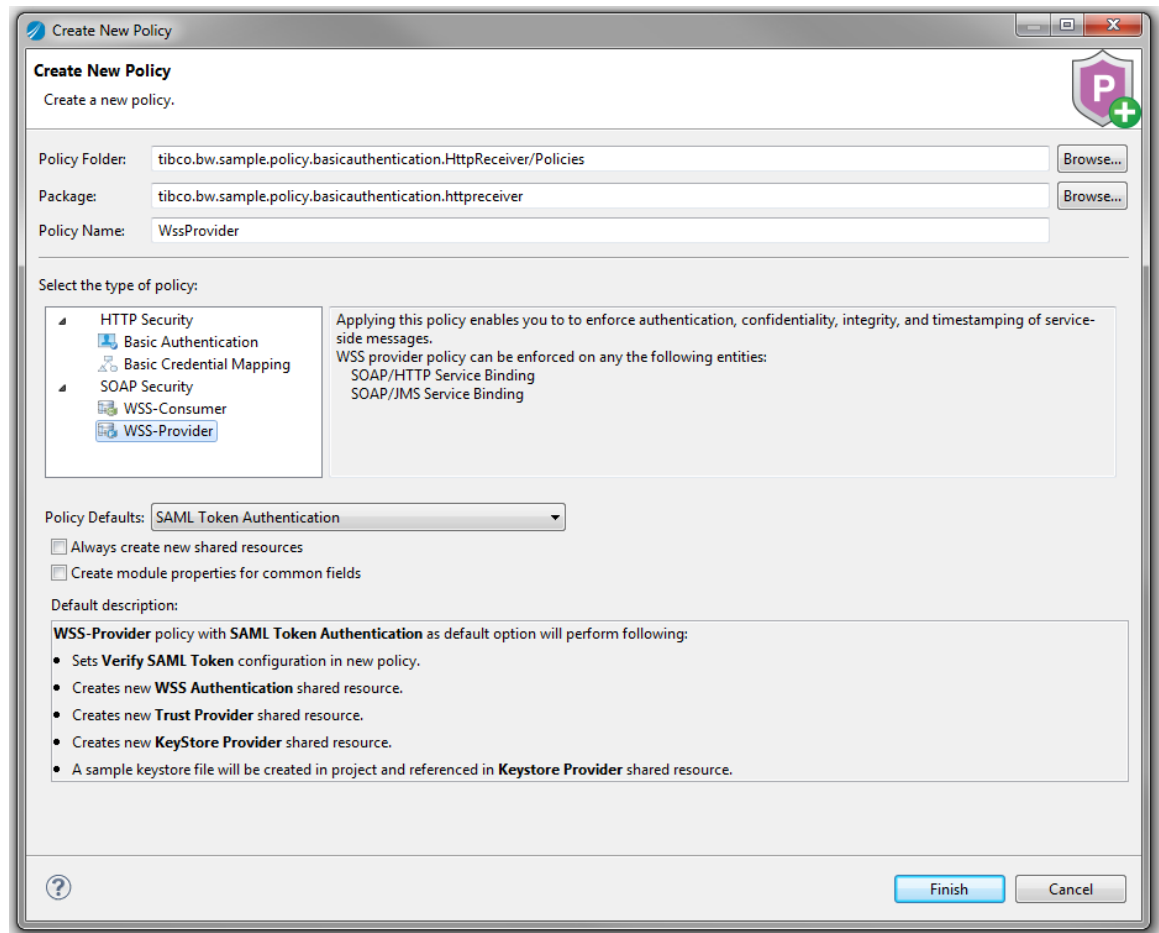
### Setting Up a Policy with Resources

Follow these steps to set up a new WSS Provider policy with resources:

1. In the **Project Explorer**, right-click the **Policies** folder and select **New > Policy**.



The Policy Wizard opens.



2. Specify the following values in the Create New Policy Window:

- **Policy Folder:** Name of the folder where policies will be located.
- **Package:** Name of the package in the module where the new policy is added. Accept the default package, or browse to select a different package name.
- **Policy Name:** Name of the new policy. By default, the policy name is configured to match the security policy you choose.

3. Under **Select the type of Policy**, select **WSS Provider**.

4. From the **Policy Defaults** drop-down menu, select one of the following options:



The **Policy Defaults** menu offers a list of commonly used policy configurations to choose from. After you select a Policy Default, a policy with preconfigured settings and related resources is created. If resources already exist in the module, the newly created policy automatically refers them. However, if no resources exist, new resources with default settings are created and referred to by the policy. Refer to the **Default description** at the bottom of the **Policy Wizard** to view policy configurations and new resources that might be created.

- **SAML Token Authentication:** Select this option to authenticate credentials through SAML assertion. A WSS Provider policy configured for SAML token-based authentication and the following resources are produced in your workspace:
  - A sample keystore file with the default file name `truststore.jks`.
  - A **Trust Provider** resource with the default file name `WssProvider_TrustStore.trustResource`

- A **KeyStore Provider** resource with the default file name  
`WssProvider_KeystoreProvider.keystoreProviderResource`
  - A **WSS Authentication** resource with the default file name  
`WssProvider_WSSAuthProvider.wssResource`
  - **UserName Token Authentication with LDAP:** Select this option to authenticate credentials through user name token authentication with LDAP. A WSS Provider policy configured for user name token-based authentication with LDAP and the following resources are produced in your workspace:
    - An **LDAP Authentication** resource with the default file name  
`WssProvider_AuthenticationProvider.ldapResource`
    - A **WSS Authentication** resource with the default file name  
`WssProvider_WSSAuthProvider.wssResource`
  - **UserName Token Authentication with Workspace XML:** Select this option to authenticate credentials through user name token-based authentication with an XML file authentication resource stored in your workspace. A WSS Provider policy configured for XML file authentication and the following resources are produced in your workspace:
    - An **XML Authentication** resource with the default file name  
`WssProvider_AuthenticationProvider.authxml`
    - A **WSS Authentication** resource with the default file name  
`WssProvider_WSSAuthProvider.wssResource`
    - A preconfigured XML file with the default file name `XmlUsers.xml` is created if an XML file does not already exist.
  - **UserName Token Authentication with Filesystem XML:** Select this option to authenticate credentials through user name token-based authentication with an XML file authentication resource stored in your local file system. A WSS Provider policy configured for XML file authentication and the following resources are produced in your workspace:
    - An **WSS Authentication** resource with the default file name  
`WssProvider_WSSAuthProvider.wssResource`
    - An **XML Authentication** resource with the default file name  
`WssProvider_AuthenticationProvider.authxml`
  - **Empty Policy (No Default) :** Select this option to create a new WSS Provider policy with no preselected options and no resources.
5. **Optional.** Select **Always create new shared resources** to ensure new resources are generated for the policy and referred to by the policy.
  6. **Optional.** Select **Create module properties for common fields** to override default properties in newly created resources with module properties. Resources with module properties for common fields are generated after you select this option.
  7. Select **Finish** to create the policy.

### Configuring Resources and the Policy

For resource configurations, refer to the following topics under the "Shared Resources" topic in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide:

- Identity Provider
- Keystore Provider
- Subject Provider

- Trust Provider
- WSS Authentication

For policy configuration details, refer to the topic "WSS Provider" under "Policy Resources" in the *TIBCO ActiveMatrix BusinessWorks Bindings and Palettes Reference* guide .

### **Associate the Policy with a Binding**

You can associate the WSS Provider policy with the following bindings:

- SOAP-HTTP Service Binding
- SOAP-JMS Service Binding

For instructions on how to enforce a policy on a binding in your application, refer to [Associating Policies](#).

## Building Projects Automatically

---

The **Auto Build** option in TIBCO Business Studio™ for BusinessWorks™ builds projects automatically.

This option can be turned on or off from TIBCO Business Studio for BusinessWorks and can also be configured from the `config.ini` file by setting the property `bw.autobuild` to `true` or `false`.

In TIBCO Business Studio for BusinessWorks, the **Auto Build** option which is enabled by default, can be turned on or off from **Project > Build Automatically**.

From the `config.ini` file set the property `bw.autobuild` to `true`, to build projects automatically.

When the value of the `bw.autobuild` property is `true`, the **Build Automatically** feature is turned on when TIBCO Business Studio for BusinessWorks is started. When the value of the property is `false`, auto building is turned off.



If the value of this property is changed, restart TIBCO Business Studio for BusinessWorks for the changes to be applied.



# XPath

---

XML Path Language (XPath) is used to navigate through elements and attributes in an XML document. XPath uses path expressions to navigate through XML documents. XPath also has basic manipulation functions for strings, numbers, and booleans.

ActiveMatrix BusinessWorks uses XPath as the language for defining conditions and transformations.

For a complete description of XPath, refer to the XPath specification (from <http://www.w3.org/>). This section covers the basics of XPath and its use in the product.

## XPath Basics

This product uses XPath (XML Path Language) to specify and process elements of data schema. These data schema are either process variables or input schema for an activity. You can also use XPath to perform basic manipulation and comparison of strings, numbers, and boolean.

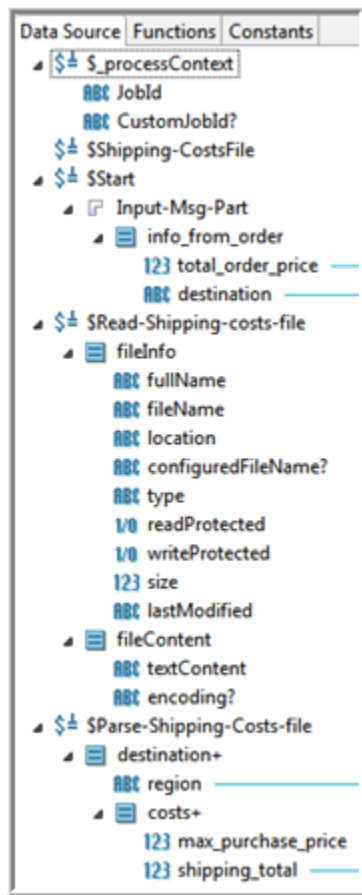
To use XPath in the product, you need to be familiar with the basic XPath concepts. However, to learn more about XPath when building complex expressions refer to the XPath specification from <http://www.w3.org/>

### Addressing Schema Elements

All data source and activity input are represented as an XML schema. The data is represented as a schema tree regardless of where the data is derived from or its format. The data can either be simple (strings, numbers, boolean, and so on), or it can be a complex element. Complex elements are structures that contain other schema elements, either simple elements or other complex elements. Both simple and complex elements can also repeat. That is, they can be lists that store more than one element of the type specified.

XPath is used to specify which schema element you refer to. For example, the following schema might be available for an activity's input.

### Schema Elements in Data Source



The data source area of the example **Input** tab shows the output schema of the activities in the process. There are two output schema, each a root node in the data source area: **Read-Shipping-Costs-file** and **Parse-Shipping-Costs-file**. Each of these schema has its own associated structure, for example, **Read-Shipping-Costs-file** has a set of simple values and **Parse-Shipping-Costs-file** has simple data and other complex data.

To reference a particular data item in any of these schema, start with the root node and then use slashes (/) to indicate a path to the desired data element. For example, if you want to specify the **region** attribute in the **destination** complex element that is in the **Parse-Shipping-Costs-file** node, use the following syntax:

```
$Parse-Shipping-Costs-file/destination[<< Filter >>]/region
```

The path starts with a dollar (\$) sign to indicate it begins with a root node and continues with node names using slashes, like a file or directory structure, until reaching the desired location name.

### Namespaces

Some schema elements need to be prefixed with their namespace. The namespace is automatically added to elements that require this element when creating mappings on the **Input** tab of an activity or when dragging and dropping data in the XPath builder.

### Search Predicates

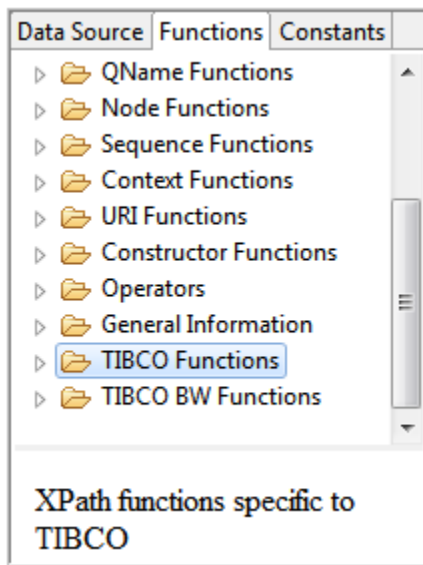
An XPath expression can have a search predicate. The search predicate is used to locate a specific element in a repeating schema element. For example, the **\$Parse-Shipping-Costs-file/destination/region** item is a repeating element. To select only the first item in the repeating element, specify the following:

```
$Parse-Shipping-Costs-file/destination[1]
```

The [1] specifies the first element of a repeating item. Sub-items can also be examined and used in a search predicate. For example, to select an element whose destinationID is equal to "3A54", specify the following:

**\$Parse-Shipping-Costs-file/destination["3A54"]**

See the online documentation available in the XPath Builder for a list of the available operators and functions in PATH.

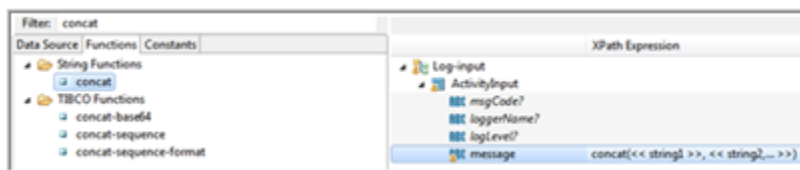


You can also use the Custom XPath Function Wizard to create your custom XPath function group. For more information, see **Creating Custom XPath Functions** in the TIBCO ActiveMatrix BusinessWorks™ *Bindings and Palettes Reference* guide.

## XPath Expression

The XPath expression is used to creating transformations on the **Input** tab of any activity.

When the function is placed into the **XPath Expression**, placeholders are displayed for the function's parameters.

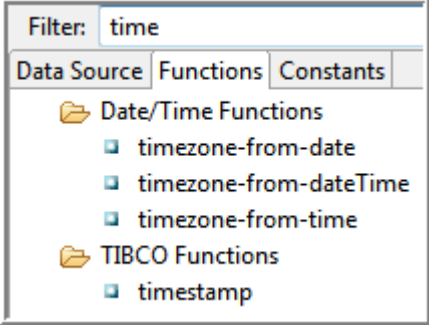


You can drag and drop schema elements from the **Data Source** tab into the function's placeholders.

### XPath Builder Formula Elements

The following table shows the different elements of XPath Builder.

| Elements    | Description                                                                                                                            |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------|
| Data Source | Displays the data source schema tree. All elements in this tree are available to drag and drop into the <b>XPath Expression</b> field. |

| Elements  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Functions | <p>Displays the available XPath functions. These are categorized into groups and each function can be dragged from the function list into the <b>XPath Expression</b> field.</p> <p>When the function is placed into the <b>XPath Expression</b>, placeholders are displayed for the function's parameters. You can drag and drop schema elements from the <b>Data Source</b> tab into the function's placeholders.</p> <p>For more information about XPath functions, select XPath functions in XPath builder. The description of the function is displayed.</p> |
| Filter    | <p>Use this field for a refined function search in the mapper.</p> <p>Clicking the <b>Functions</b> tab displays the <b>Filter</b> field.</p> <p>For example, type "time" in the <b>Filter</b> field to obtain consolidated results relating to "time" function.</p>                                                                                                                                                                                                           |


| Elements  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Constants | <p>Displays the constants available for use in XPath expressions. These are categorized into groups and each constant can be dragged from the constants list into the <b>XPath Expression</b> field.</p> <p>Constants are useful for inserting special characters, such as quotes, symbols, and so on, into XPath formulas. Constants are also defined for commonly used items, such as date/time formats.</p> <p>Constants can also be used for inserting the following <b>TIBCO BW Predefined Module Properties</b>.</p> <ul style="list-style-type: none"> <li>• <b>Activity Name</b> - returns the name of the activity on which the module property is set.</li> <li>• <b>Application Name</b> - returns the application name.</li> <li>• <b>Application Version</b> - returns the version of the application specified in the <b>Version</b> field under the <b>Overview</b> tab of the application.</li> <li>• <b>Application Full Version</b>- returns the three digit version of the application in the form of &lt;major&gt;.&lt;minor&gt;.&lt;micro&gt;.</li> <li>• <b>AppNode Name</b> - returns the name of the AppNode on which the application is deployed.</li> <li>• <b>AppSpace Name</b> - returns the name of the AppSpace on which the application is deployed.</li> <li>• <b>Deployment Unit Name</b> - returns the ID of the application specified in the <b>ID</b> field under the <b>Overview</b> tab of the application.</li> <li>• <b>Deployment Unit Type</b> - returns the deployment unit type as application.</li> <li>• <b>Deployment Unit Version</b> - returns the deployment unit version specified in the <b>Version</b> field under the <b>Overview</b> tab of the application.</li> <li>• <b>Domain Name</b> - returns the name of the domain in which the application is deployed.</li> <li>• <b>Module Name</b> - returns the name of the application module.</li> <li>• <b>Module Version</b> - returns the version of the module specified in the <b>Version</b> field under the <b>Overview</b> tab of the application module.</li> <li>• <b>Process Name</b> - returns the name of the process in which the module property is used.</li> <li>• <b>Process Stack</b> - returns the entire process path including the nested subprocesses, and the parent process. For example<br/> <code>main.Process/SubProcess1-&gt;sm.SubProcess1/<br/> SubProcess2-&gt;sm1.SubProcess2</code></li> <li>• <b>Engine Name</b> - returns the name of the engine. By default, the name of the engine is Main. You can change the engine name by setting the property <code>bw.engine.name=Main</code> in the appspace <code>config.ini</code> file.</li> </ul> |

| Elements            | Description                                                                                                                                                                                       |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Documentation Panel | Describes each selected function.<br><br>On clicking a function on the <b>Function</b> tab, the documentation panel gives a brief description of the selected function with one or more examples. |
| XPath Expression    | Displays the XPath formula you want to create.<br><br>You can drag and drop items from the <b>Data Source</b> tab or the <b>Functions</b> tab to create the formula.                              |

## XPath Builder

Using XPath Builder, you can drag and drop schema elements and XPath functions to create XPath expression.



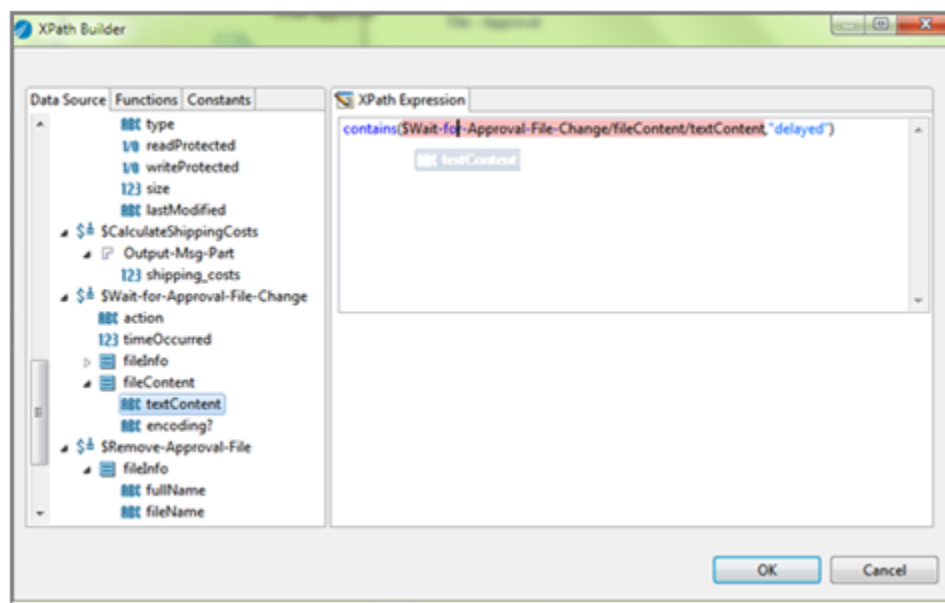
Click the Transition in the process. On the **General** tab, select **Success with condition** option in the **Condition Type** field. This displays the **Expressions** field. Click  icon to open the XPath Builder window.



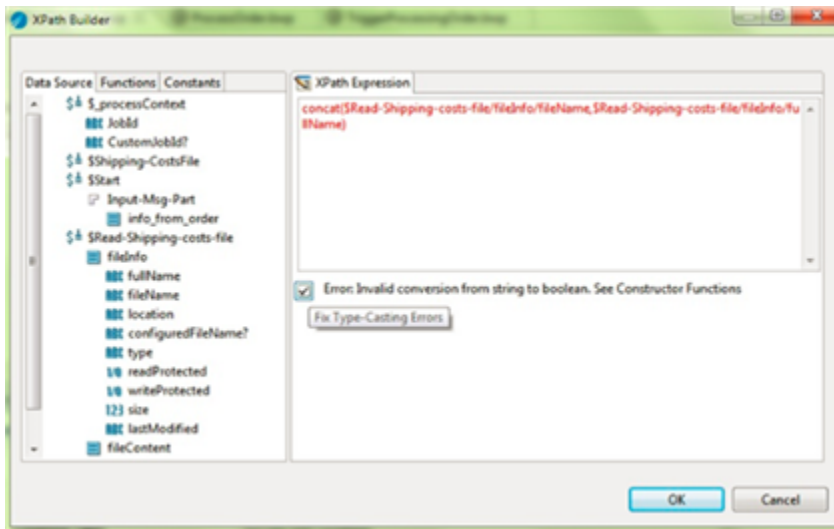
XPath Builder is also available from **Sequence Key** field and **Custom Job Id** field of all process starter activities (such as **Timer**, **File Poller**, and so on).

The following image shows how you can use XPath Builder to drag and drop schema elements into function placeholders.

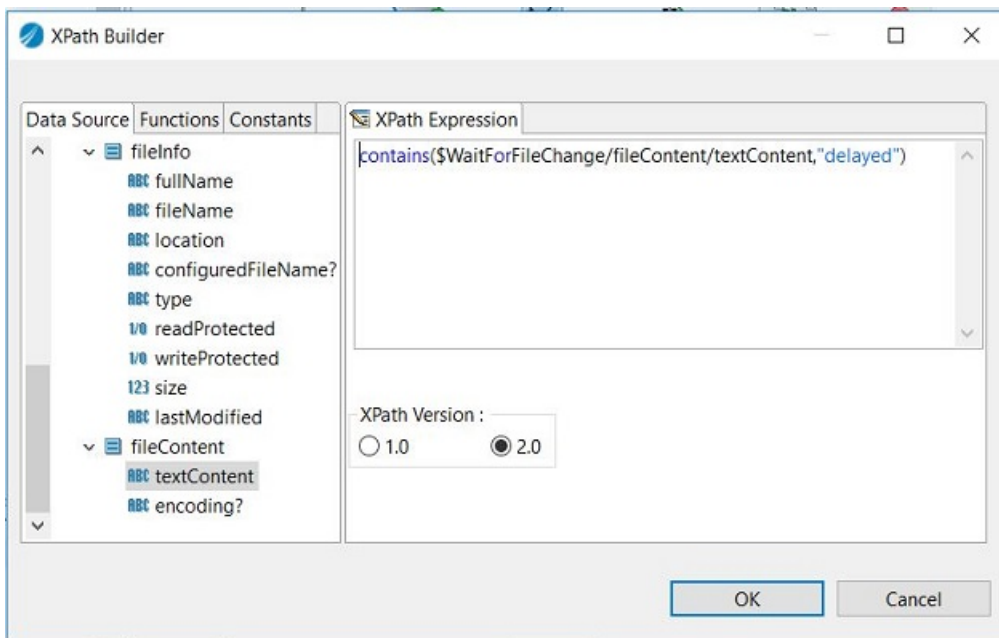
### XPath Builder



See the following image for the displayed result of evaluating the formula shown in the **XPath Expression** field. The errors in the formula are displayed here.



For Group activities and transitions, you can see the new field **XPath Version** added in the dialog box as follows:



### TIBCO BW Functions

XPath Builder can be used to fetch process related information for any activity. These functions are listed under the **TIBCO BW Functions** group.

- **getModuleProperty**: Returns the value of a module property. Also see **TIBCO BW Predefined Module Properties** under the **Constants** section.
- **getSystemProperty**: Returns the value of a Java system property.
- **restartedFromCheckpoint**: Returns `true` if the process instance recovered from a checkpoint, otherwise returns `false`.
- **generateEPR**: Returns an 'Endpoint Reference' as a string. This value can be used as an input to the **Set EPR** activity.
- **getHostName**: Returns host name of the host machine.



The XPath function `xsd:string()` saves double values in scientific notation if the double value has 7 or more digits before the decimal point. For example, if the value is 1000000.333, the `xsd:string()` function renders the value as 1.000000333E6.



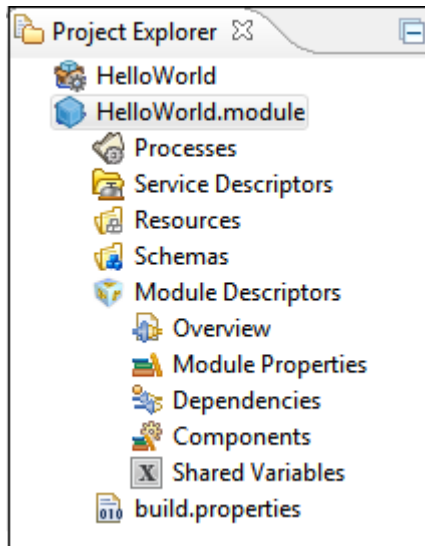
# Developing a SOAP Service

A SOAP service makes a Process service available as a SOAP web service. You can achieve this by applying a SOAP service binding on the target process service.

## Implementing a SOAP Service Provider

### Procedure

1. Click on the process package, for example, "HelloWorld", and then click on the **Create a new Business Works Process**  icon.

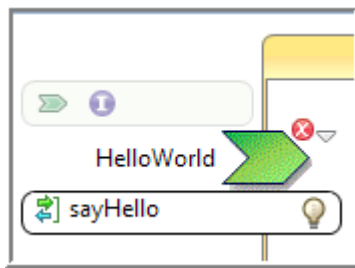


2. Select a process on which you want to add a service, and click the **Create Service** icon.

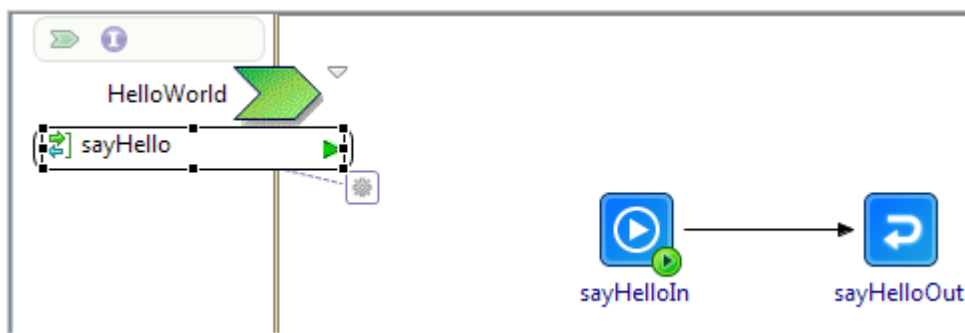
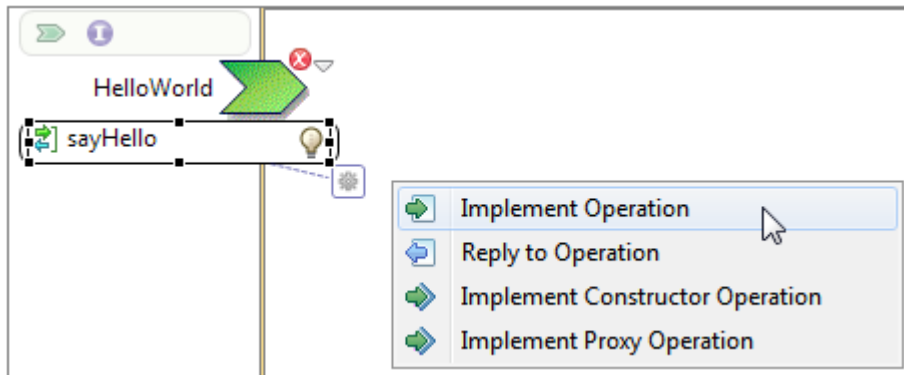


The New Service dialog opens.

3. In the New Interface section specify the **Interface Name** as HelloWorld and **Operation Name** as sayHello. Click **Finish**.



4. To implement the operation, drag and drop the sayHello operation, and select **Implement Operation**.

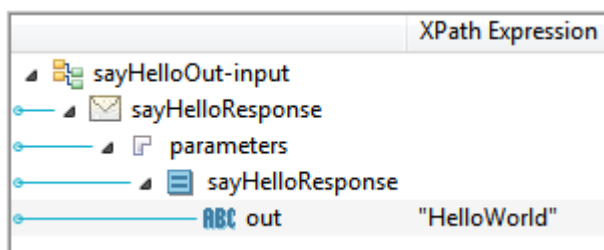


Choose **Implement Constructor Operation** option, if there are multiple operations in a Port type.

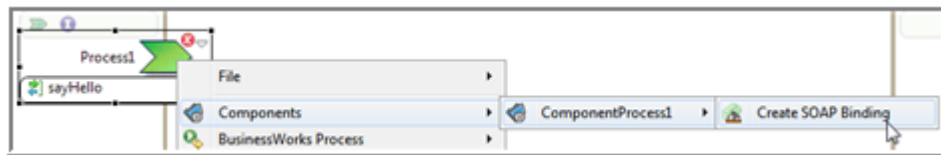


The option **Implement Operation** implements a single operation and creates a single Receive activity and a Reply. The option **Implement Constructor Operation** implements a constructor. A constructor provides for multiple operations. Use this option if the PortType has multiple operations which must be implemented by this process.

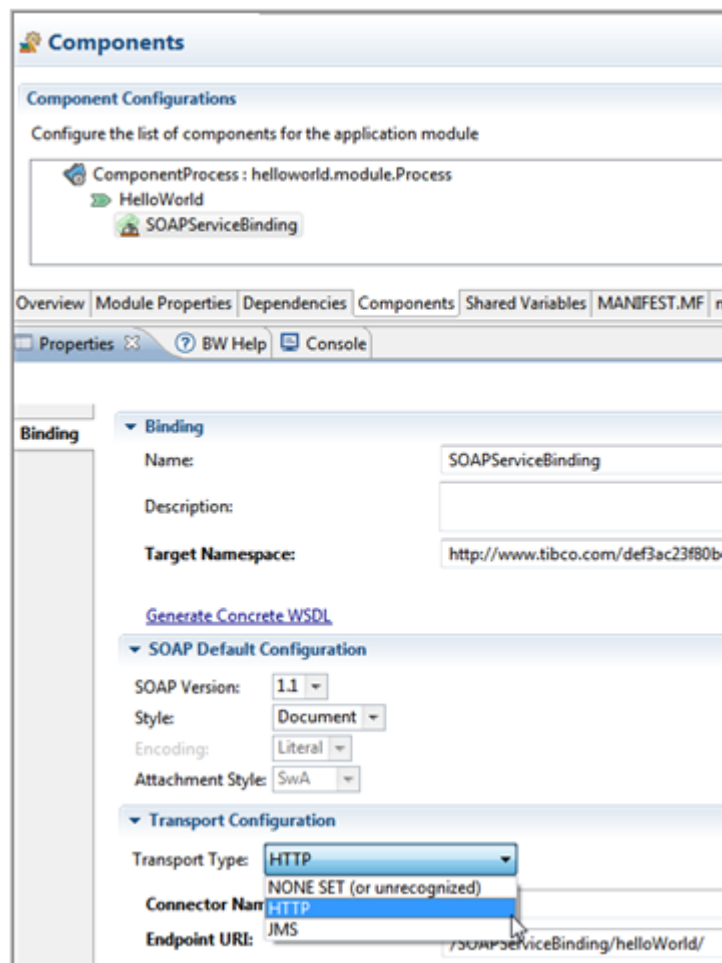
5. Click on the **Reply** activity (**sayHelloOut**) and under the Properties view, click the **Input** tab. Configure Reply message.



- Right-click on green chevron and select **Components > ComponentsProcess > Create SOAP Binding**. The Binding Configuration dialog displays.



- To configure transport on the SOAPServiceBinding, select **HTTP** from the **Transport Type** drop-down list in **Transport Configuration**.

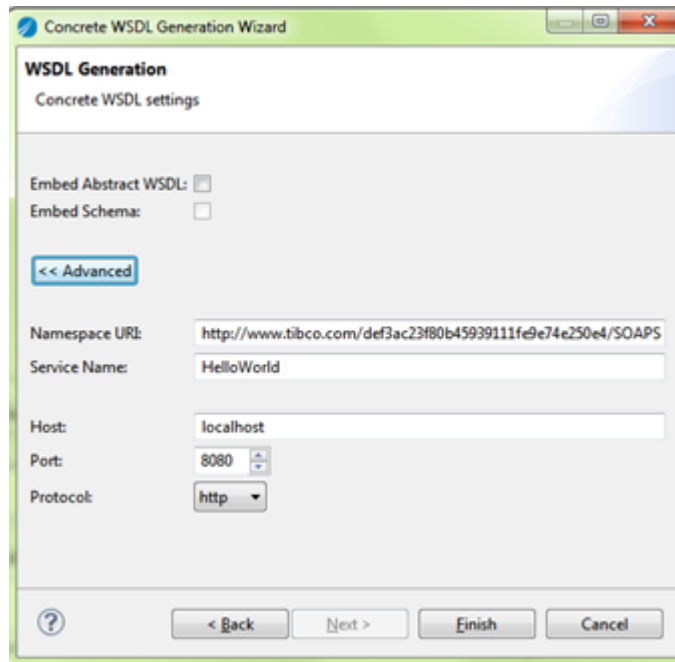


- Click on **Create Shared Resource** button and click **Finish** on the Create HttpConnResource Resource Template.  
The default port used by this shared resource is 8080. The service binding is now created.
- To generate the concrete WSDL of the SOAP service created in the above steps, click **Generate Concrete WSDL** link.
- Click **Workspace**. In the Folder Selection window and select the **Service Descriptor** folder of the current module and click **OK**.  
The Generate Concrete WSDL screen will now show the specified location and the name of the WSDL.



To create the Concrete WSDL in a desired location other than the workspace location, specify it by using **File System** button and click **Finish**.

11. To avoid namespace resolution error, click **Next** and clear the **Embed Abstract WSDL** and **Embed Schema** check boxes and click **Finish**.



Click on the **Advanced** tab to override the **Namespace URI**, **Service Name**, **Host**, **Port**, and **Protocol** fields.


The concrete WSDL is generated at the specified location.

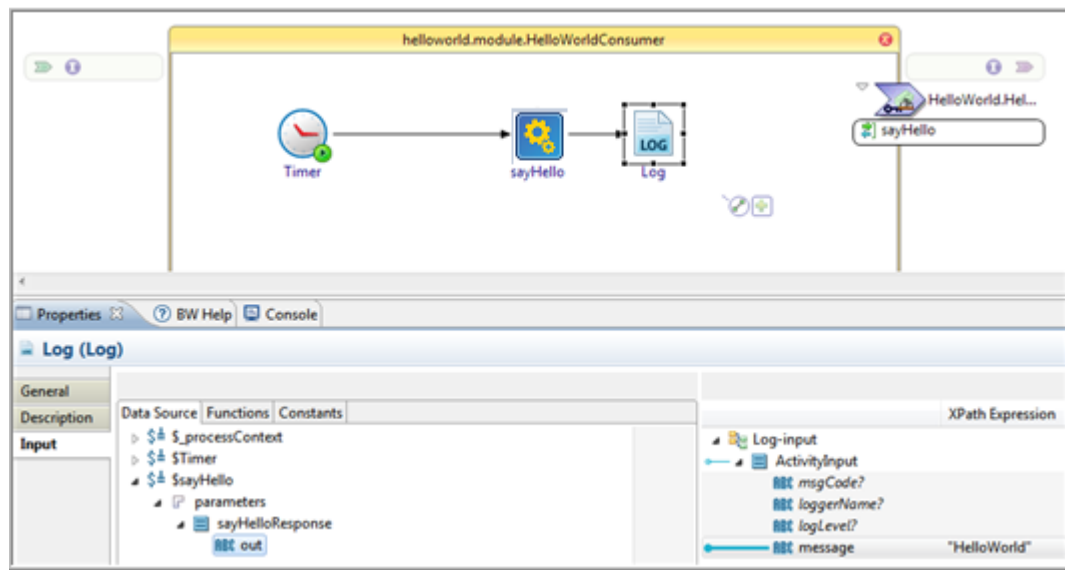
## Consuming SOAP Services

The request message is generated by the SOAP reference binding for a service and response message is received by the reference binding from the service.

### Creating a Consumer for SOAP Service

#### Procedure

1. Click on the process package, for example, "HelloWorld", and then click **Create a new Business Works Process**  icon.
2. Specify the process name as HelloWorldConsumer and click **Finish**.
3. Drag and drop the **HelloWorldSOAP** portType to the right of the process editor.
4. Add a Reference Binding to the SOAP service for the **Reference Type** field by selecting the required reference from the drop down list.
5. Select and drop a **Timer** and a **Log** activity on the process and join it with the **Invoke** activity as shown in the image. Also, configure the **Log** activity with a message.



The SOAP reference binding is created.

6. Run the project.

## Developing a RESTful Service

---

Services are used to invoke a process and to call out of the process so that a process receives data from a service and routes data to a service.

The key abstraction of information in REST is a resource. REST ignores the details of component implementation and protocol details. TIBCO ActiveMatrix BusinessWorks™ currently allows the following HTTP methods to be performed on resources: GET, PUT, DELETE, and POST. Both XML and JSON are supported as data serialization formats along with support for definition of custom status codes, path(URL) parameters, key-value parameters, query parameters, and custom HTTP headers.

### General Restrictions

- No wildcards or attribute wildcards. For example, any element and any attribute is not supported.
- Complex types might not contain both an attribute and a child element with the same local name.
- Complex types might not be of the pattern "simple type plus attributes".
- Complex types might not contain mixed content.
- Attributes that are not part of the default (empty) namespace, cannot be used for Complex Elements.
- The 'choice' and 'sequence' compositors might not have `maxOccurs > 1` (same as the restriction on 'all' in the schema specification).
- Substitution groups are not supported.
- Element of simple type with an attribute is not supported.
- The `elementFormDefault` can only be qualified for schemas used by REST binding and JSON activities.
- Schemas should not contain cyclic dependencies within same schema, or on the other schemas.
- Schemas should not have a type that has two child members with the same local name, but different namespaces.
- For float and double values, XML schema always shows exponential values of type `1.0E0`

## Implementing a REST Service Provider

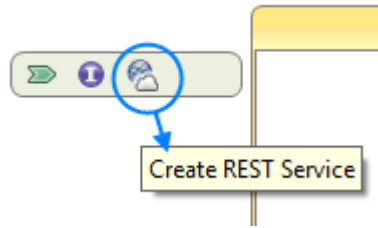
A REST service provider exposes the resources in a process definition that can be invoked by clients using one of the following operations- POST, GET, PUT, PATCH, and DELETE.

### Prerequisites

If a schema definition does not exist, create (or import) a schema definition in the process to which you want to add the REST service.

### Procedure

1. In the Project Explorer, select the process to which you want to add the REST service. There are multiple ways to invoke the wizard to create a REST service.
  - From the main menu, select **File > New > BusinessWorks Resources > BusinessWorks REST Resource**.
  - Right-click the menu, select **New > BusinessWorks REST Resource**.
  - Click **Create REST Service** in the process editor area. (Note that REST services can only be created in stateless BusinessWorks processes.)



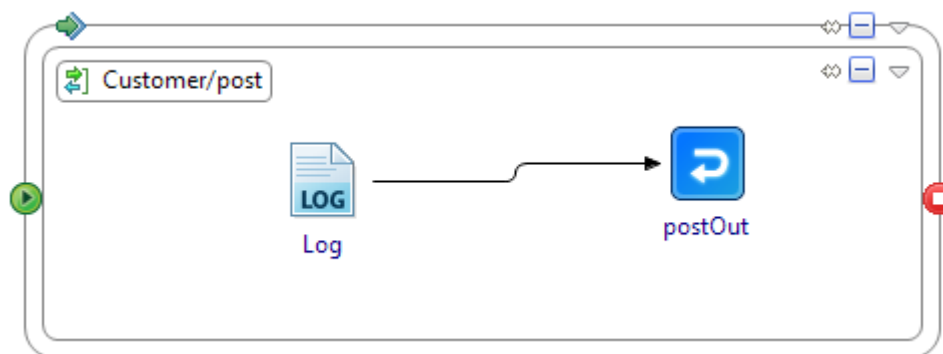
For more information, see "REST Binding" in the *TIBCO ActiveMatrix BusinessWorks™ REST Reference* guide.

2. In the Create a New REST Service wizard, configure the REST service implementation by specifying the values for Resource Service Path, Type of Resource, Operations, and Implementation Data.
  - **Summary** about the new REST service.
  - **Resource Service Path:** Specifies the URI that is used to access the REST service.
  - **Type of Resource:** Select if the service works on a single resource or a collection.
  - **Operations:** By default, the GET operation is selected. Select or deselect the operations as needed.
  - **Resource Schema:** Select a resource schema for the REST service, if needed.
  - **Implementation Data:** Choose between structured and opaque implementation data.
3. Optionally, click **Next** to configure the selected operations individually to specify the nickname for the operation (default nickname is of the format `<operation><resource_name>`), summary, and the request and response elements and their data types.
4. Click **Finish**.  
The wizard adds the REST service and the selected operations, and also creates a process definition with the multiple operations.



The REST service always implements the constructor operator.

5. Add activities to the process and configure them appropriately. For example, update the POST process to add a **Log** activity to log the requests and connect the postOut activity to **Log** activity.



6. Configure the input and output properties for the activities. For example, select postOut activity and then select **Properties > Input**. Expand the data tree on the **Data Source** tab and map the post element from the left to the post Response element on the right to echo the element. Similarly, for **Log** activity, map the post element on the left to the ActivityInput message element on the right.
7. Save your changes.

## Result

The REST service is built and can be tested using the built-in tester Swagger UI. For more information on Swagger UI, see "Testing the REST Service" in the *TIBCO ActiveMatrix BusinessWorks™ Getting Started* guide.

## Discovering API Models from TIBCO Business Studio™ for BusinessWorks™


You can use the **API Explorer** view in the TIBCO Business Studio for BusinessWorks to view the APIs that reside on your local machine or on a remote server.

### Prerequisites

For the API Explorer to discover the APIs residing on a remote server, the remote server must be up and running.

You can set up the locations to which you want the API Explorer to connect and look for the APIs. To do so, follow the steps below.

### Procedure

1. In TIBCO Business Studio for BusinessWorks, go to the **API Explorer** view.
2. In the button bar within the **API Explorer** tab, click the **View Menu** downward-facing triangle icon () and select **Settings**.  
The Settings dialog will open.

The registries for the ActiveMatrix BusinessWorks - API Modeler and the samples folder installed on your local machine are configured and appear in the API registry configurations box by default. In this dialog, you can specify how the discovered APIs will appear in the API Explorer:

- **API Presentation** - specifies how the APIs will appear in the **API Explorer**
  - Flat** - displays the APIs as a flat list with each API's version number displayed next to its name in parenthesis. If there are multiple versions of the same API, each version will be shown as a separate API, hence multiple APIs with the same name but different version numbers.
  - Hierarchical** - displays every API as a hierarchy of API name label with version number folder under it and the actual API under the version folder. If there are multiple versions for an API, each version will be listed in its own separate folder under the API name label.
  - Latest Version** - displays only the latest version of the API, even though there might be multiple versions available.
- **Group by API registry** - groups the APIs according to the registry from which they were discovered
- **API registry configurations** - displays the list of API registries that are currently configured in your TIBCO Business Studio for BusinessWorks installation.

Select the API registry check boxes to display the APIs.

You can edit an existing registry by clicking the **Edit** button, delete the registry configuration by clicking **Remove**, or changing the order in which the registries show up in the API Explorer by using the **Up** and **Down** button. These button get activated when you click on an API registry name.

3. Click **New** to add a new registry.
4. In the **Create new API Registry client configuration** dialog do the following:
  - a) Enter a name for the API registry that you will be mapping to in the **Name** text box.









- b) Select the **Local** radio button to map a location where the APIs are stored on your local machine's hard drive and navigate to the location using the **Browse** button. Alternatively, select the **Remote** radio button if you want to map to a remote server that contains the APIs and enter the URL for the server in the **URL** text box.
5. Click **Finish**.  
You should now see the APIs displayed in the **API Explorer** in the format that you specified in the Settings dialog. Expanding an API will show you its version, the resource path, and the operations you can perform on that resource.



Organizations can have multiple owners, and a list of owners is displayed in the Edit API Registry client configuration page.

The **API Explorer** view has the following quick-access buttons that you can use to format the way the APIs are listed:

-  **Refresh**
-  **Expand All**
-  **Collapse All**
-  **Group by API Registry**
-  **API Presentation**
-  **API Registries**. Selecting a registry from this drop-down list toggles between displaying and hiding the registry in the **API Explorer**.

Use the search filter that appears at the bottom of the **API Explorer** view to search for API names that match the string that you enter in the **Filter** text box. You can search by typing in the version number, the full API name, or a full word within an API name. Wildcards are not supported. The search is case insensitive.

## Importing an API Model into your Workspace

The APIs that are discovered from local and remote servers are displayed in the **API Explorer** tab of the TIBCO ActiveMatrix BusinessWorks™. You can use these APIs in your project by importing them into the **Service Descriptors** folder of the project. The `.json` file for the API gets copied into the application module.

To import the APIs from the **API Explorer** into your project follow these steps.

### Procedure

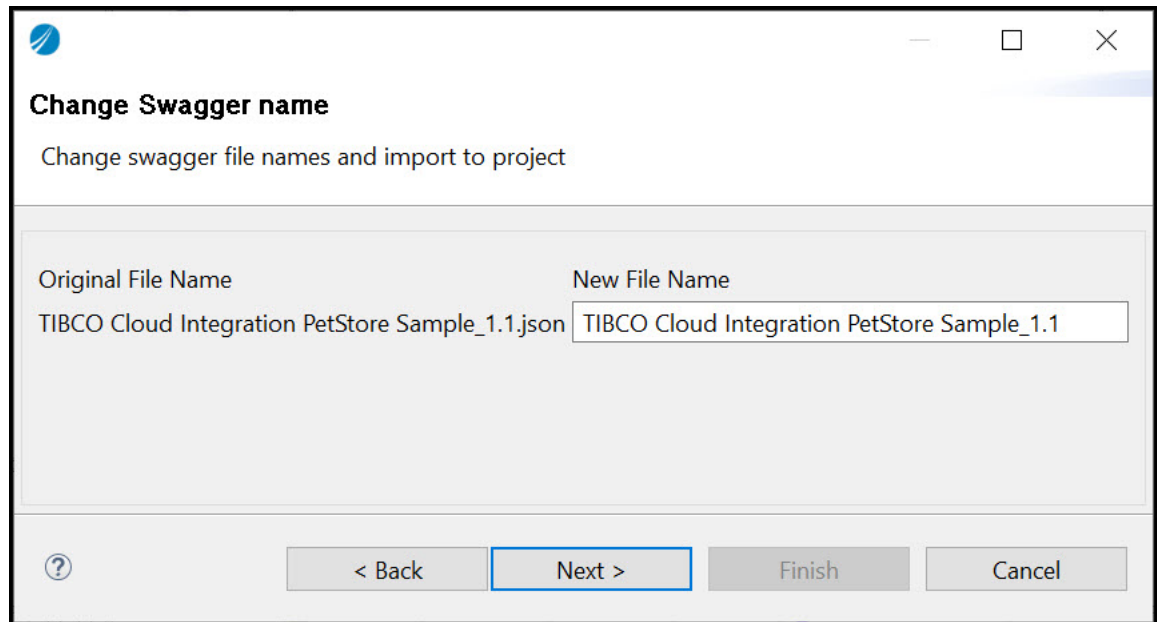
1. Right-click on one or more API names in the **API Explorer** and select **Import**.  
The Import API dialog opens.

Every API you selected in the **API Explorer** is listed in this dialog. If an API has multiple versions, all versions are listed. By default, all APIs listed here are selected. You can deselect APIs that you do not want to import by clearing its check box.

2. Select the appropriate action and click **Next**.

| Option                                                        | Description                                                                                                                  |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <b>Import to project</b>                                      | Select the radio button to import the API into an existing project and browse to the project using the <b>Browse</b> button. |
| <b>Create a new project and import API to the new project</b> | To create a new project and import the API into that project select the radio button.                                        |
| <b>API list to import</b>                                     | Select the API or the appropriate version of the API when there are multiple versions of the API available.                  |

The Change Swagger name dialog box opens.



Change the swagger file name if required. Click **Next**.

The New BusinessWorks Application Module dialog box opens.

3. Create a new application module with appropriate details and click **Finish**. You should see the API(s) under the **Service Descriptors** folder of the project. You can create sub-folders under the **Service Descriptors** folder and drag-and-drop APIs into them if you prefer to organize the APIs into a meaningful folder structure.

As an alternative to the above procedure, you can also drag and drop the API from the **API Explorer** into the project's **Service Descriptors** folder.



APIs that were created using a Swagger file must be implemented exactly as defined by the Swagger file. TIBCO Business Studio for BusinessWorks allows you to only view the parameters and operations that are defined in the Swagger file. You cannot create any new parameters or operations for such applications.

## Creating an XML Schema for a Swagger 2.0 File Imported in TIBCO Business Studio™ for BusinessWorks™

TIBCO Business Studio for BusinessWorks supports the creation of an XML schema for an imported Swagger 2.0 file.

You can create an XML schema for the Swagger 2.0 files in one of two ways described below.

### Prerequisites

The Swagger 2.0 file must exist in the **Service Descriptors** folder of the project. Be sure to import the Swagger file into the **Service Descriptors** folder before you follow these steps:

### Procedure

1. Drag and drop the Swagger file on the right side of the canvas to create a REST service binding. This action generates an XML schema for the Swagger file under the **Schemas** folder. The XML schema file has the same name as the Swagger file.  
Or
2. Right-click the Swagger file in the **Service Descriptors** folder and select **Refactor > Generate XSD Schema**.



- To see which XML schema is related to the Swagger file, right-click the Swagger file and select **Refactor > Open XSD Schema**.
- If you have multiple Swagger files all of which contain a definition for the same object, the definition for the object in all the Swagger files must be identical.
- If you have multiple Swagger files with one file (a master file) containing a super set of definitions contained in the other files, generate an XSD file from the master Swagger file that contains the super set, and create links to the other files in the master Swagger file. If you create a link to the super set file in one of the subset files and then create an XSD from the subset file, then the XSD will contain only those elements that are common to both files. It will not contain elements for definitions that exist only in the super set file.

## Synchronizing the Imported REST API Models in TIBCO Business Studio™ for BusinessWorks™

If a REST service developer has made changes to the service API after creating the service, the changes need to be propagated to all the places where the service is used. You can check for updates to a Swagger file that has been imported into TIBCO Business Studio for BusinessWorks. The icon to the left of the Swagger file in the **Project Explorer** in the TIBCO Business Studio for BusinessWorks displays an indication that the file has been modified in its original location and the local copy of the file is not in synchronization with its source.

You can check for differences between the original Swagger file and its copy that was created when importing it into the TIBCO Business Studio for BusinessWorks. You can also compare the differences between the two and update your local copy if need be. To do so, follow these steps:

### Procedure

1. Right-click the Swagger file under **Service Descriptors** in the **Project Explorer**.
2. Select **Remote Interface**.

The **Check for Differences** menu option checks for differences between the imported copy and its original.

The **Compare Differences** menu option first checks for differences between the imported copy of the Swagger file and its original. If there is a difference, the file will appear in the **Synchronize** tab and if you double click it there it displays the two files side by side with the differences highlighted.

The **Update Local Copy** menu item updates the copy of the file in your workspace to match its original. It also regenerates the schema.



No changes are performed for processes that have already been created.

# Developing Java Applications

The enhanced Java development tooling in TIBCO Business Studio™ for BusinessWorks™ can be used to develop and debug the Java code. Using the software, you can develop applications graphically (without coding), use existing Java classes, or write custom Java code.

## Adding Java-Specific Behavior to Projects

Eclipse projects use the **project nature** definition to tag a project as a specific kind of project. By configuring a project to use the **Java nature**, you can leverage on the enhanced Java development tooling available in TIBCO Business Studio for BusinessWorks to develop Java application. A project with Java nature contains a default source folder for Java classes, `src`, in addition to other folders.



You can choose a different source folder by configuring the specified folder as the source folder and including the folder in the build path.

You can specify the project nature for an application module in one of the following ways:

- When creating a new application module, select the **Use Java configuration** check box.
- For an existing application module, right-click the project name in the Project Explorer view and select **Configure > Convert to Java project**.

## Accessing Java Classes or Libraries from an ActiveMatrix BusinessWorks™ Application

An ActiveMatrix BusinessWorks™ application can invoke Java classes or reference libraries containing the Java code, using activities from the **Java** palette. Depending on the use case, the Java classes or libraries can reside in one of the following locations:

- Within the same application module as the ActiveMatrix BusinessWorks™ process: when the Java code need not be accessible from other applications, include the Java class within the same application module. See [Using a Simple Java Invoke Activity](#) for details.
- In a shared module or Eclipse plug-in project : when the Java code must be shared by multiple applications, use a shared module with Java nature or an Eclipse plug-in project to contain the Java code.
- External to the ActiveMatrix BusinessWorks™ application: when you do not have access to the Java source files and only the Java classes are available, you can invoke the Java methods stored in JAR files.

## Using a Simple Java Invoke Activity

The **Java Invoke** activity can invoke a Java method from a class that resides in the same application module, a shared module or an eclipse Plug-in project.

### Prerequisites

The project must be configured with Java nature. For more information, see "Adding Java Nature to a Project" in the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palette Reference* guide.

### Procedure

1. In the **Project Explorer** view, expand the application module project and right-click the Java source folder, `src` (default), and select **New > Class**.
2. In the New Java Class wizard, specify the package name and name of the Java class, and click **Finish** to create the Java class in the specified package. For example, type `com.tibco.myjavapackage` for the package name and `HelloWorld` for the class name.

3. Add one or more methods to the class. For example, add a static method, `sayHello`, which echoes a message "Hello World!" when invoked.

```
public static String sayHello(String input){
}
```



You can invoke static or non-static methods using **Java Invoke** activity. For more information about Java Invoke activity, see the *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference* guide.

4. Add the implementation for the methods. For example, add the following implementation code to the `sayHello` method as shown:

```
public static String sayHello(String input){
 return "Hello " + input;
}
```

After implementing Java methods, you can proceed to design the process in the Process Editor.

5. Open the process in the Process Editor where you want to invoke the Java method and add a **Java Invoke** activity from Java Palette. Add transitions to the activity as required.
6. Configure the **Java Invoke** activity from the Properties view of the activity as described.
  - Click **Browse** in front of the **Class Name** field. In the Class Selection dialog, type the first few letters of the class name to search for the class you want to access. From the list of matching items, select the class you want to access. For example, select `HelloWorld`. Click **OK**.
  - From the drop-down list, select the method you want to invoke. For example, select `sayHello`.
  - If the method requires input parameters, provide the values for the input parameters from the **Input** tab of **Java Invoke** activity. For example, in the `sayHello` method, add the string "World!" to the input parameter.
7. Complete configuring your process and map the inputs for the activities as required. Then save the process. You can run or debug the application module in TIBCO Business Studio™ for BusinessWorks™ and verify the output of the **Java Invoke** activity.

## Accessing Module Properties from Java Global Instance

You can access module properties from Java Global Instance so that at the time of deployment, these properties can be configured.

To access the ActiveMatrix BusinessWorks 6.x Module Properties in a user-defined Java code referenced in Java Global Instance, follow these steps:

### Procedure

1. In the ActiveMatrix BusinessWorks 6.x module, specify a dependency on the package "`com.tibco.bw.palette.shared.java`" using **Import-Package**.
  - a) Double-click **Dependencies** located under **ActiveMatrix BusinessWorks 6.x Module > Module Descriptors**. This opens **BW Manifest Editor**.
  - b) In the **Imported Packages** section, click the **Add** tab to add the dependency on the `com.tibco.bw.palette.shared.java` package.
2. Add the `@ModuleProperties` annotation to the method that accepts only one parameter of type `java.lang.HashMap`. Through this `HashMap` you can access the name or value pair of ActiveMatrix BusinessWorks 6.x module properties.

## Accessing Module Properties from Java Invoke Activity

You can access the ActiveMatrix BusinessWorks 6 module properties and Java system properties from the user-defined code invoked from the Java Invoke activity and Java Event Source.

## Procedure

1. Under the ActiveMatrix BusinessWorks 6 module, click **Module Descriptors** , and then double-click **Dependencies**.  
This opens BW Manifest Editor.
2. In the **Imported Packages** section, click **Add**.  
The Package Selection dialog opens.
3. Select the **com.tibco.bw.palette.shared.java** and **com.tibco.bw.runtime** package and click **OK**.
4. Add the **@BWActivityContext** annotation to the method which accepts only one parameter of type **com.tibco.bw.runtime.ActivityContext**.  
The module property can be accessed from ActivityContext class using the methods "registerModuleProperty" and "getModuleProperty".

## Accessing Module Properties in User-Defined Java Code Referenced in JavaProcessStarter

Retrieve EventSourceContext from the `getEventSourceContext()` method of abstract Java class "JavaProcessStarter". The module property can be accessed from EventSourceContext class using the methods "registerModuleProperty" and "getModuleProperty".




# Creating an Application

An application is a collection of one or more modules and can be executed in the runtime. Modules are packages containing one or more processes, shared resources, and metadata such as name, version, dependencies, and so on. Package names must be unique within an application. If there are two packages with the same name in an application, then you must either rename one of the packages or remove one of the packages from the application.



An application created using TIBCO ActiveMatrix BusinessWorks Express can run in TIBCO ActiveMatrix BusinessWorks Enterprise. However, an application created using ActiveMatrix BusinessWorks Enterprise cannot run in ActiveMatrix BusinessWorks Express.

The New BusinessWorks Application wizard helps create an application. There are multiple ways to launch the wizard:

- From the main menu, select **File > New > BusinessWorks Resources** and then select  **BusinessWorks Application**.
- From the **Module Descriptors > Overview** getting started area, click  **Create a BusinessWorks Application**.
- Right-click in the Project Explorer view and select **New >**  **BusinessWorks Application**.

Specify the values for the following fields in the wizard:

1. **Project name:** Name of the application.
2. **Use default location:** Specifies the location on disk to store the application's data files. By default, this value is set to the workspace. To change, clear the check box and browse to select the location to be used.
3. **Version:** Version of the application.
4. **Deployment Target:** Select the required deployment platform(s).
 

Optional. You can set the default deployment profile to create applications, and migrate the existing TIBCO ActiveMatrix BusinessWorks™ 5.x projects with the set preference. Navigate to **Window > Preferences > BusinessWorks > Deployment Profile**.
5. **Create Application Module:** Selected by default to create an application module with the specified name. Clear the check box if you do not want to create an application module.
6. Click **Finish**.

## Result

An application with the specified name is created and opened in the workbench. If the option to create an application module was selected, the application module with the specified name is also created.

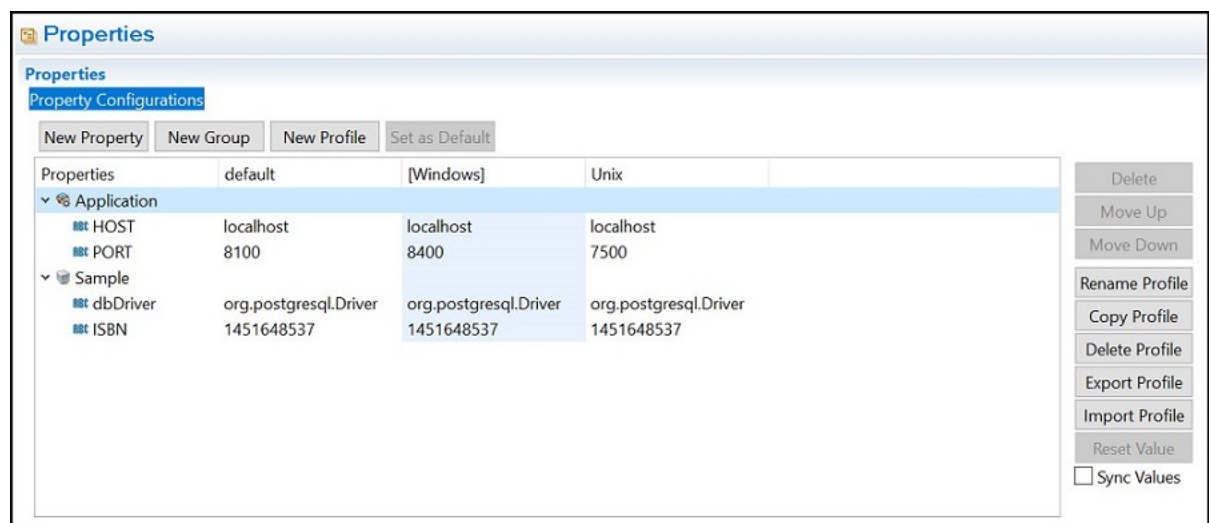


# Working with Application Properties

## Creating an Application Property

You create new application properties in the **Properties** page of an application in TIBCO Business Studio™ for BusinessWorks™. The application properties editor can also be used to create and manage custom profiles. You can also promote a module property to be an application property.

The application properties editor can be used to create and manage new application profiles. You can add a new application property to a profile, rename a profile, and also delete a profile. Application profiles can also be exported from an application, and used in a different application. Similarly, exported profiles can be imported into a different application.



Follow these steps to create a new application property:

### Procedure

1. In the Project Explorer, fully expand `<application_name>.application`.
2. Double-click **Properties** under **Package Unit** to open the application properties in the right pane.
3. Click **Application** in the **Properties** column and click **New Property**.  
A new property gets created under **Application**.
4. Click the property name to edit it.
5. Click the corresponding default profile column (the default column is indicated with [ ] around it) or another profile column in case you have multiple profiles set up for the property to enter a value for the property.

For more information about setting an application profile as a default profile, see "Setting the Default Application Profile" in the *TIBCO ActiveMatrix BusinessWorks™ Samples* guide.

6. Save your application.



Be sure to map a module property to the newly created application property. Keep in mind that an application property can only be used when mapped to a module property.

For information on creating profiles see [Creating an Application with Multiple Profiles](#).

When a property value is same across all profiles, and you want to keep the property values consistent across all profiles after updating a value for any one profile, select the **Sync Values** check box .

For more information about editing application properties using bwadmin command line utility or by using Admin UI, see "Editing Application and Application Instance Properties" in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

## Exporting an Application Profile

An application profile can be exported from the application. After an application is configured with a profile, it becomes part of the application archive. The exported application profile can be used to configure another application by importing it into that application.

### To export an application profile:

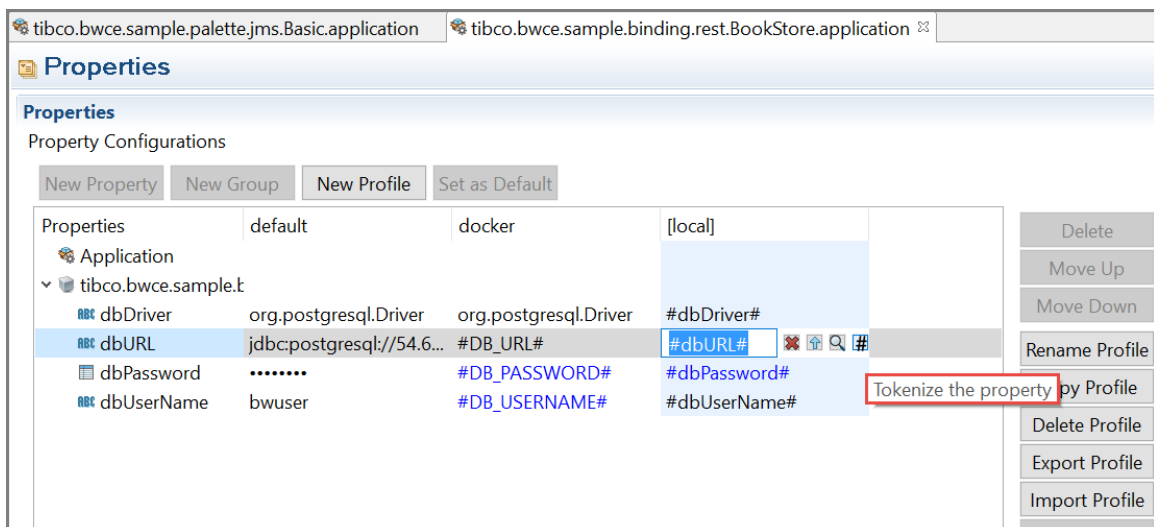
Do the following to export an application profile:

1. Click the **<Project>.application**, expand **Package Unit** folder and double-click **Properties** to open the Properties page.
2. Click the profile name of the profile you want to export to select it and click the **Export Profile** button.
3. Select the application and use the **Browse** button to browse to a location where you want to download the profile file.
4. Click **Finish**.

A `<profile-name>.substvar` file gets created in the location you specified. You can now import this profile file into another application.

## Tokenizing Application Properties for exporting in the Properties file

To tokenize application properties, a new button **Tokenize the property** is added in the Properties view for application properties.



After tokenization, the property value is set in the format `#property_name#`. Once the user tokenizes a property, the original default value for the property is lost.

The **Tokenize the property** button is available only for applications with the Deployment Target set as Container. When you open **Export Profile** wizard, the tokenized values are already selected.

For more information about how to export tokenized properties to Consul server, see "Exporting Tokenized Properties to Consul Server from TIBCO Business Studio™ for BusinessWorks™" topic in the *TIBCO BusinessWorks™ Container Edition Application Development* guide.

Follow the steps to tokenize the application or module properties and export them in properties files.

### Procedure

1. Double click the application properties.
2. In the Properties view, select the profile, and click the **Export Profile** button. The **Export Profile** wizard opens.
3. In the **Export Profile** wizard, select the properties to be exported.
4. Select the check box **Export as properties file** and browse the location to export the properties file. The properties are available in the `.properties` file in the form of key-value pairs.

```

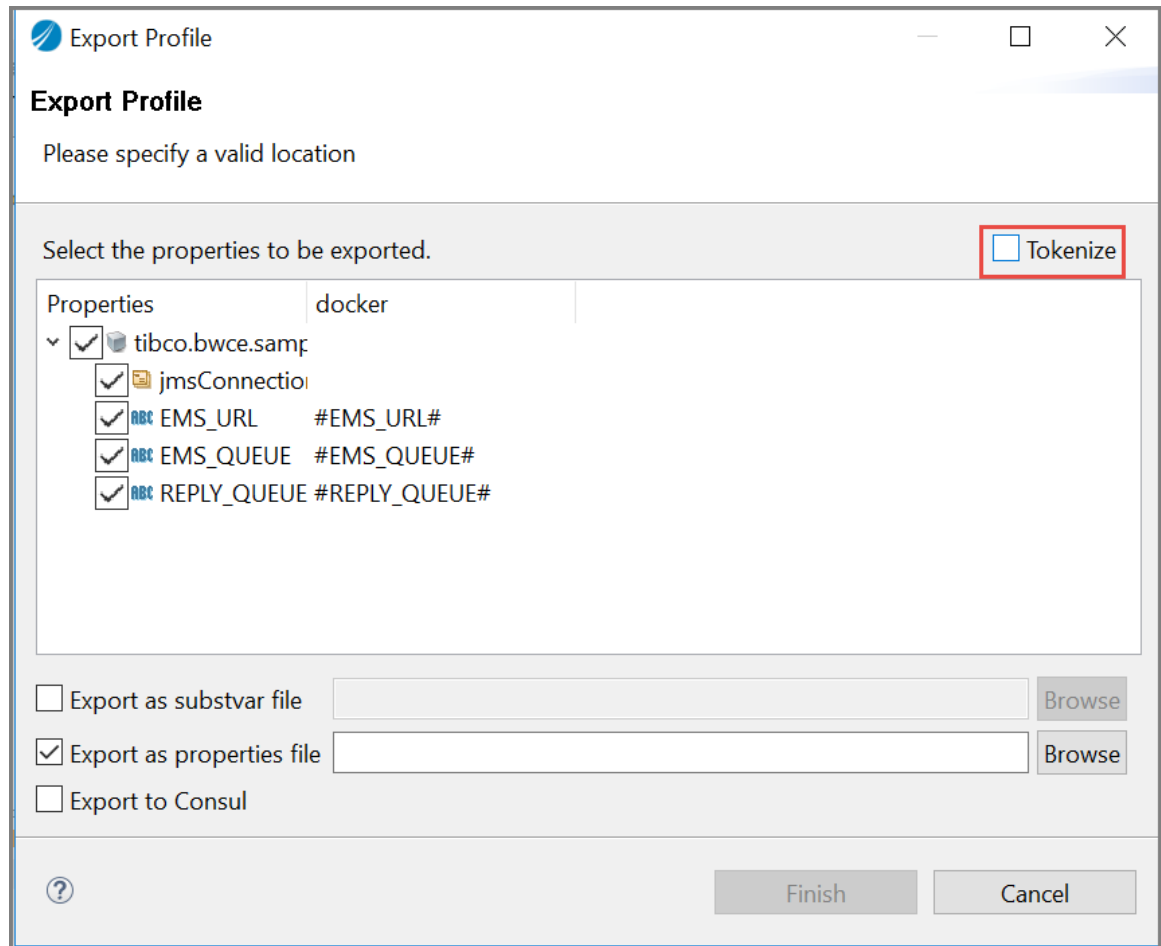
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
tibco.bwce.sample.binding.rest.BookStore.application-local.properties
1 #Tue May 14 14:52:30 IST 2019
2 dbUserName=bwuser
3 dbURL=jdbc:postgresql://54.67.121.175:5432/postgres
4 dbDriver=org.postgresql.Driver
5 dbPassword=#!GGJ+JL881ZfUCyq9BUmk92uzoH5HOM2XBF+Rv+JtedM=
6

```

The properties that are selected while exporting the profile for Properties file will be generated as key-value pairs in the `.properties` file. The keys in the Properties file is generated using the names of the properties that are exported.

After the property is exported, it is auto-tokenized in TIBCO BusinessWorks™ Studio depending on the deployment target selected for the application project.

- If the selected deployment target is Container, then the properties are auto-tokenized.
- If the selected deployment target is Tibco Cloud, then the properties are not tokenized.
- If the selected deployment target is both Container and Tibco Cloud, the **Tokenize** check box is enabled in the **Export Profile** wizard. On selecting the **Tokenize** checkbox, the properties are tokenized.



This functionality supports properties of type boolean, string, integer, long, and password.

The following table lists the values of five different types of properties if the default values are provided for all the properties

| Data Type | Property Name | Before Tokenization | After Tokenization | Before Export  | After Export in the properties file |
|-----------|---------------|---------------------|--------------------|----------------|-------------------------------------|
|           |               | <b>Default</b>      | <b>Default</b>     | <b>Default</b> | <b>Default</b>                      |
| String    | Property1     | test                | #Property1#        | test           | test                                |
| Integer   | Property2     | 26                  | #Property2#        | 26             | 26                                  |
| Long      | Property3     | 18                  | #Property3#        | 18             | 18                                  |
| Boolean   | Property4     | true                | #Property4#        | true           | true                                |
| Password  | Property5     | *****               | #Property5#        | *****          | <encrypted value of the password>   |

The following table lists the values of five different types of properties if the default values are not provided for all the properties

| Data Type | Property Name | Before Tokenization | After Tokenization | Before Export | After Export in the properties file |
|-----------|---------------|---------------------|--------------------|---------------|-------------------------------------|
|           |               | Default             | Default            | Default       | Default                             |
| String    | Property1     |                     | #Property1#        |               | Property1                           |
| Integer   | Property2     |                     | #Property2#        |               | 0                                   |
| Long      | Property3     |                     | #Property3#        |               | 0                                   |
| Boolean   | Property4     |                     | #Property4#        |               | false                               |
| Password  | Property5     |                     | #Property5#        |               | <encrypted value of the password>   |

The following table lists the values of five different types of properties if there are all tokenized properties provided for all the properties

| Data Type | Property Name | Before Tokenization | After Tokenization | Before Export | After Export in the properties file |
|-----------|---------------|---------------------|--------------------|---------------|-------------------------------------|
|           |               | Default             | Default            | Default       | Default                             |
| String    | Property1     | #Property1#         | #Property1#        | #Property1#   | Property1                           |
| Integer   | Property2     | #Property2#         | #Property2#        | #Property2#   | 0                                   |
| Long      | Property3     | #Property3#         | #Property3#        | #Property3#   | 0                                   |
| Boolean   | Property4     | #Property4#         | #Property4#        | #Property4#   | false                               |
| Password  | Property5     | #Property5#         | #Property5#        | #Property5#   | a String:<br>PASSWORD               |

## Importing an Application Profile

After exporting an application profile you can import it into another application. Do the following to import an application profile:

### Procedure

1. Click the <Project>.application, expand **Package Unit** folder and double-click Properties to open the Properties page.

- Click the **Import Profile** button.  
The Import Profile dialog box opens.

**Import Profile**  
Wizard for importing a profile.

Location:

Module Property Table:

| Source | Target |
|--------|--------|
|        |        |

Profile Name:

Module Properties to be imported:

| Property | Profile value |
|----------|---------------|
|          |               |

Override existing profile values  
 Match profile exactly (Deletes variables if required).

- Use the **Browse** button to browse to a location of the <profile-name>.substvar profile file you want to import.
- To override existing profile values with the values in the imported profile, select the **Override existing profile values** check box. To keep values from the imported profile only, select the **Match profile exactly (Deletes variables if required)** check box.
- Click **Finish**.  
The imported profile is visible under **Property Configuration**.

## Generating Deployment Artifacts

A deployment artifact is an archive file that contains all the information required to deploy the application to runtime. It is the only artifact that is handed from the design phase to the runtime as it contains all the bundles and metadata that is required to deploy and run the application.

Applications are developed using the features available in TIBCO Business Studio™ for BusinessWorks™ and can range from simple to very complex. An application consists of an application module, which consists of one or more processes that define the business logic, and zero or more shared modules. For more information, see "Application Modules" and "Shared Modules" in the *TIBCO ActiveMatrix BusinessWorks™ Concepts* guide. Applications can also contain OSGi bundles that do not contain ActiveMatrix BusinessWorks™ artifacts.

Applications are developed using the features available in TIBCO Business Studio™ for BusinessWorks™ and can range from simple to very complex.

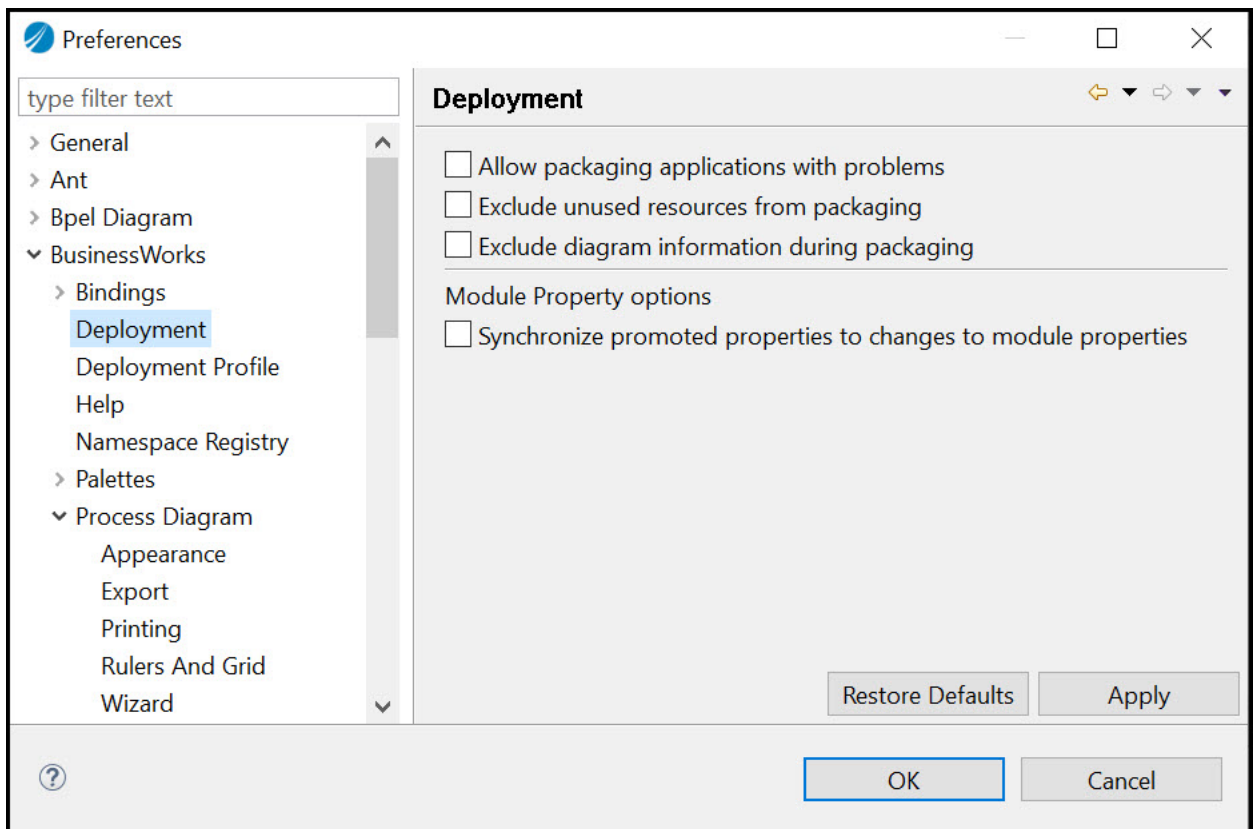
An application archive contains one or more OSGi bundles, one each for all the modules referenced directly or indirectly by the application. It also contains application metadata which is used during deployment.



If any further changes to the design or configurations are made, the deployment artifact (archive file) must be regenerated.

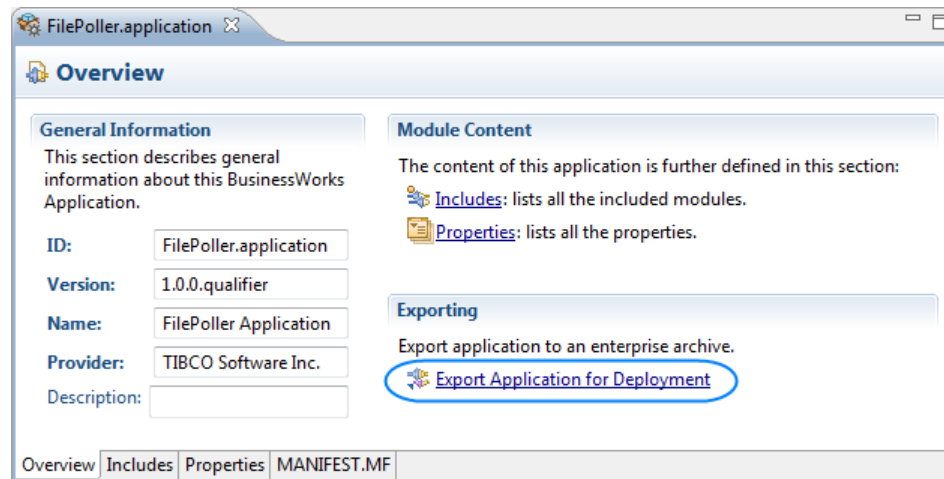
When creating an archive file for an application, the application packager also generates the ActiveMatrix BusinessWorks processes in SVG format, which can be rendered in the Admin UI.

By default, the diagram information is present when exporting a process. To remove diagram information when exporting a process, navigate to the **Window > Preferences > BusinessWorks > Deployment** and select the **Exclude diagram information during packaging** check box.



There are multiple ways to create a deployment artifact:

- From the **Project Explorer** view in TIBCO Business Studio for BusinessWorks, open **<Project>app > Overview** and click **Export Application for Deployment** link as shown below.



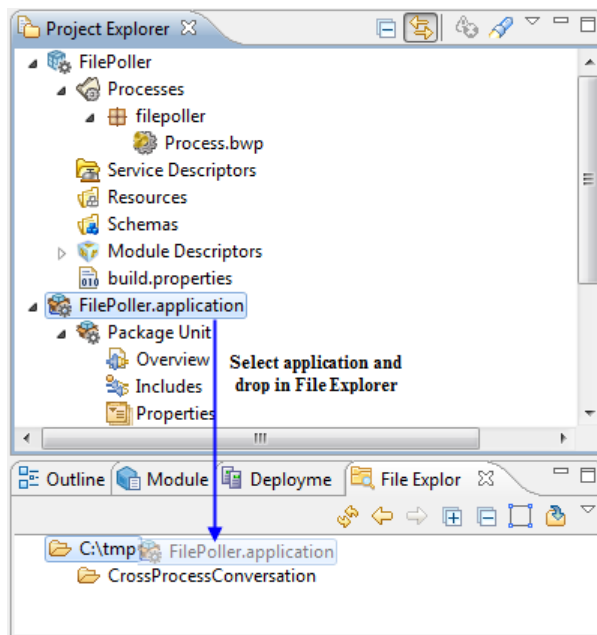
When importing projects created in a version of the software that is lower than ActiveMatrix BusinessWorks™ 6.4.x, if the application module or shared module version does not contain a **.qualifier** version, a design time validation error is thrown by default. Preference options can be set to ignore this validation error. Navigate to **Window > Preferences > BusinessWorks > Validation > Missing .qualifier literal for module version**. Preferences can be set to one of the following options:

- Error: The validation error is displayed in the **Problems** tab.
- Warning: A warning is displayed in the **Problems** tab.
- Ignore: The validation error is not displayed in the **Problems** tab.

In the EAR Export window, specify the location for the archive file and provide a custom name to the archive file, if needed, by clearing the **Use Default EAR file name** check box. Click **Finish** to create the deployment artifact (archive file).

- By selecting the project application in the **Project Explorer** and dropping it in the File Explorer an archive file for the application is created. If needed, change the default location in the File Explorer by using the **Open Directory to Browse** option in the File Explorer and select a custom folder. For example `c:/tmp`.





Artifacts intended to be exported from a shared module must be contained in their respective special folders. For example, schemas must be contained in the **Schemas** folder, WSDL files in the **Service Descriptors** folder, and processes must be contained in a package under the **Processes** folder.

When you deploy an application, each application in an AppSpace is identified by its unique name and a *major.minor* version number. The version number is important as it provides traceability and helps troubleshoot in case of an error at run time. If any further modifications are made to the application, the archive file must be regenerated with an updated version number and then deployed to the AppSpace.

When you deploy and start an application, if the archive file contains the SVG format of business processes, you can view the process diagrams for the processes from the Admin UI. For more information, see the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

# Deploying an Application

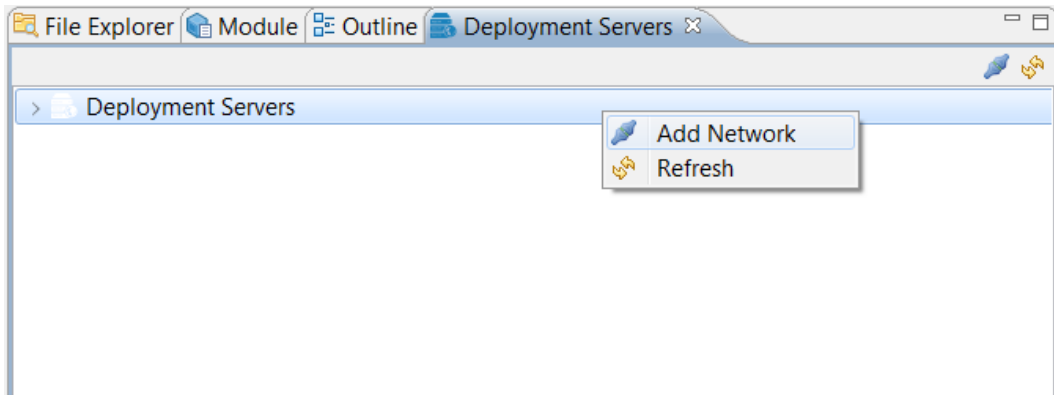
You can deploy an application in TIBCO Business Studio™ for BusinessWorks™.

## Prerequisites

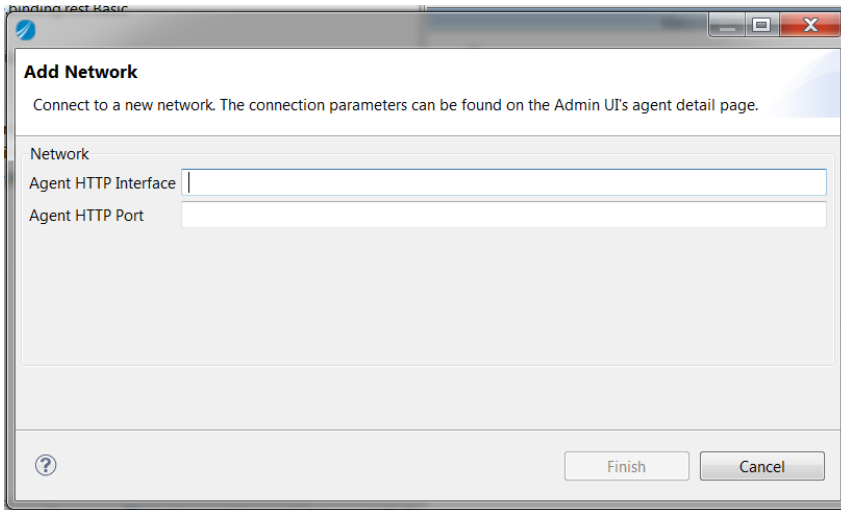
The bwagent must be running. For information on runtime entities, see the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide. If no network exists, you can create one by entering values in the Add Network dialog box and creating runtime entities in the Deploy Application dialog box.

## Procedure

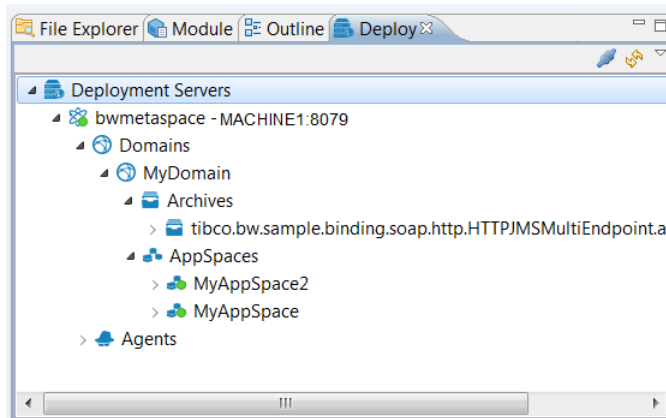
1. Connect to a deployment server.
  - a) In the Deployment Servers pane, right-click Deployment Servers and select **Add Network**.



- b) In the Add Network dialog, specify the HTTP interface and port for the network. The default HTTP interface is the name of the bwagent. The default port is 8079. The defaults might have been changed by your administrator.

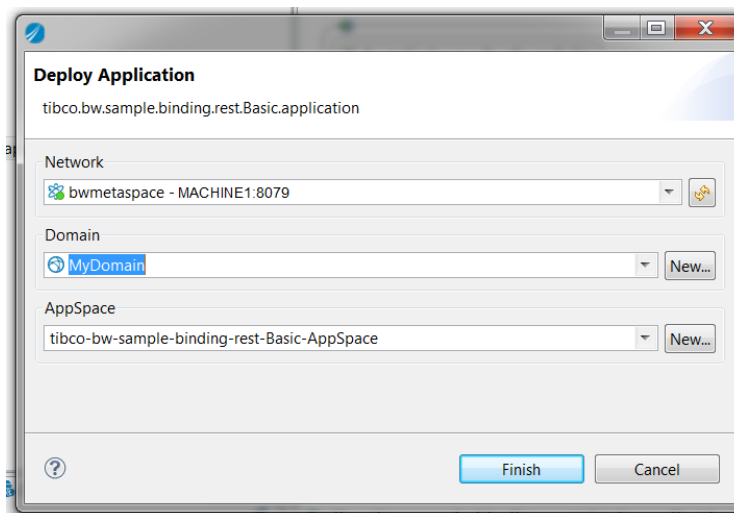


The selected network is displayed in the **Deployment Servers** pane:



The **Deployment Servers** pane contains information related to a single bwagent. Unlike the Admin UI, this pane does not have information about the other bwagents and hence cannot be used to perform operations on other bwagents. For example, you cannot create an AppNode in another bwagent within the existing domain.

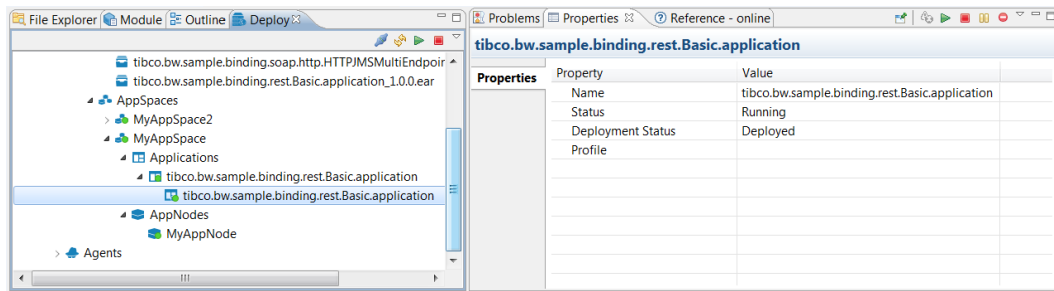
2. Deploy and start your application. There are several ways to deploy:
  - Drag an application project into an AppSpace in the **Deployment Servers** pane and drop it.
  - Drag an archive file from the **Project Explorer**, from Mac Finder, or from Windows File Explorer into an AppSpace in the Deployment Servers pane.
  - Drag an archive file from the Archives folder (in the Deployment Servers pane ) into an AppSpace and drop it.
  - Right-click the application and choose **Deploy Application** When the arget platform is updated, export the EAR file and deploy it to the selected platform. Options such as
    - a) In the Deploy Application dialog box, choose:
      - **Network:** The network to deploy to.
      - **Domain:** The domain to upload to. If there are no domains in the network, a default value is provided. Click **New** to create a new domain.
      - **AppSpace:** The AppSpace to deploy to. The default AppSpace name is based on the application name. Click **New** to create a new AppSpace in the selected domain.



If an AppNode does not exist, an AppNode is created. The AppNode is computed from the AppSpace name and an HTTP management port value is assigned.

The application is deployed.

3. Right-click the application in the **Deployment Servers** window and choose to **Start** it. Select it in the **Deployment Servers** pane to view the status in the Properties pane:



4. Right-click the application in the **Deployment Servers** window and choose **Stop** to stop it.

# Refactoring a Shared Resource or Policy Package

---

Follow these steps to refactor a resource or policy.

## Renaming a Resource or a Policy Package

Follow these steps to change the name of a resource, or a policy, package and to update its corresponding references in the project.

### Procedure

1. To rename a resource package, right-click the package under the Resources folder and select **Refactor > Rename Resource Package**. To rename a policy package, expand the Policies folder, right-click the policy package, and select **Refactor > Rename Policy Package**.
2. Enter a new name for the resource, or policy, package and select **OK**.
3. Optional. In the Rename Package Name dialog, confirm the changes that will happen, and the resource references that will be updated, by ensuring the correct resources are selected.
4. Select **OK**.

## Changing the Location of a Resource or a Policy

Follow these steps to change the location of a resource, or a policy, and to update its corresponding references in the project.

### Procedure

1. To update the location of a resource, right-click the package under the Resources folder and select **Refactor > Rename Resource Package**. To update the location a policy, expand the Policies folder, right-click the package containing the policy you want to rename, and select **Refactor > Rename Policy Package**.
2. Enter a new location for the resource, or policy, and select **OK**. For example, to change the location of a Basic Authentication policy residing in `refactoringproject.TestPackage`, modify the package name to `refactoringproject.NewPackage.TestPackage`.
3. Optional. In the Rename Package Name dialog, confirm the changes that will happen, and the resource references that will be updated, by ensuring the correct resources are selected.
4. Select **OK**.  
A new folder structure under the Resources, or Policies, folder is created, and the resource is moved to the newly created location.

## Working with Multiple Component Processes

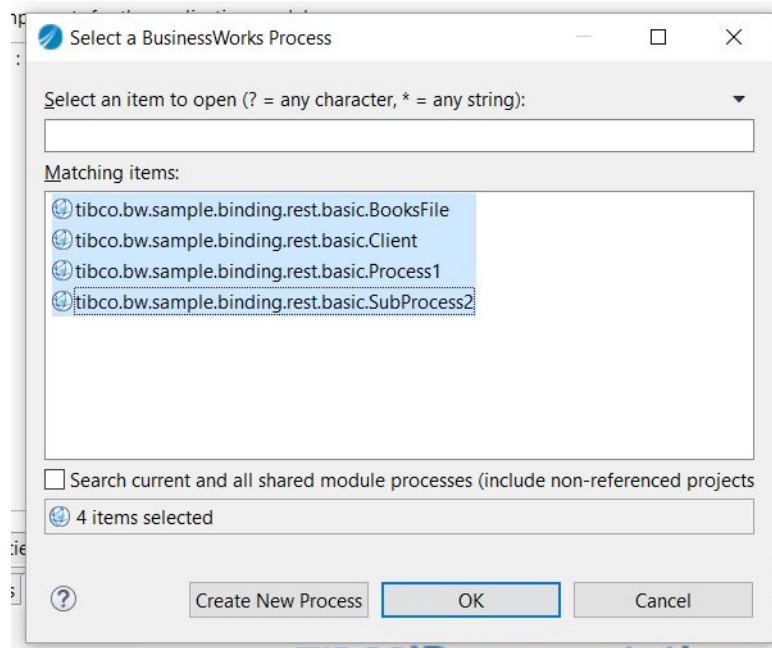
Using the Components editor in TIBCO Business Studio™ for BusinessWorks™, perform tasks such as selecting or deselecting components, adding or removing components.

### Adding Multiple Component Processes

Follow these instructions to add more than one component process to an application.

#### Procedure

1. To open the Components editor in TIBCO Business Studio for BusinessWorks, in the **Project Explorer**, navigate to the `Module Descriptors` folder, and double-click the `Components` folder.
2. Click the **Create Process Component**  icon.
3. From the Select a BusinessWorks Process wizard, select a component process, hold down the **Shift** button, and use the up or down directional buttons on your keyboard to select additional processes.



4. Click **OK**.

#### Result

The processes you specified are displayed in the Component Process editor.

### Deleting Multiple Component Processes


Follow these instructions to remove more than one component process to an application.



Component processes with REST services cannot be deleted from the Components editor. Instead, delete them from the Project Explorer.

#### Procedure

1. To open the Components editor in TIBCO Business Studio for BusinessWorks, in the **Project Explorer**, navigate to the `Module Descriptors` folder, and double-click the `Components` folder.

2. Select a component process, hold down the **Shift** button, and use the up or down directional buttons on your keyboard to select additional processes.
3. Click the  icon to remove the component processes you selected.
4. Click **OK**.

## Result

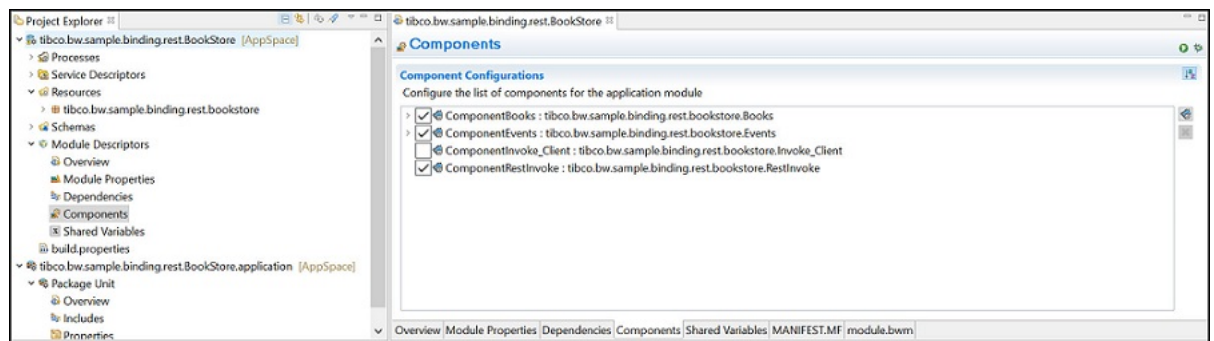
The processes you specified are removed, and no longer display in the Component Process editor.

## Enabling Auto Start of Component Process

Select or clear the check boxes for each component process to decide whether to enable auto start of a component process or not during execution. If you clear the check box, corresponding component process is not executed during application startup.

### Procedure

1. To open the Components editor in TIBCO Business Studio for BusinessWorks, in the **Project Explorer**, navigate to the **Module Descriptors** folder, and double-click the **Components** folder.



2. Clear the check box for corresponding component processes.
3. Click **Save**.



For more information about how to enable or disable components from Admin UI, see "Starting a component in an Application" and "Stopping a component in an Application" in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

## Analyzing Dependencies and References

The Dependency Visualizer provides graphical visuals of all the direct and indirect dependencies and references for an application. You can use this option to view the hierarchy of processes, shared resources, WSDL files, XSD files and so on. Dependency Visualizer can also be used to explore the dependencies and references for a selected resource and all the objects within a workspace.

The Dependency Visualizer provides information on how an application is organized and also helps identify potential problems and possible improvements in the application.

Dependency and reference information can be viewed for the following:





- Applications
- Application Modules
- Shared Modules
- WSDL files
- XSD files
- Processes
- Shared Resources

All the information to be displayed can be configured as required by moving the nodes on the canvas.

These changes can also be restored to the original settings by selecting the **Refresh** icon .

The information can be configured using the **Change Graph Layout** option . The layout options available are:

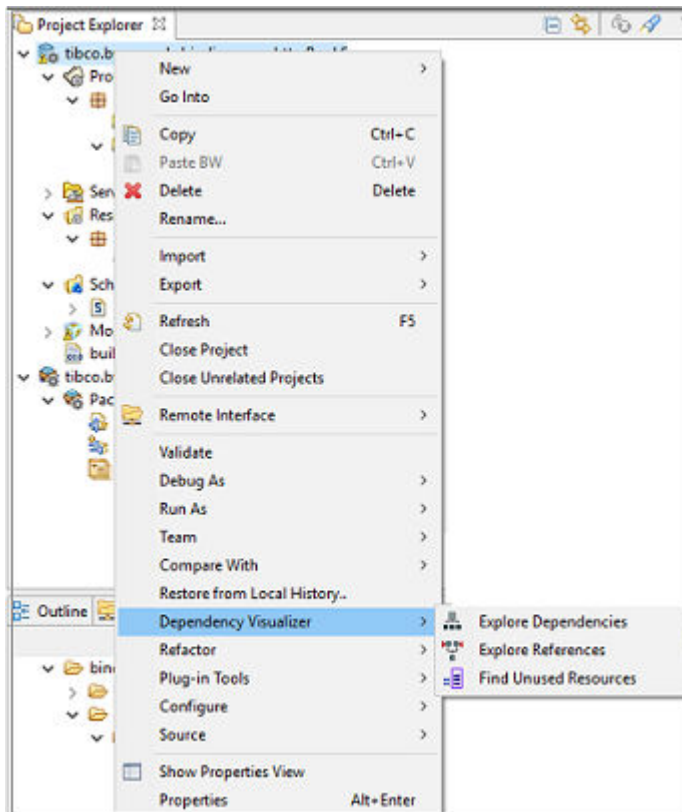
- Spring Layout
- Tree Layout
- Grid Layout
- Horizontal Tree Layout
- Radial Layout

Other additional options available are Take a Screenshot , Search a Node in graph , Highlight Dependencies/References  and Zoom In/Out .

### Exploring Dependencies

To access the Dependency Visualizer option from TIBCO Business Studio™ for BusinessWorks™, navigate from the right-click menu of the application or the shared resource to **Dependency Visualizer > Explore Dependencies**. The dependency graph is displayed.

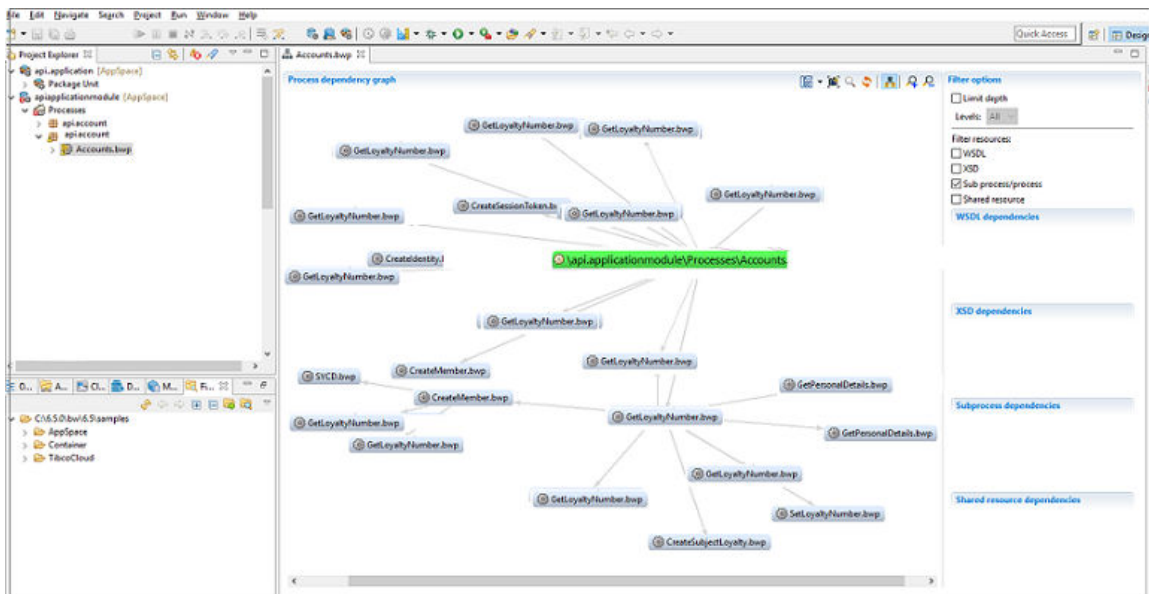




The dependencies between application modules and shared modules can be viewed and analyzed. The Project Dependency Graph shows the inter-dependency of modules for the selected Application, Application Module and Shared Modules.

The viewer window displays all objects of the same type as the current selection. For example, for a selected project, all the projects in the workspace are displayed and all the projects related to the current selection are highlighted.

The dependencies for the selected resource is displayed in the following way:



In the above example, the **Process Dependency Graph** for Account.bwp is displayed. The root node, Account.bwp is first highlighted in green. On selection, the root node highlights its level 1 dependencies. The **Levels** dropdown option contains the level options 1, 2, 3, 4 and All. You can use

this filter to limit the depth of the levels to be displayed. When any of the level 1 dependencies are selected, the level 1 dependencies of that selected node are highlighted.

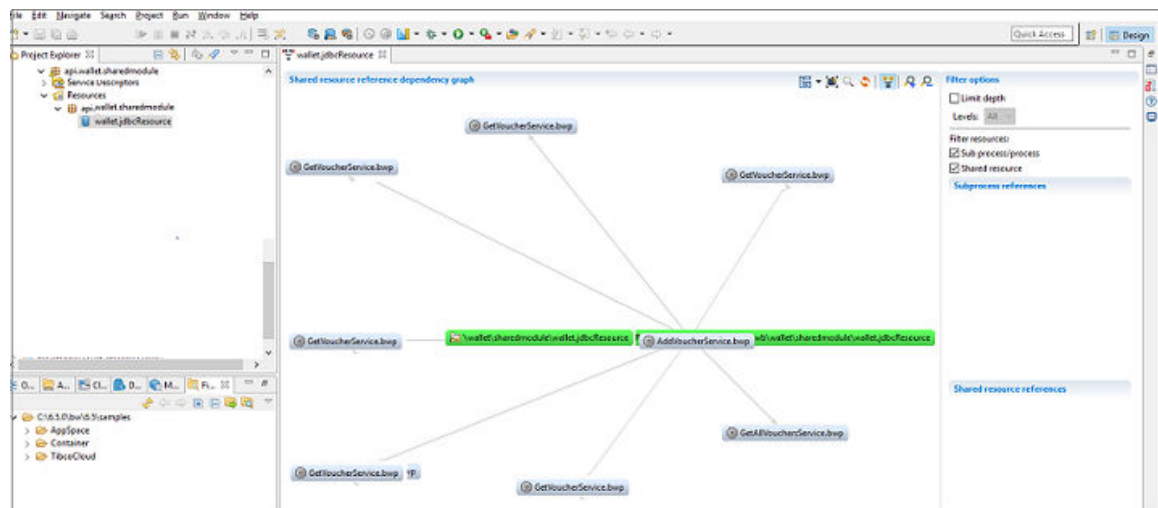
The required dependent nodes can also be highlighted using the option **Highlight Dependencies**.

When you click the root node or any of the level dependencies, the dependencies are displayed in the categories listed below. You can use the **Filter Resources** filter to refine the view to display only the required dependencies.

- WSDL Dependencies
- XSD Dependencies
- Subprocess Dependencies
- Shared Resource Dependencies

### Exploring References

Navigate from the right-click menu of the application or the shared resource to **Dependency Visualizer > Explore References**. The **Reference Dependency Graph** displays where the selected resources are referred. In the example in the image below, the **HTTP Connection** shared resource is referenced by the following processes.



The required referenced nodes can be highlighted using the option **Highlight References**.

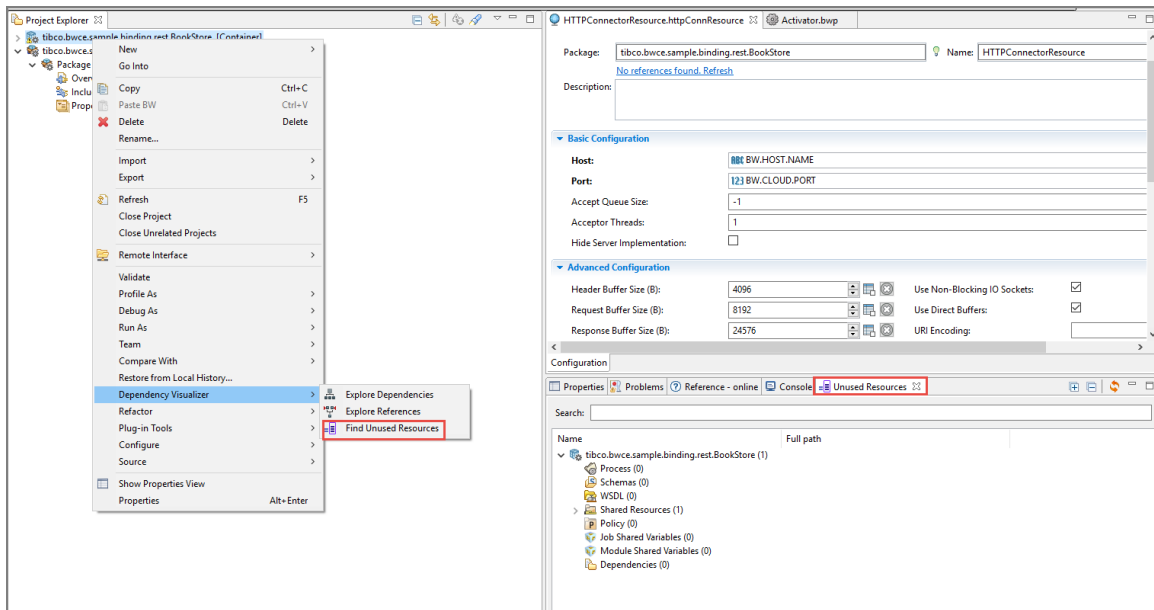
When you click the root node or any of the level references, the references are displayed in the following categories:

- WSDL References
- XSD References
- Sub process/processes References
- Shared Resource References

### Unused Resources

The view displays unused resources from selected module and its dependent modules.

To display a module's unused resources, right-click the module and select **Dependency Visualizer > Find Unused Resources**. The **Unused Resources** view is displayed.




On the **Unused Resources** view, the total number of unused resources is displayed. This includes:

- Process
- Schemas
- WSDL
- Shared Resource
- Policy
- Job Shared Variables
- Module Shared Variables

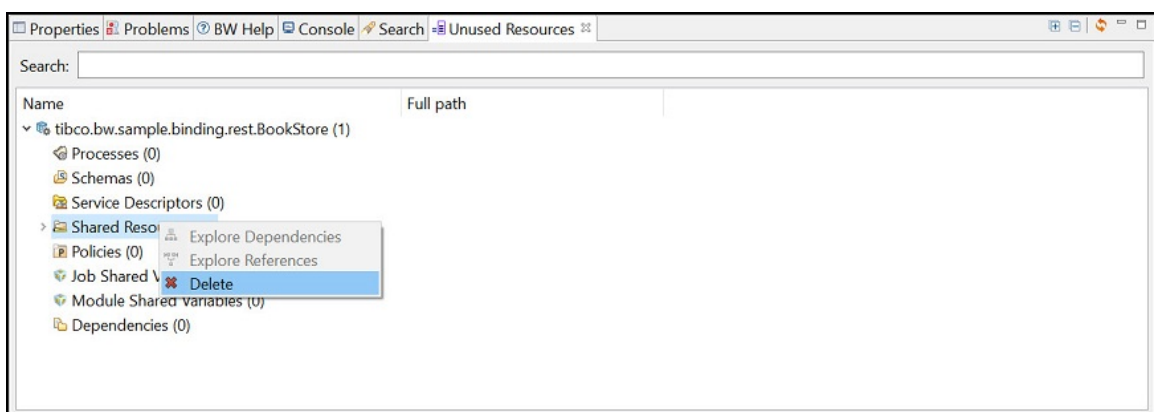
The **Include Application Modules** check box is now removed from the **Unused Resources** view. Implicitly, application modules are included during unused resources calculation. When calculating unused resources for application modules, associated shared modules are also considered.

If you add, remove, or re-factor unused resources in the application and you need to find unused

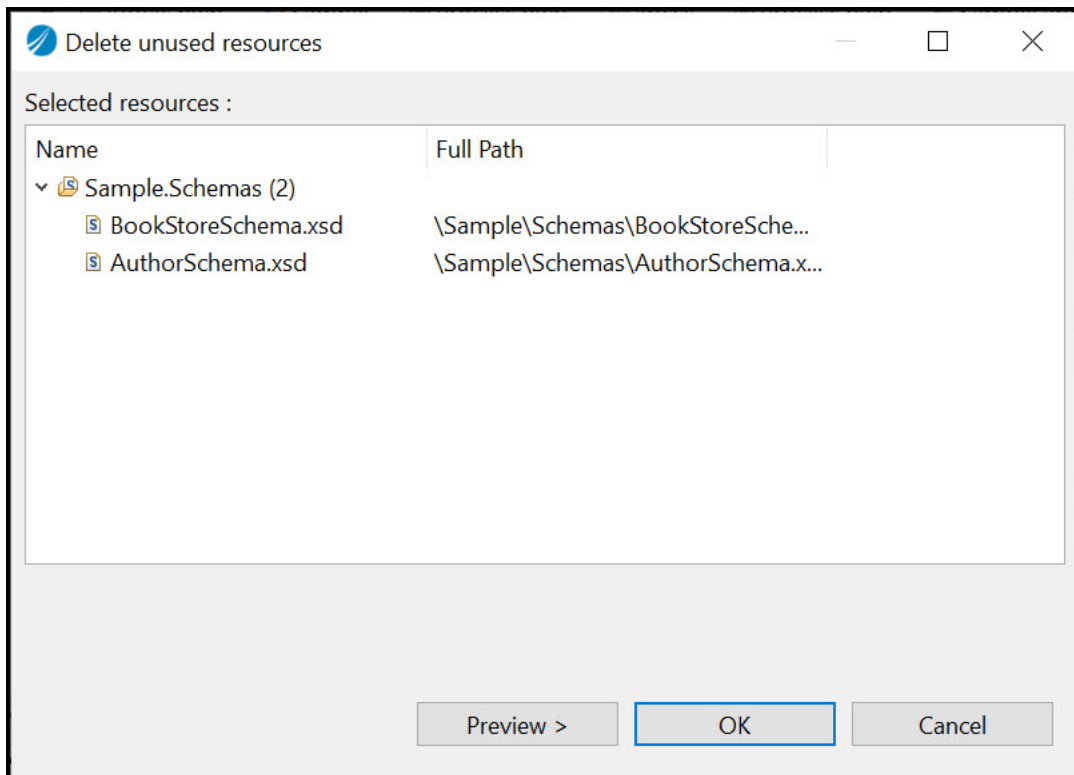
resources again, use **Refresh**  button in the **Unused Resources** view to get the updated unused resources. Refresh operation takes some time when using with large projects.

If you remove or close any project from the **Project Explorer** view, the **Unused Resources** view is cleared.

To delete unused resources by type, select the resource and right-click > **Delete**.

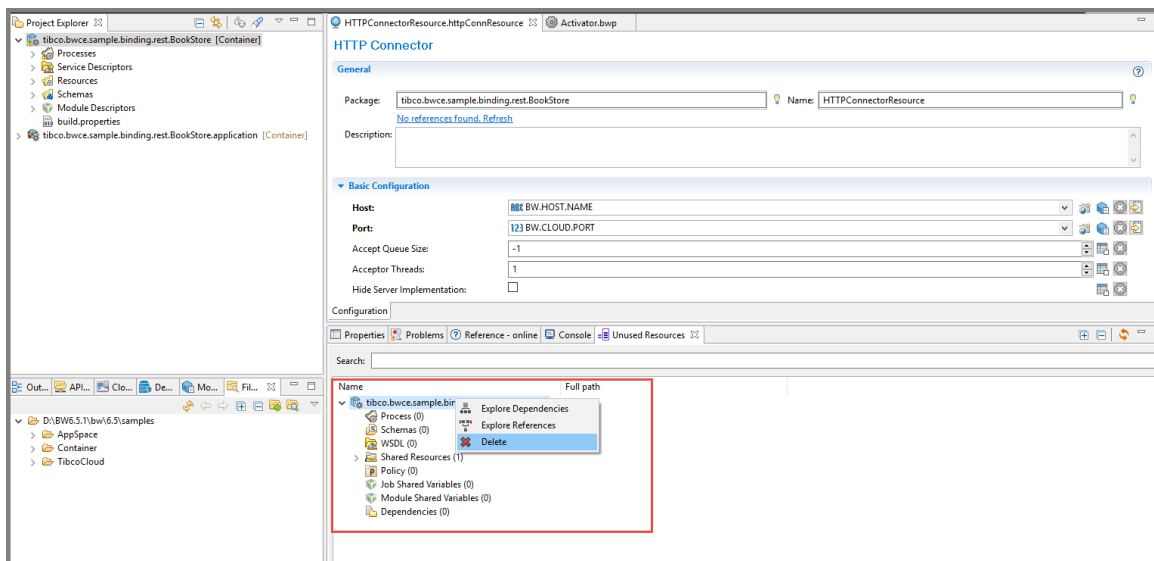


When you try to delete an unused artifact which has dependencies as well as references on the other unused artifacts, a new Delete unused resources dialog box opens.



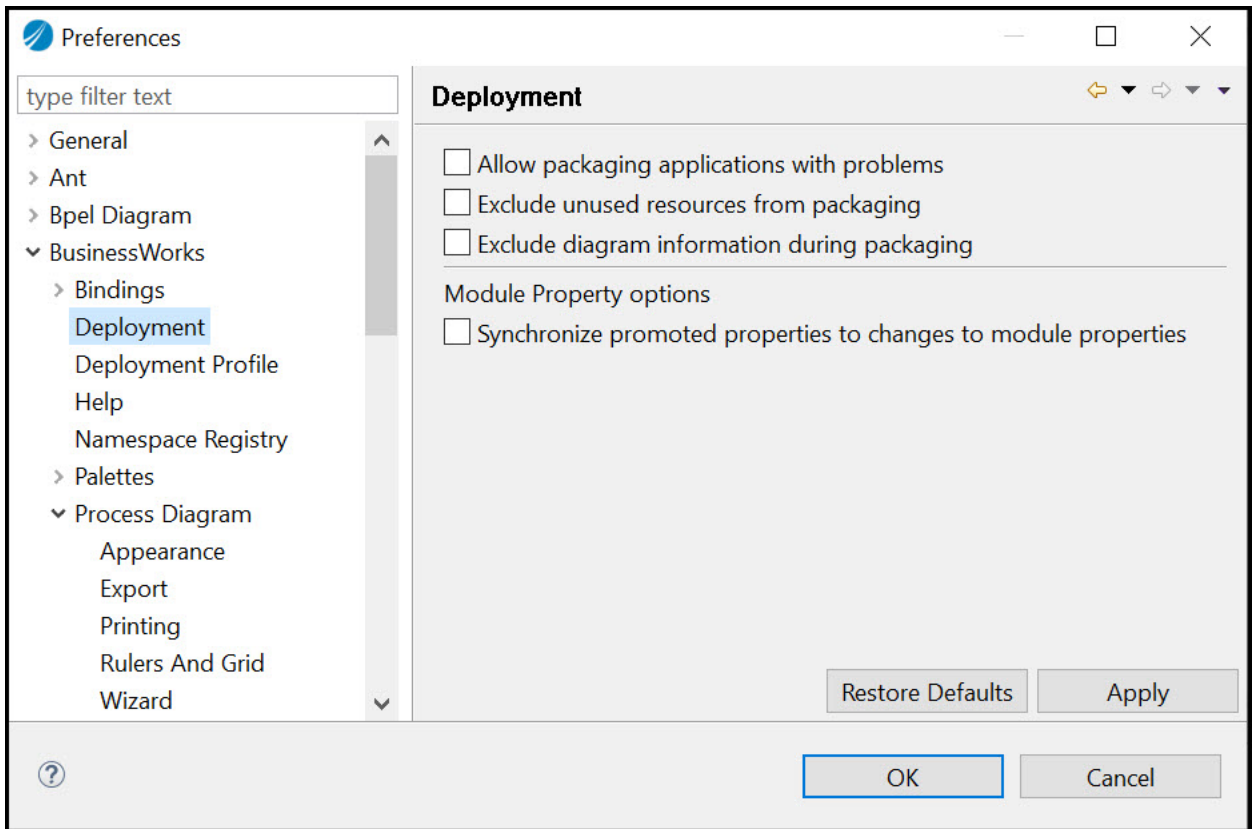
It shows the list of selected resources. To see the list of corresponding dependencies, click **Preview** button. To delete dependent resources, click **Ok**.

To delete all the unused resources together in one go, select all the resources and right-click > **Delete**. However, you need to repeat this process more than once to remove all unused resources.

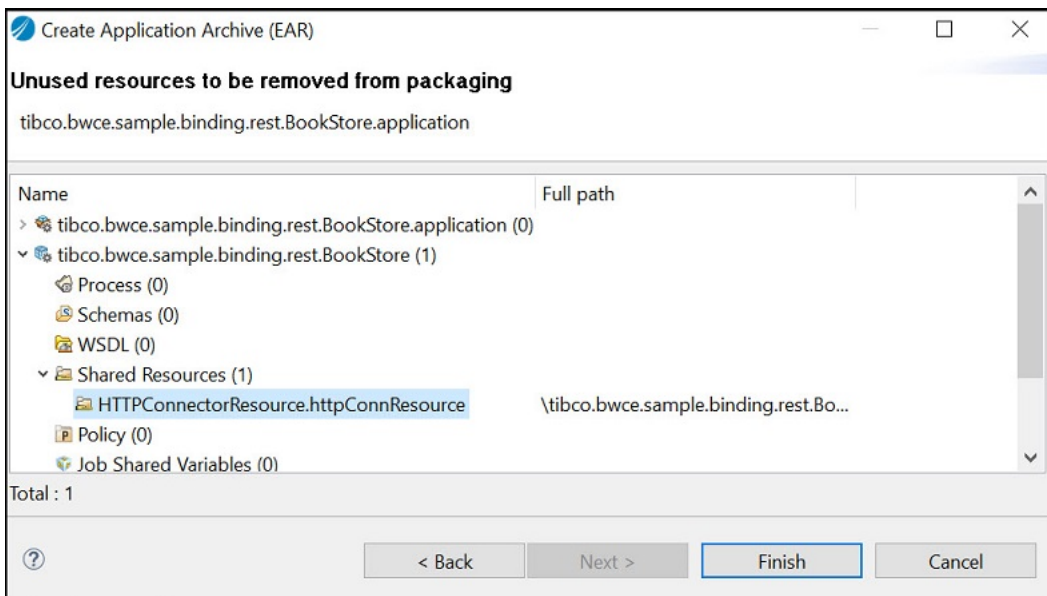


### Generate EAR without unused resources

To generate an application archive file (.EAR) without unused resources, select the **BusinessWorks** > **Deployment** > **Exclude unused resources from packaging** check box in the Preferences dialog box.



When you create an .EAR file, preview of unused resources is displayed in the Create Application Archive wizard.



An .EAR file is generated without unused resources.

From CLI, use `-removeunused` argument for `export` command in the `bwdesign` utility. For more information, see [Using bwdesign utility](#).

## Repairing TIBCO ActiveMatrix BusinessWorks™ Projects

Repairing the BusinessWorks projects is one of the refactoring activities available on right-clicking the project in the Project Explorer pane, and clicking **Refactor > Repair BusinessWorks Projects...**

The Repair BusinessWorks Projects... option is used to update data models in the selected files. When a repair tool is executing, its logic is applied to the selected files. If there is no data to update, the repair tool does not make any changes. So, it is fine if you run the tool multiple times.

For the existing projects with data models in an old data format, the repair tool can upgrade the data models with a new data format. For the existing projects with defects, for example, some data is missing, the repair tool can recover or regenerate the missing data. When a new feature is added, the existing projects do not have the serialization, which is provided by the new feature. In this scenario, the repair tool can apply the new feature and generate the corresponding serialization in the existing projects to enable the new features.



The term existing projects includes the projects which are migrated from ActiveMatrix BusinessWorks 5.x to 6.x, and the projects created in ActiveMatrix BusinessWorks and before ActiveMatrix BusinessWorks 6.5.1.

Following is the Repair BusinessWorks Projects wizard.

Repair BusinessWorks Projects

Select the BusinessWorks Projects:

- test2

Options:

- Update the inline schemas of the Processes
- Update activity error variables
- Update activity input copyOf variable
- Update memory saving variables
- Remove memory saving variables
- Refresh Project Cache and do Project Clean

Preview > OK Cancel

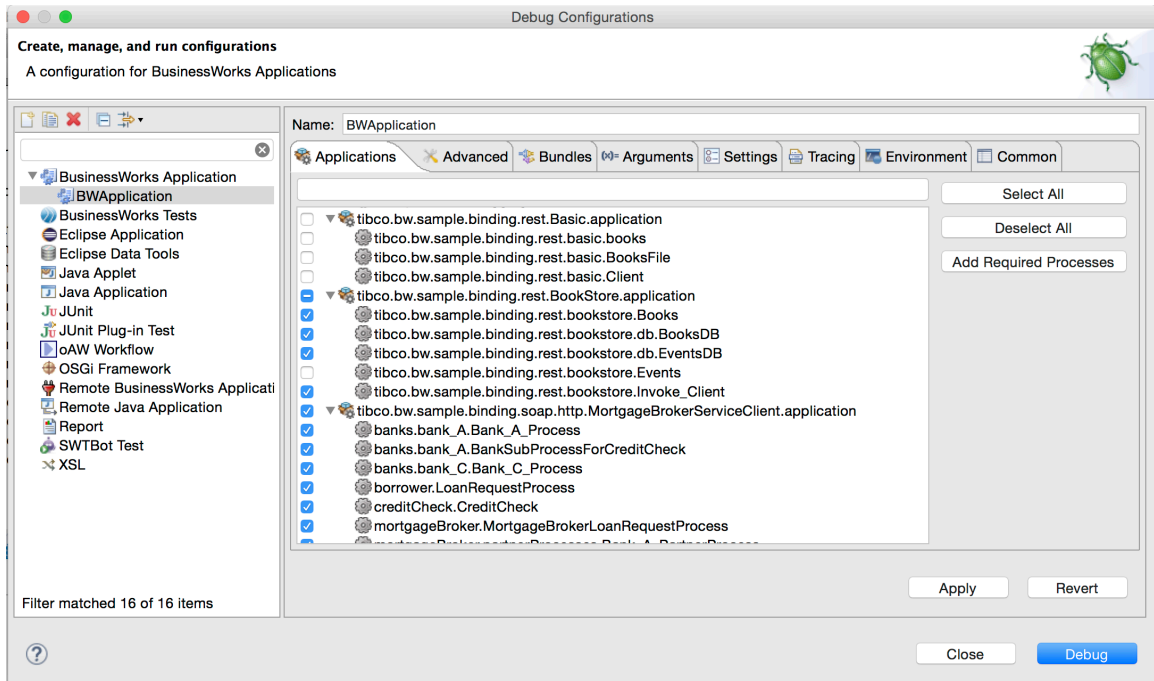
The following table describes the options available in the Repair BusinessWorks Projects wizard.

| Field                                      | Description                                                                                                                                                                                               |
|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Update the inline schemas of the Processes | To update the Engine type inline schemas in the existing projects with new format                                                                                                                         |
| Update activity error variables            | To create an extra error variable for the activities in the existing projects                                                                                                                             |
| Update activity input copyOf variable      | To update the <i>tibex:copyOf</i> extension attribute for the existing projects                                                                                                                           |
| Update memory saving variables             | To calculate the variables to be freed after an activity is executed at the run time in the existing projects                                                                                             |
| Remove memory saving variables             | To remove already serialized memory saving variables from various activities                                                                                                                              |
| Refresh Project Cache and do Project Clean | To reload the project cache and working copy of resources in the projects. For such repair tool, there is no validation error associated with the projects and no changes can be made in the data models. |



# Using the Debugger

The debugger enables different configurations of an application to be run in design phase.






By default, the debugger lists all the process and sub processes of an application module, shared module and nested shared module in the debug configuration window. You can select applications, and processes in an application, to launch in the debugger.














The Debug perspective consists of set of views which are related to the debugging task. Some views, for example the Project Explorer view, are not available in the Debug perspective, while others, such as Debug and Breakpoints, are available in the Debug perspective. There are multiple ways to open the Debug perspective:

- From the main menu, select **Window > Open Perspective > Other** and then select  **Debug**.
- From the **Module Descriptors > Overview** Testing area, click  **Launch BusinessWorks Debugger**.

The Debug perspective consists of the following views, starting from the upper left corner clockwise:

- **Debug:** Shows the list of debug launches and allows you to manage them using the icon bar as follows:
  -  Remove All Terminated Launches
  -  Resume
  -  Suspend
  -  Terminate
  -  Disconnect




-  Step Into, Step Over, Step Return, Drop to Frame
-  Use Step Filters
-  Remove Completed Process
- **BusinessWorks Jobs:** shows all running jobs and allows you some basic management such as, to Clear All Jobs .
- **Servers:** shows the servers that are available. You can also define a new server using the New Server Wizard, which allows you to define a new server as well as to download additional server adapters.
- **Variables:** shows the variables associated with the process being debugged. The main management tasks associated with the variables are:
  -  Show Type Names
  -  Show Logical Structure
- **Breakpoints:** shows the breakpoints used for debugging. The main management tasks associated with the breakpoints are:
  -  Show Breakpoints Supported by Selected Target
  -  Go to File for Breakpoint
  -  Skip All Breakpoints
  -  Link with Debug View
  -  Add Java Exception Breakpoint
- **Job Data:** shows available information about the running process instances.
- **Process Launcher:** shows available sub-processes that can be launched. Inputs to the process instance can be provided in the process launcher.
- **Properties:** shows available information about the properties in the process being debugged.
- **Tasks:** shows all debugging tasks listed by their resource, path, location, and type.
- **Console:** gives the output of the debugging task.

## Configuring the Debugger

You can use Debug configuration to create, manage, and run configurations in TIBCO Business Studio™ for BusinessWorks™.

There are multiple ways to access Debug Configurations window:

- From the menu **Run > Debug Configurations**.
- From the **Module Descriptors > Overview** Testing area, click  **Launch BusinessWorks Debugger**.

Using the Debug Configurations dialog box, you can select the following:

- Applications to debug.
- Advanced configurations such as logging configuration and engine debug port.

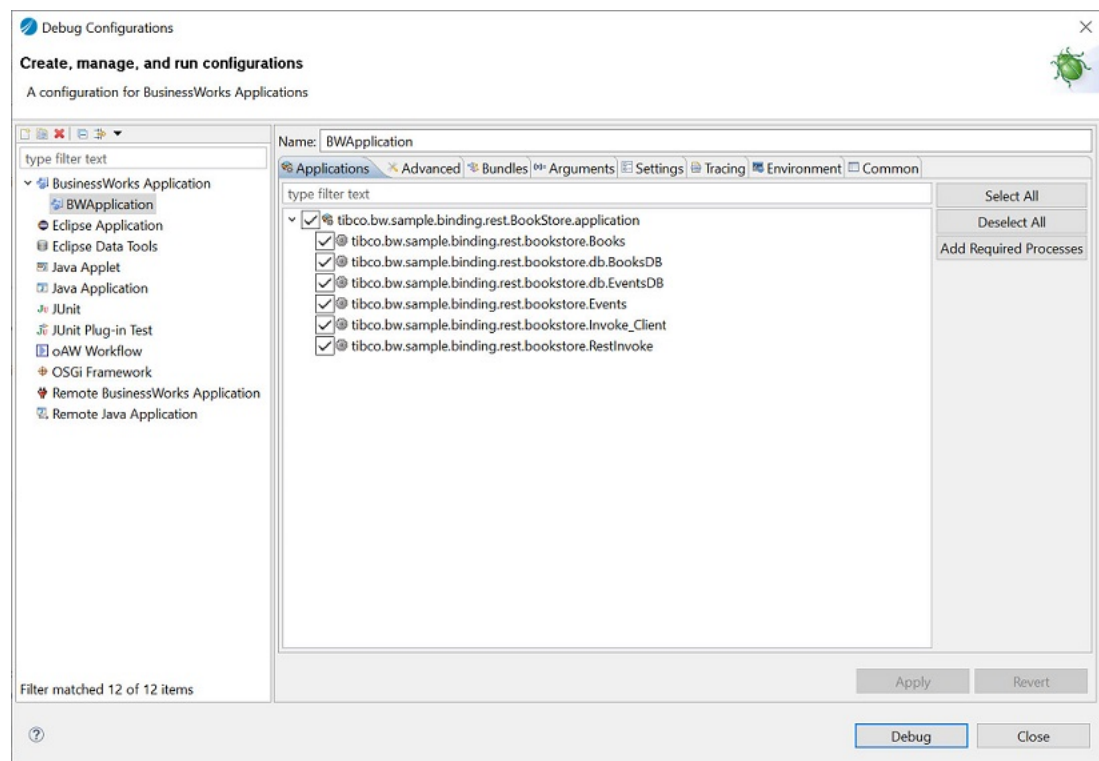
- Arguments: program arguments such as the target operating system, target architecture, target web services, working directory, and so on, and VM arguments such as *TIBCO\_HOME*, *port number*, or any engine properties that need to be set when running the application.
- Settings that define the Java Runtime Environment such as Java executable and runtime JRE, configuration area, and so on.
- Tracing criteria for the available OSGi bundles. By default, tracing is disabled. When enabled, you can choose among the available OSGi bundles, and then select the desired tracing criteria for each of them.
- Environment variables such as *PATH*, *LD\_LIBRARY\_PATH*, and so on.
- Common settings where you can save the configuration either as a local or a shared file and display them in the favorites menu (Debug and/or Run), define encoding for the files, and so on.

After selecting the options, click **Apply** to apply the changes or **Debug** to launch the debugger with the selected debug configuration.

For components or main processes that have dependent subprocesses, the debug configuration operation allows you to add required processes.

To add required processes:

1. Select **Run > Debug Configurations... > BusinessWorks Application > BWApplication** from the main menu.



2. On the **Applications** tab, select the component application in the applications tree.
3. Click **Add Required Processes**.




If one of the required processes is missing, the following error message is displayed: `BX-600018: Process [test.SubProcess] not found.`

## Testing an Application in TIBCO Business Studio™ for BusinessWorks™

Using TIBCO Business Studio for BusinessWorks, you can test your applications during design phase using the debugger.

The debugger provides the runtime environment to test your application in TIBCO Business Studio for BusinessWorks by starting the ActiveMatrix BusinessWorks engine, domain (BWEclipseDomain), AppSpace (BWEclipseAppSpace), and AppNode (BWEclipseAppNode) from within TIBCO Business Studio for BusinessWorks. When you run an application using the debugger, the Console view displays all messages when executing the application.

### Procedure

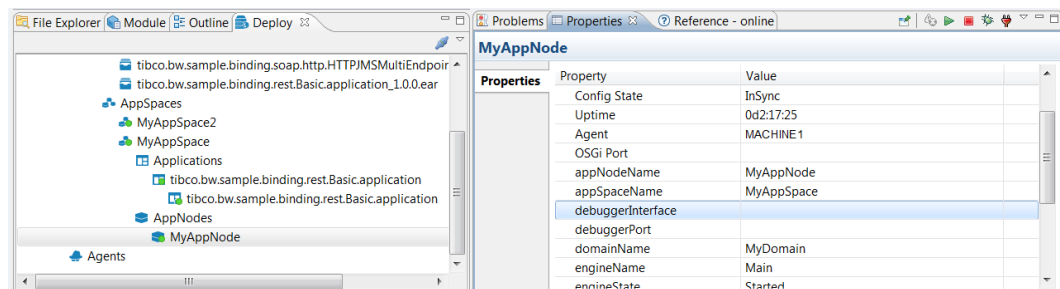
1. Open the application module in TIBCO Business Studio for BusinessWorks and select the component process in the Project Explorer. The selected process opens in the Process Editor.
2. From the menu, click **Run > Debug Configurations**.
3. In the Debug Configurations window, expand the tree under **BusinessWorks Application** and select **BWApplication**.
4. Click the **Applications** tab. If multiple applications are selected, click **Deselect All**. Then select the check box next to the application name you want to run. If needed, specify additional information such as engine properties in the debug configuration. For more information, see [Configuring the Debugger](#).
5. Click **Debug** to run the application in Debug mode. The engine and the runtime entities such as domain (BWEclipseDomain), AppSpace (BWEclipseAppSpace), and AppNode (BWEclipseAppNode) are started and the selected application deploys. The Console view displays a log of the execution.
6. After completing the execution, click the **Terminate**  icon to stop the process.


## Remote Debugging

You can debug an application running on a remote AppNode through TIBCO Business Studio™ for BusinessWorks™.

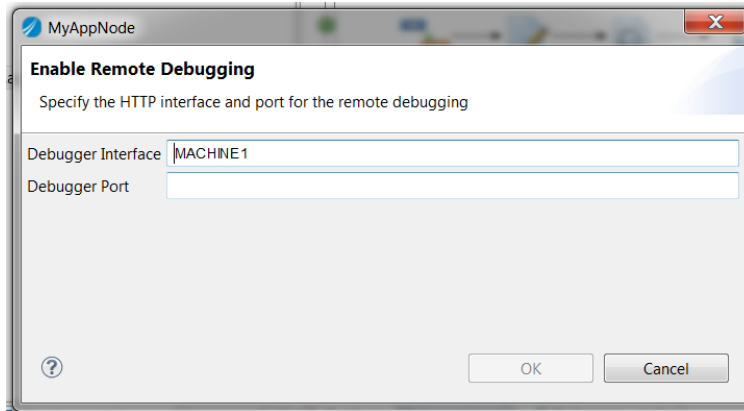
### Procedure

1. Enable the AppNode for debugging. (The AppNode must be running.)
  - a) Open the network in the **Deploy** pane and choose the AppNode. The AppNode properties are displayed in the **Properties** pane.



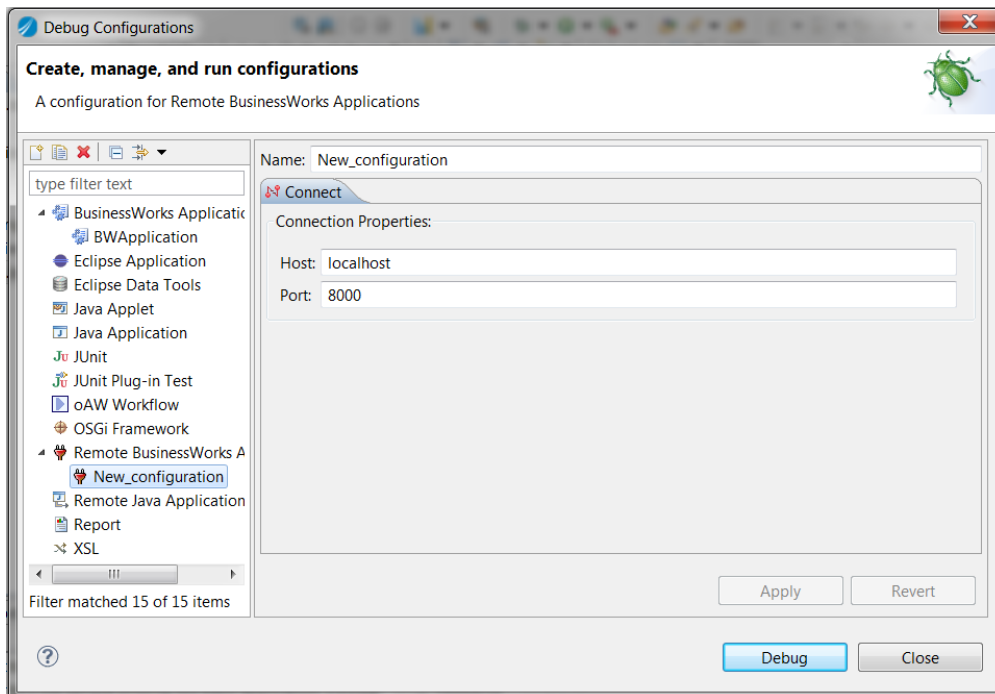
- b) Click the **Enable Debug**  icon in the Properties pane to enable remote debugging.
- c) Enter the interface and port for remote debugging on the selected AppNode in the Enable Remote Debugging dialog box.

- **Debugger Interface:** The interface for the debugger. This value is auto-generated.
- **Debugger Port:** The port to use for remote debugging. This is the same as the port number you entered for the remote debug configuration. This port cannot be in use. If the port is in use, the following message is displayed at the top of the dialog box: Internal server error

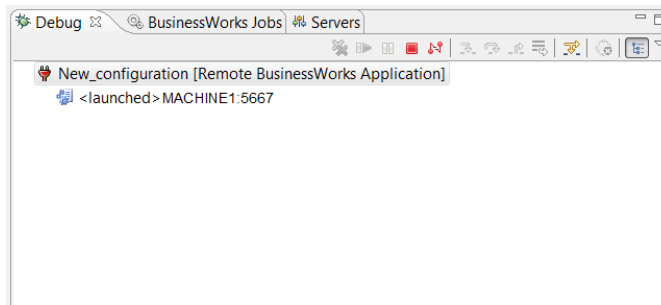


The remote debugger can also be launched with the **Debug** icon in the Properties view. The connection parameters on the Enable Remote Debugging dialog box will automatically be entered based on the AppNode configuration.

2. In TIBCO Business Studio for BusinessWorks, create a Remote Debug launch configuration.
  - a) Choose **Run > Debug Configurations**.
  - b) In the Debug Configuration dialog box, choose **Remote BusinessWorks Application > New\_configuration**. Enter the following information:
    - **Name:** The name of the configuration.
    - **Host:** The name of the host. This is the agent name. To find the agent name, right-click the network name in the Deployment Servers window and choose **Edit**. The agent name is displayed in the **Agent HTTP Interface** field of the Add Network dialog box.
    - **Port:** The remote debug port. The port cannot be in use.



3. Deploy the application you want to debug to a network. For more information, see [Deploying an Application](#).
4. Launch the application using the Remote Debug launch configuration. The application is launched in the debugger. Confirmation is displayed in the Debug window.



# Unit Testing

---

Unit testing in TIBCO ActiveMatrix BusinessWorks™ consists of verifying whether individual activities in a process are behaving as expected. While you can run unit tests on processes any time during the development cycle, testing processes before you push the application to the production environment might help you to identify issues earlier and faster.

## Running Test Assertions

Unit tests focus on testing small units of work, which in TIBCO ActiveMatrix BusinessWorks™ maps to individual sub-processes. Ideally this is done in a standalone manner, with no touchpoints or dependencies on other components or interfaces. This is distinct from interface or system testing which would test the service or operation as a whole. Interface tests are run using other tools such as SOAP UI.

By design, ActiveMatrix BusinessWorks™ test assertions don't test the top-level process - they only test sub-processes. For example, if you had an EMS receiver as the start of a top-level process this would require an interface to EMS to run the tests which creates a dependency.

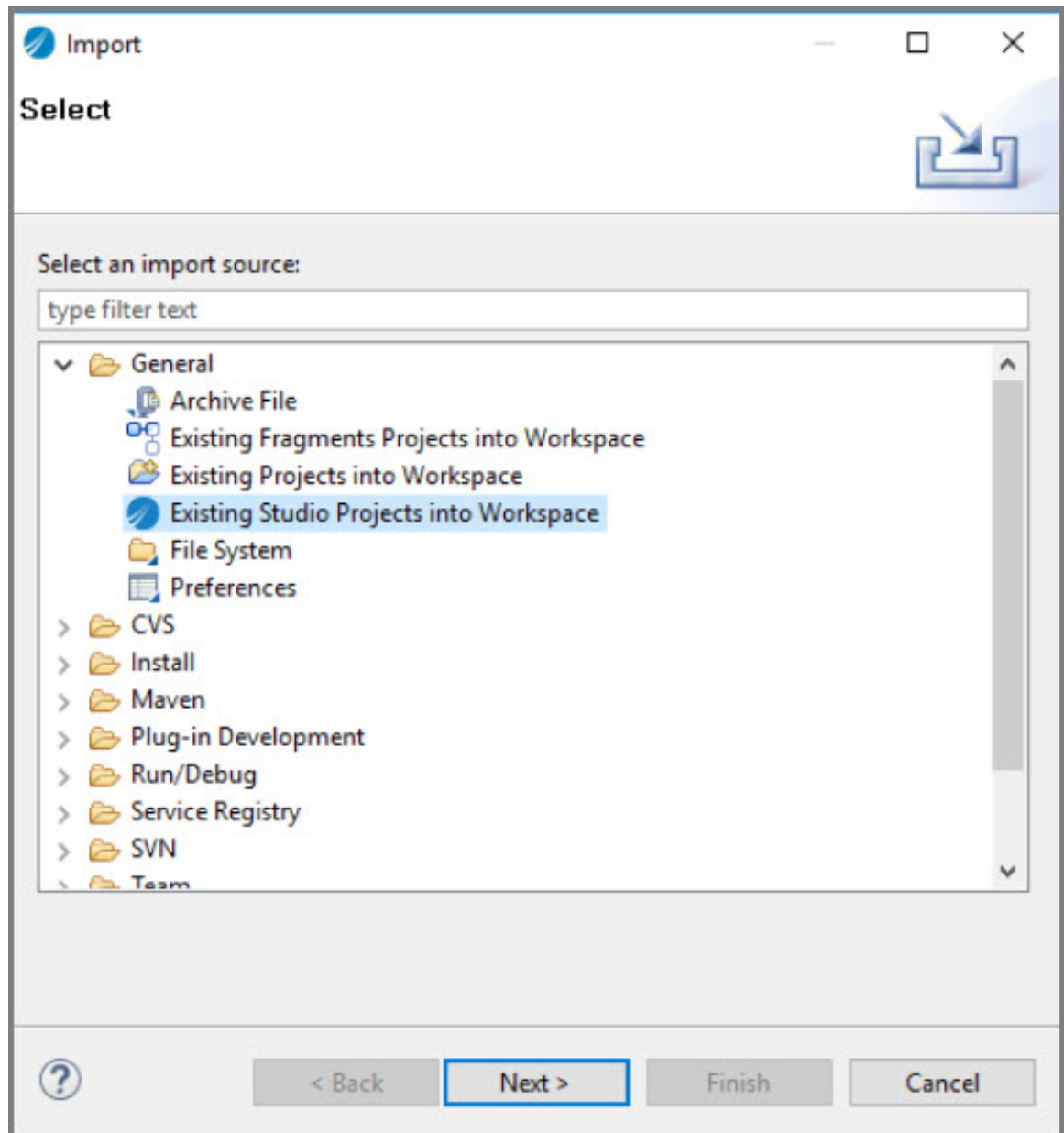
While test assertions have been available in previous versions of ActiveMatrix BusinessWorks, these instructions are specific to ActiveMatrix BusinessWorks 6.5.0 which introduced the ability to run unit tests in isolation, i.e. just test individual sub-processes using Maven without having to start the whole service. ActiveMatrix BusinessWorks 6.5.0 also introduces new capability around unit test reporting and test coverage reporting.

## Using Demo Projects

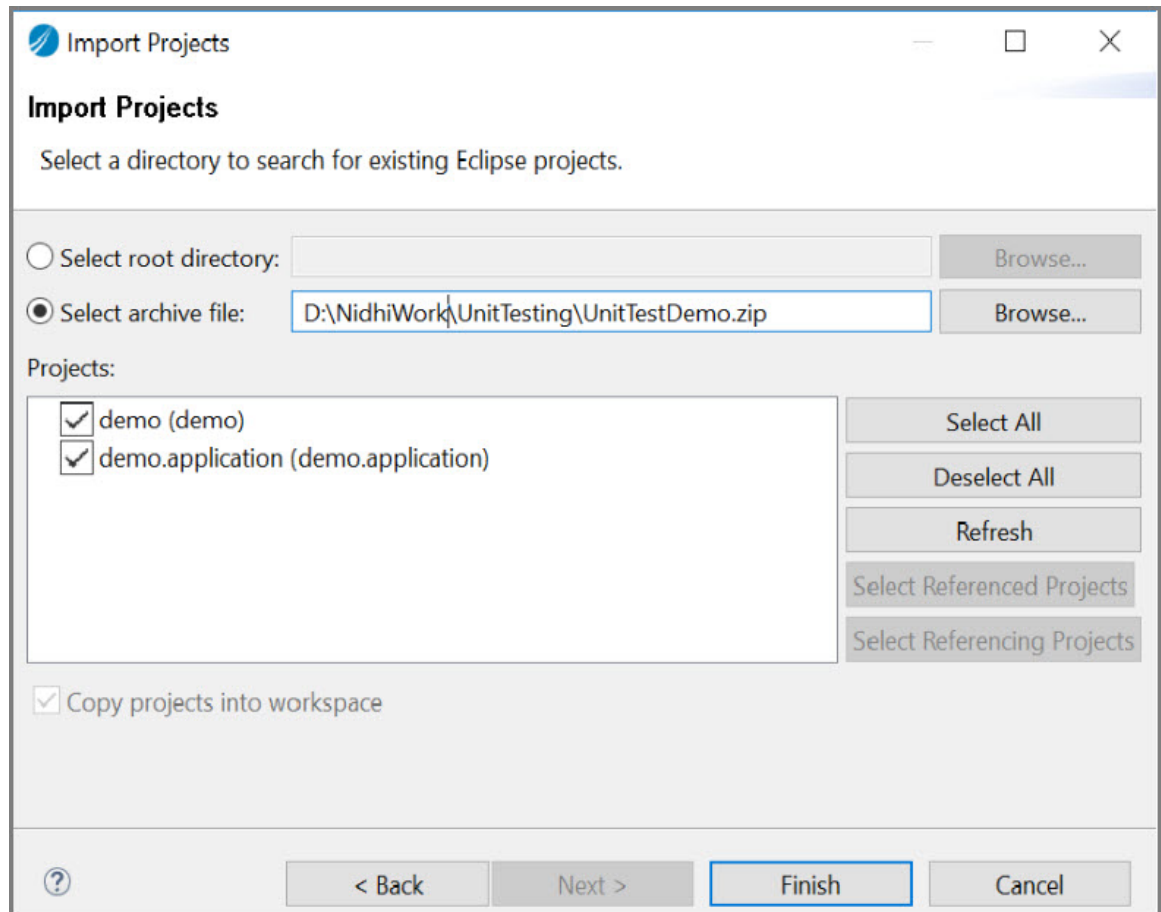
To open a demo project in TIBCO Business Studio™ for BusinessWorks™, follow these steps:

### Procedure

1. Open TIBCO Business Studio™ for BusinessWorks™ and create a new Workspace.
2. From the main menu, select **File->> Import > General > Existing Studio Projects Into Workspace** and click **Next**.



3. Select the **Select archive file:** option to import the projects in .zip file where the required project is stored.
4. Click the browse button next to the **Select archive file:** option to import the projects in a .zip file or copy paste the path of the required .zip file in the **Select archive file:** input field and click **Finish**.



- In Project explorer, select the `UnitService.bwp` process from the demo project. This is a top level process which offers two REST calls to retrieve the distance and temperature units.

## Adding Unit Test Assertions

To add unit test assertions in TIBCO Business Studio™ for BusinessWorks™, follow these steps:

### Prerequisites

- Apache Maven:  
<https://maven.apache.org/download.cgi>
- TIBCO ActiveMatrix BusinessWorks™ Maven Plugin:  
<https://github.com/TIBCOSoftware/bw6-plugin-maven/releases>
- The `UnitTestDemo.zip` must be present in an accessible location.

### Procedure

- In TIBCO Business Studio™ for BusinessWorks™, on the demo project right-click the Tests folder and select **New > Add Test File**. The **New Test File** wizard displays with the **Test File** page.
- In the **New Test File** wizard, change the file name to `GetDistanceUnit-NAM.bwt` and keep the Tests folder as default. Click **Next**.



New Test File

### Test File

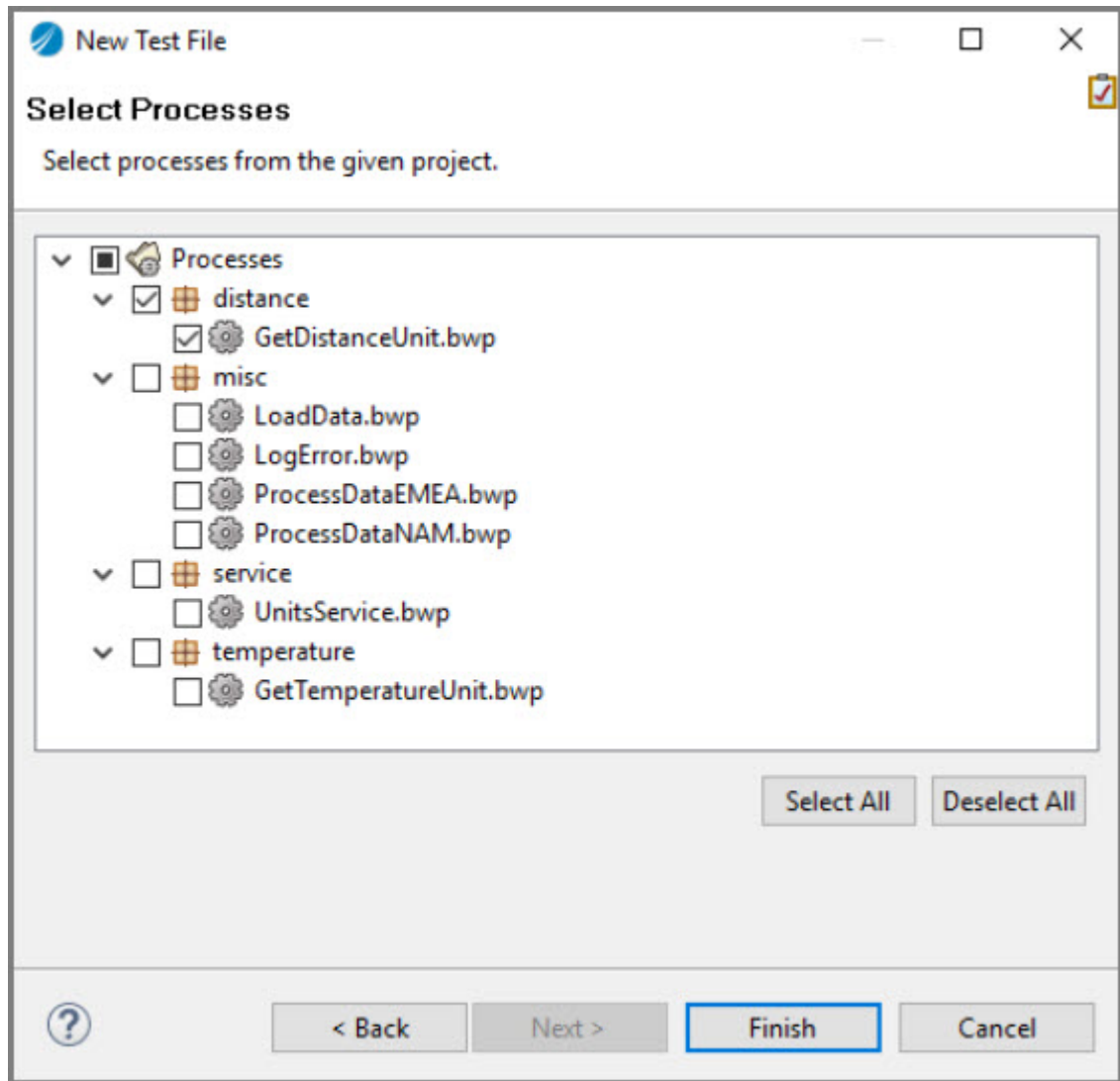
Create a new test file

Select parent Test folder and target file name:

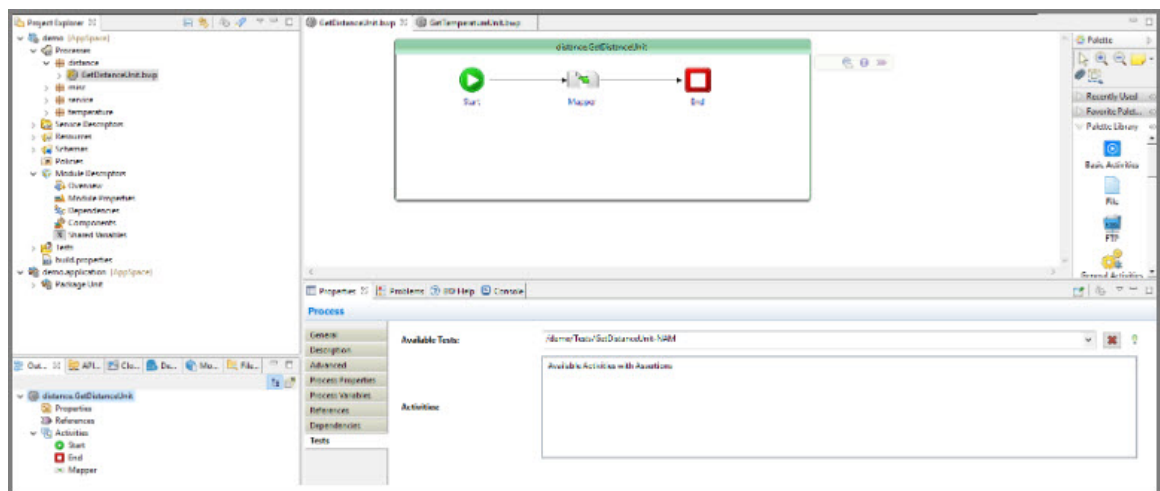
Test folder:

File:

3. Add the `GetDistanceUnit.bwt` to the process and click **Finish**.

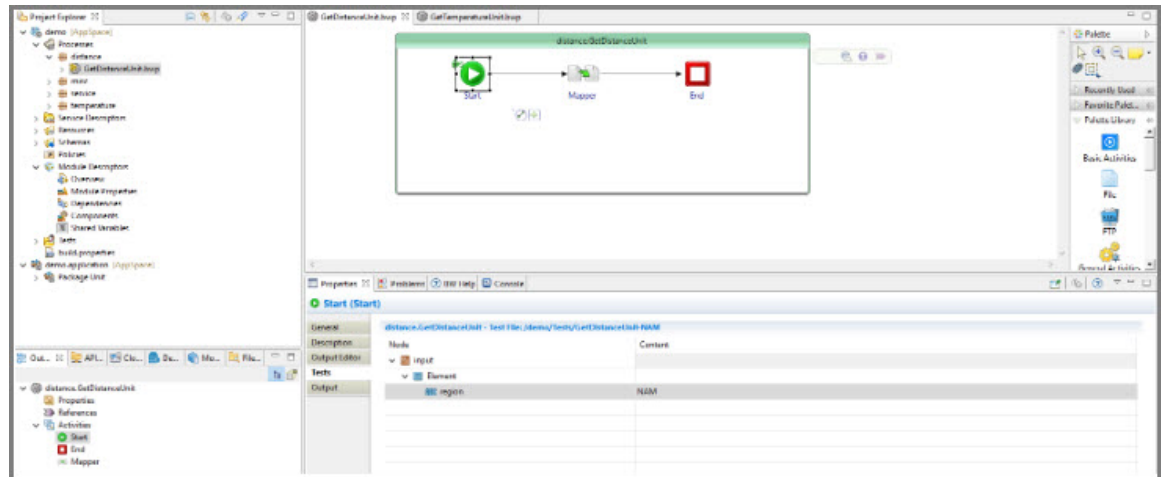


4. In the Project Explorer, open `GetDistanceUnit.bwp` and click on the `distance.GetDistanceUnit` process (green box) and select the **Properties** tab. Since this process is added to the Tests file, the **Tests** tab appears on the **Process** panel. Click on the Tests tab and the created file is selected in the **Available Tests** dropdown.



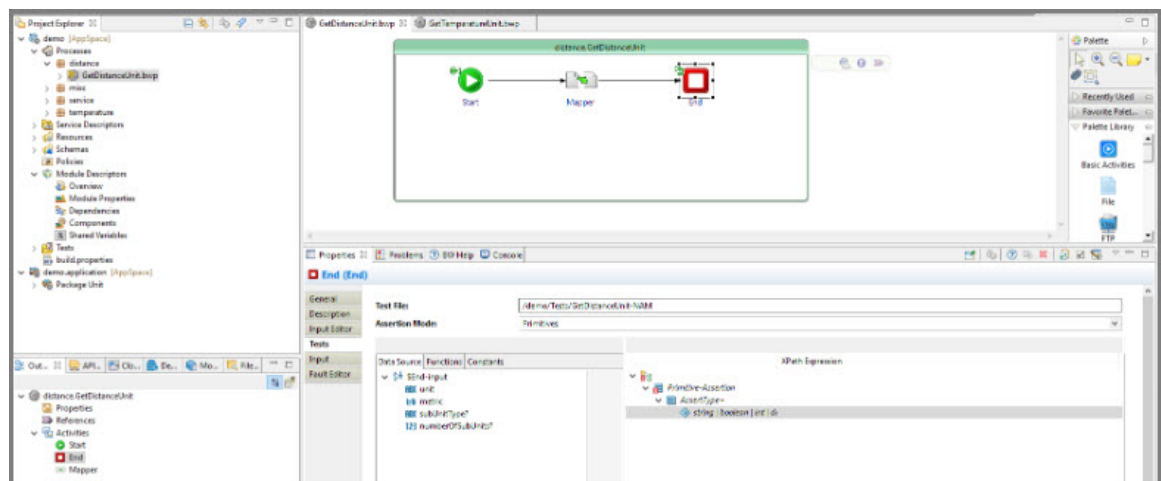
5. Right-click the **Start** activity and select **Add Test > Add Input**. Click on the **Tests** tab under **Properties** and add NAM in the **Content** column for the **region** field.

Note: NAM should not contain any double quotes ("").

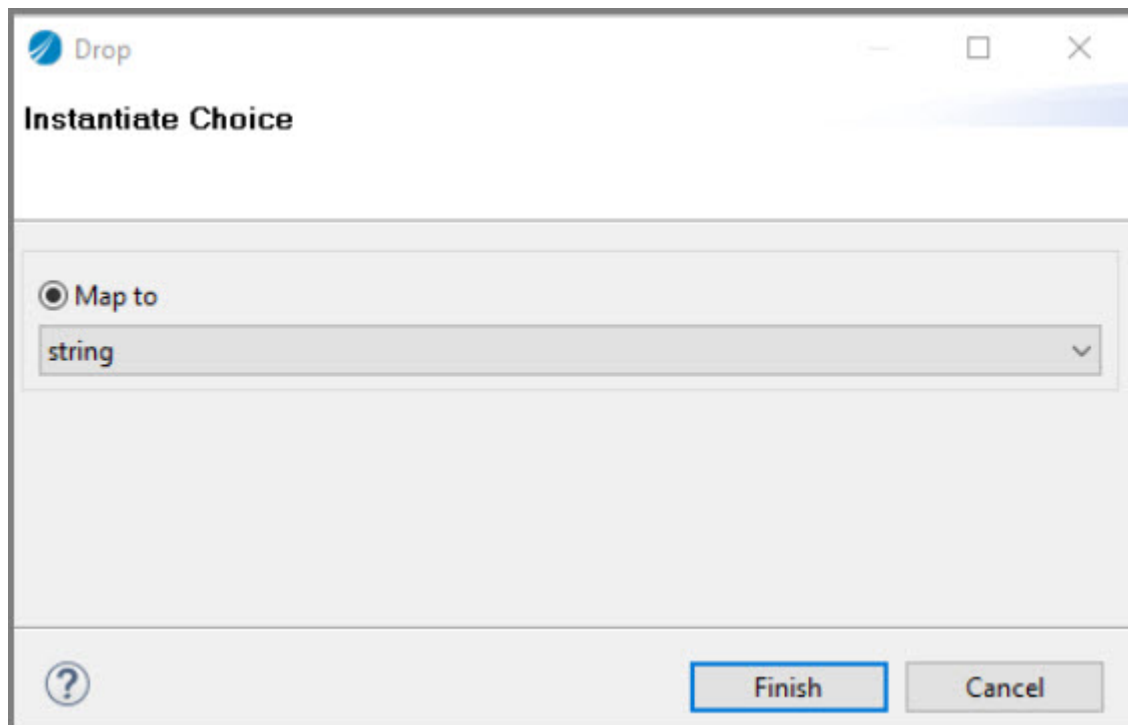


The process does not need to be saved after adding the test inputs and assertions.

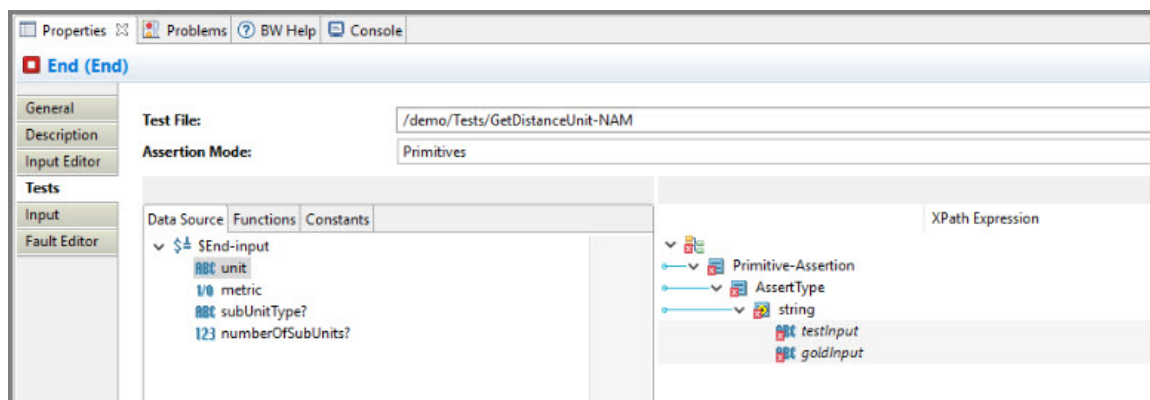
- Right-click the **End** activity and select **Add Test > Add Input**. Click on the **Tests** tab under **Properties** and expand **AssertType+** and **\$End-input** which is both the sides of the mapper.



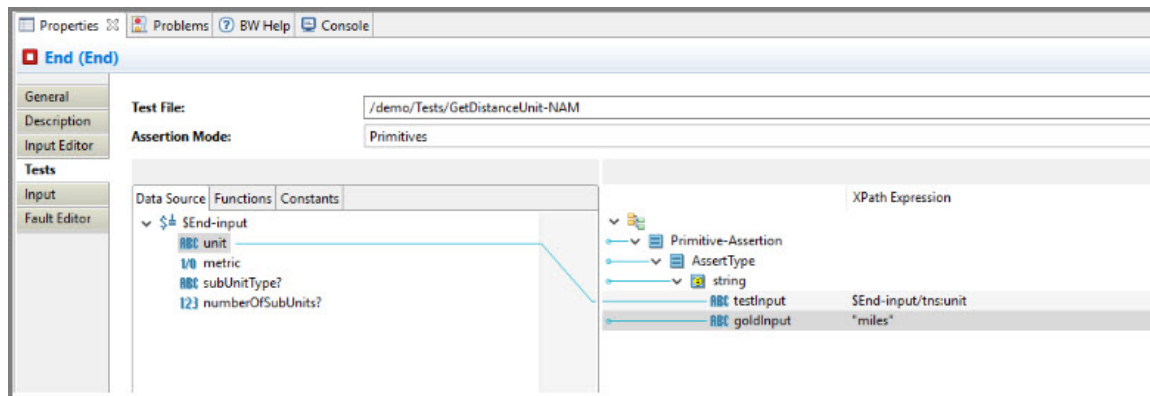
- Drag the **string | boolean | ...** element from the right-hand side to any element on the left hand side of the mapper underneath **\$End-input**. The **Drop** wizard opens to select a data type. Select the "String" data type and click **Finish**.



The **testInput** and **goldInput** fields are displayed.

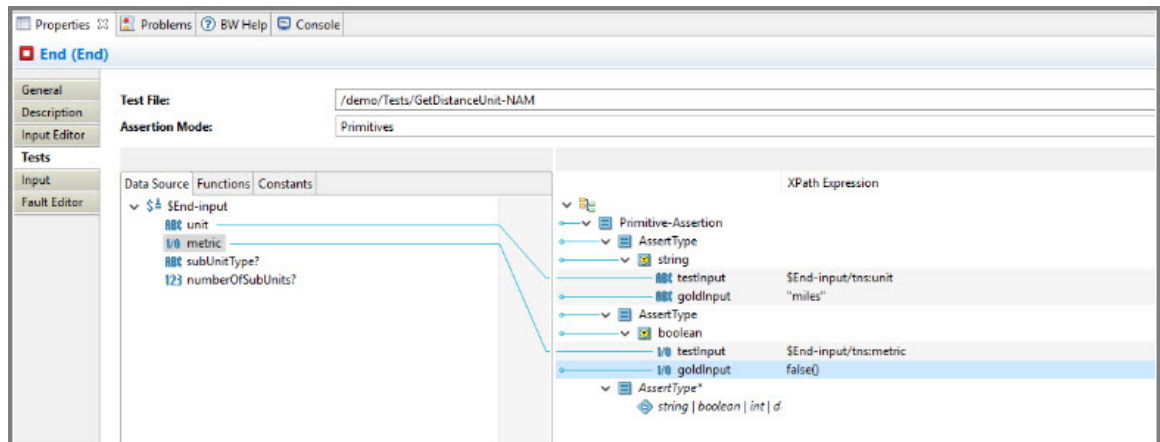


8. In the **Data Source** tab, drag "unit" to the **testInput** field. This is the value that you are evaluating in the assertion. Add miles as an input to the **goldInput** field.

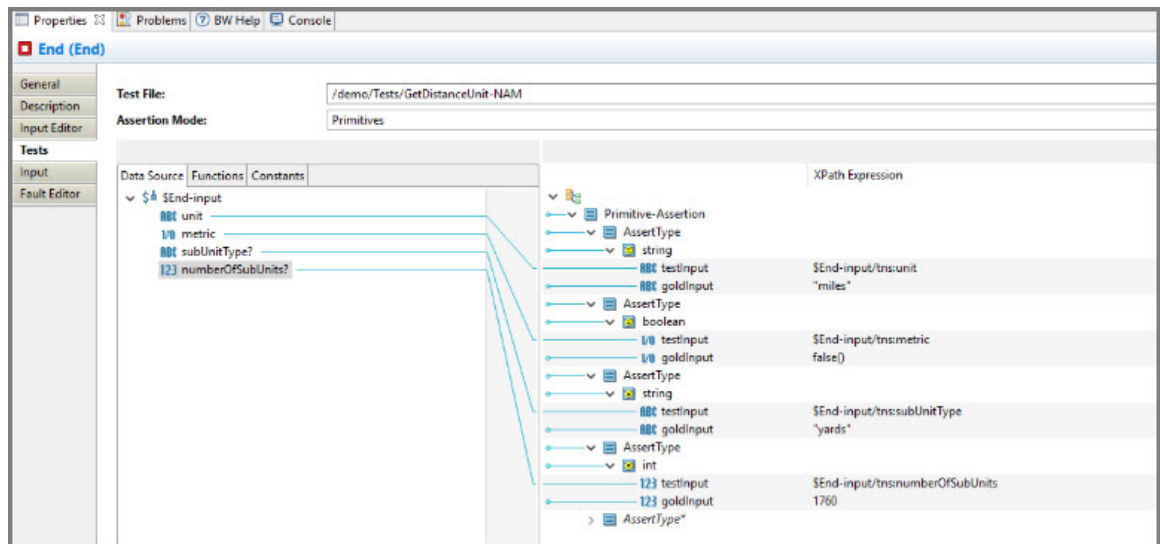


9. Right-click the **AssertType** and choose **Duplicate**. Right-click on **Primitive-Assertion** and choose **Expand All**. Under the second **AssertType** element right-click the **AssertType** and choose **Remove Mapping**. Drag the **string | boolean...** element from the right-hand side to any element under

**\$End-input** on the left and choose "boolean" data type. Drag the "metric" element from the left onto the **testInput** field under Boolean and enter `false()` in the **goldInput** field.



10. In a similar way as above, complete the mappings so that you also assert "subUnitType" and "numberOfSubUnits"



11. To add a new test file, right-click the Tests folder and select **New > Add Test File**. In the **File** field add the name of the file as `GetDistanceUnit-EMEA.bwt` and click **Next**.

New Test File

**Test File**

Create a new test file

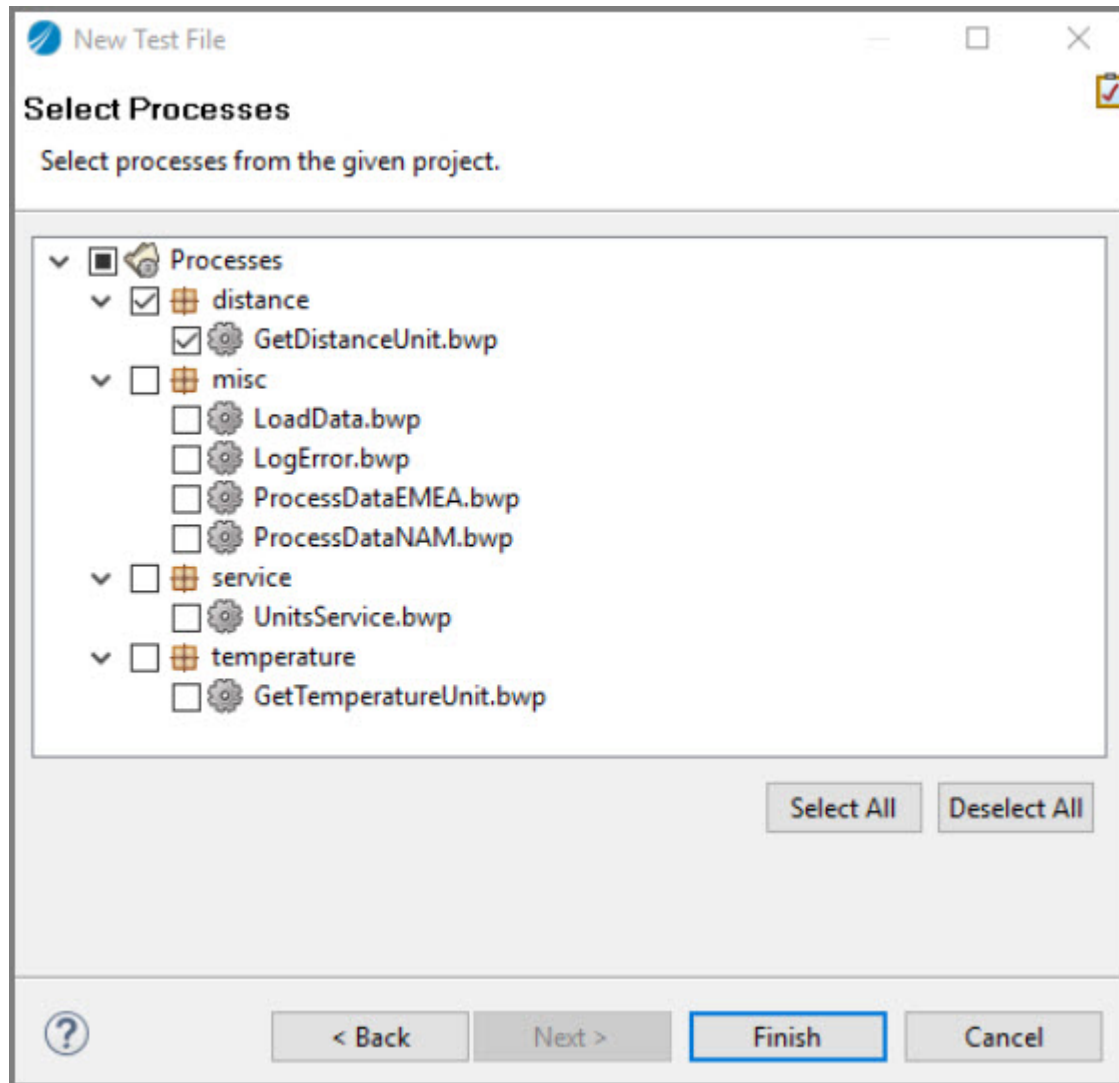
Select parent Test folder and target file name:

Test folder:  Browse..

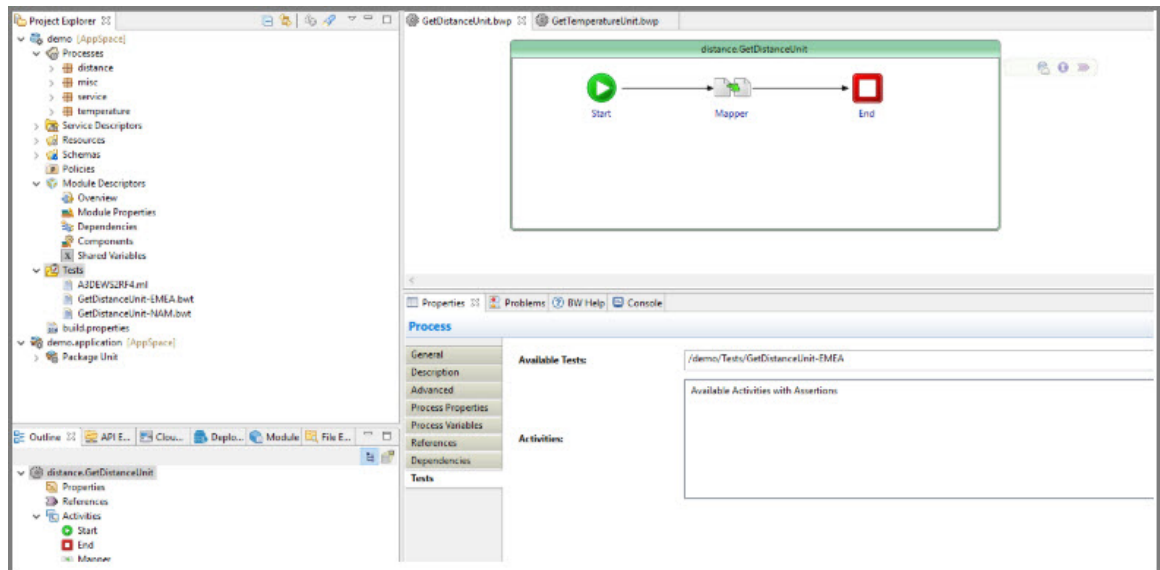
File:

? < Back Next > Finish Cancel

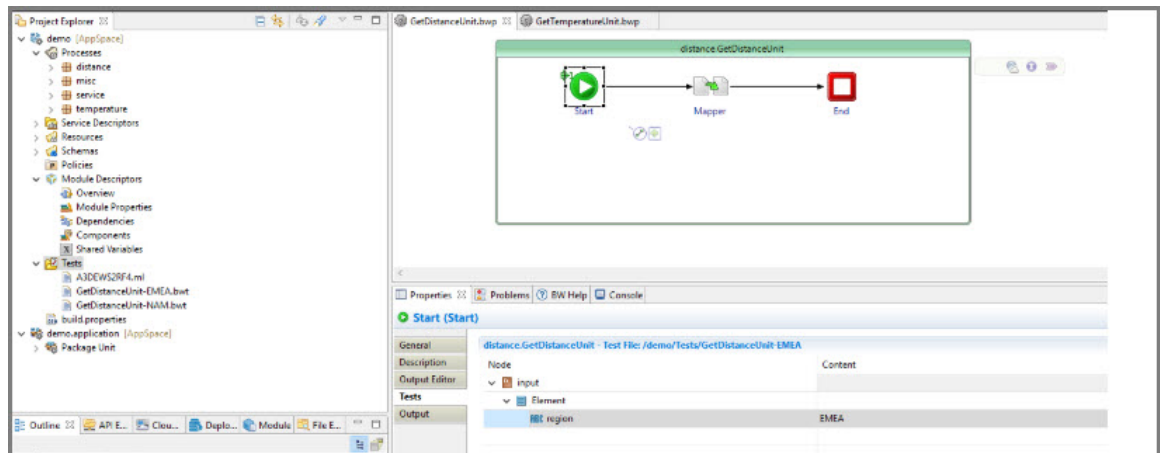
12. Select `GetDistanceUnit.bwt` and click **Finish**.



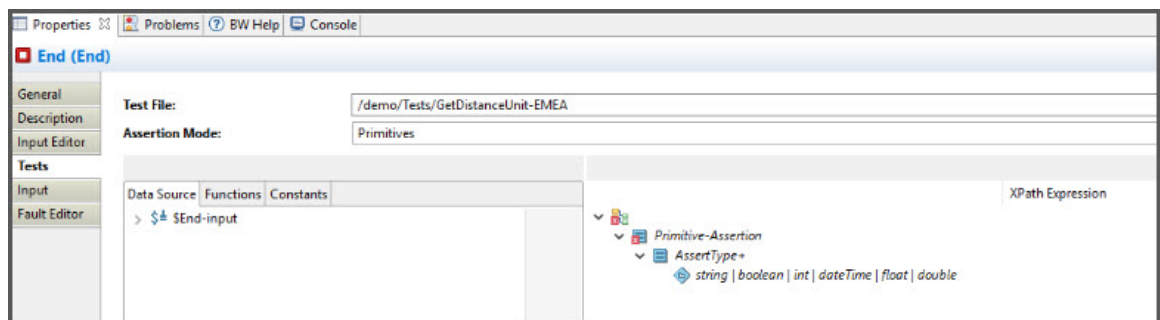
- In the Project Explorer, open `GetDistanceUnit.bwp` and click on the `distance.GetDistanceUnit` process (green box) and select the **Properties** tab. Since this process is added to the Tests file, the **Tests** tab appears on the **Process** panel. Click on the Tests tab and the `demo/Tests/GetDistanceUnit-EMEA` test file is selected in the **Available Tests** dropdown. If not, select it manually.



14. Right-click the **Start** activity and select **Add Test > Add Input**. Click on the **Tests** tab under **Properties** and add EMEA in the **Content** column for the **region** field.



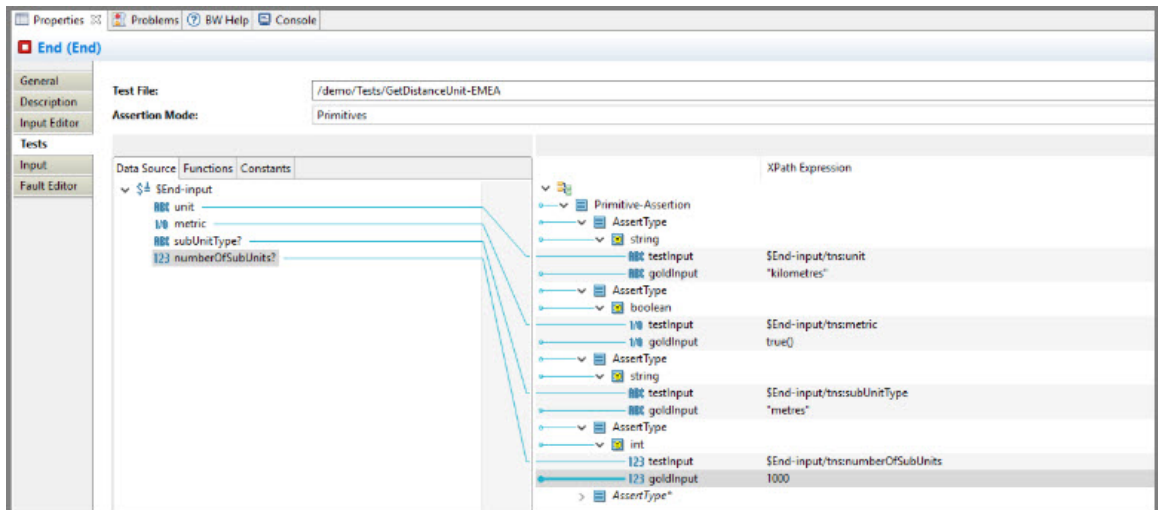
15. Right-click the **End** activity and select **Add Test > Add Input**. Click on the **Tests** tab under **Properties** and expand **AssertType+** and **\$End-input** which is both the sides of the mapper.



16. Repeat steps 7, 8, 9, and 10 to set the assertions for GetDistanceUnit-EMEA with "unit", "metric", "subUnitType", and "numberOf SubUnits".

The output should look as follows:





To run Unit Tests in TIBCO Business Studio for BusinessWorks, see [Running Unit Tests in Studio](#).

## Running Maven from Command Line

To run Maven plug-in from command line, follow these steps

### Procedure

1. Open your command prompt and navigate to the location where your demo project is present.
2. Run the command `clean initialize site package` on your command prompt terminal.

This produces the same result as running Debug within TIBCO Business Studio™ for BusinessWorks™.

```
D:\>cd D:\tibco-workspace\runtime\bw6_runtime\BWUnitTesting5\demo.application.parent
D:\tibco-workspace\runtime\bw6_runtime\BWUnitTesting5\demo.application.parent>mvn clean initialize site package
[INFO] Scanning for projects...
[INFO] Starting Maven Build for BW6 Project.....
[INFO] Checking for In-Project JAR dependencies if any and Pushing them to Local Maven Repository
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] demo.application.parent [pom]
[INFO] demo [bwmodule]
[INFO] demo.application [bwear]
[INFO]
[INFO] -----< com.tibco.bw:demo.application.parent >-----
[INFO] Building demo.application.parent 1.0.0-SNAPSHOT [1/3]
[INFO] -----[pom]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ demo.application.parent ---
[INFO] Deleting D:\tibco-workspace\runtime\bw6_runtime\BWUnitTesting5\demo.application.parent\target
[INFO]
[INFO] --- maven-site-plugin:3.7.1:site (default-site) @ demo.application.parent ---
[WARNING] Input file encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[WARNING] Report plugin org.apache.maven.plugins:maven-project-info-reports-plugin has an empty version
```



The screenshot shows a web browser displaying a 'demo.application' BW Coverage Report. The report is titled 'BW Coverage Report' and includes a 'Summary' section with the following data:

| Module %     | Process %      | Activity %      | Transition %   |
|--------------|----------------|-----------------|----------------|
| 100% (2 / 2) | 25.67% (1 / 4) | 16.67% (2 / 12) | 24.29% (2 / 8) |

Below the summary is a 'Coverage BreakDown By Process' table:

| Module | Process                        | Activity %   | Transition % |
|--------|--------------------------------|--------------|--------------|
| demo   | misc.LogError                  | 0% (0 / 2)   | 0% (0 / 1)   |
| demo   | misc.ProcessDataNAM            | 0% (0 / 2)   | 0% (0 / 1)   |
| demo   | misc.ProcessDataMEA            | 0% (0 / 2)   | 0% (0 / 1)   |
| demo   | distance.GetDistanceUnit       | 100% (2 / 2) | 100% (2 / 2) |
| demo   | misc.LoadData                  | 0% (0 / 6)   | 0% (0 / 7)   |
| demo   | temperature.GetTemperatureUnit | 0% (0 / 3)   | 0% (0 / 2)   |



The top-level process is not included in this list as it cannot be unit tested. This could be used within CI tools such as Jenkins to ensure that a certain percentage of tests are covered in order to allow a build to proceed.

## Limitations for Unit Test Assertions

The following are the limitations for the Unit Test Assertions:

- The top level processes are not supported for unit tests, only the sub processes are used for unit testing.
- TIBCO Business Studio™ for BusinessWorks™ must be installed on the same server where the tests are to run.
- Mocking of outbound service calls is not currently supported. For example, Invoke REST API, DB calls, EMS queue sender.
- Unit Tests can currently only be invoked with Maven.

## Running Activity Assertions

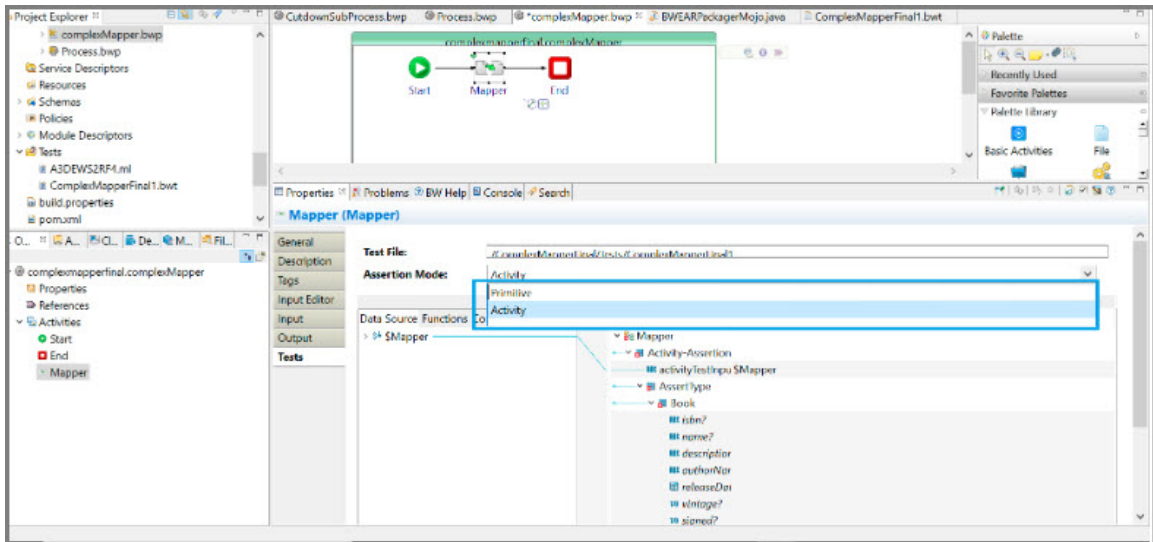
To run activity assertions in TIBCO ActiveMatrix BusinessWorks™, follow these steps:

### Procedure

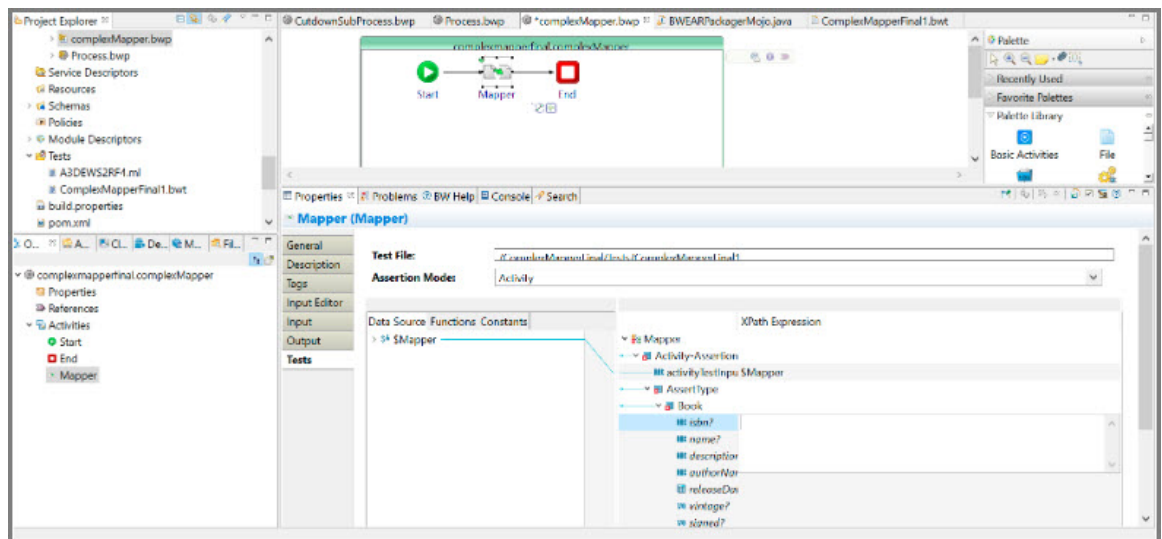
1. Right-click on the activity from the subprocess and select **Add Test > Add Assertion**. This will add the **Test** tab to the activity.
2. On the **Tests** tab, navigate to the **Assertion Mode** drop down.

The Assertion Mode drop down has two modes:

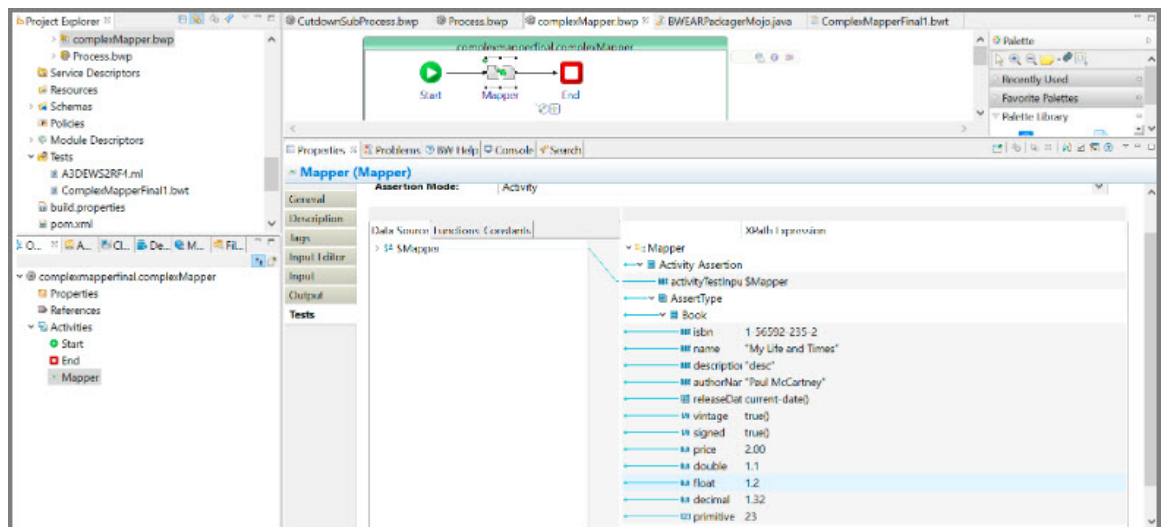
- **Primitive:** In this mode, only the primitive types of elements are tested.
- **Activity:** In this mode, the complete activity outputs are tested.



3. Select the **Activity** option from **Assertion Mode** drop down. The complete activity output schema gets loaded with editable value field Under Assert Type node. Map the activity variable from datasource section (In Image you can see it's Mapper) to **activityTestInput** field.



4. Provide the gold input to all the elements of activity schema which is under the assert node.





You don't have to save the process after adding test inputs and assertions. Also If the schema having the fields with data type decimal, double, float then add the value in the decimal format, for example 1.2 or 4.3234.

## Adding Mocking Support for Activities

This document provides steps for adding mocking support for BW activities in TIBCO ActiveMatrix BusinessWorks™ with the Maven Plug-in. User can skip execution of activity (usually activities that are based on external service) whose process is under Unit Testing. Mocking support functionality is required mainly for the ActiveMatrix BusinessWorks™ activities that are based or dependent on some external Cloud Service or Database systems which are eventually under Unit Testing. To execute Unit Testing successfully on processes that contain the ActiveMatrix BusinessWorks activities, we need to mock the ActiveMatrix BusinessWorks activities. Now a dummy output can be added to mock activities that can be used in Unit testing for successful execution. Currently, Mocking Feature doesn't support the main process - they only mock the activities from sub-processes.

### Prerequisites

- Apache Maven:  
<https://maven.apache.org/download.cgi>
- TIBCO ActiveMatrix BusinessWorks™ Plug-in for Maven Plugin 2.2.0 should be installed. Please download the plug-in from:  
<https://github.com/TIBCOSoftware/bw6-plugin-maven/releases>
- Activities to be mocked should be present in sub-process where sub-process should be under Unit testing.
- Generate valid Mock Output XML file. For more information on generating the mock output file, see [Generating Mock Output File](#).

### Adding Mock Output to an Activity

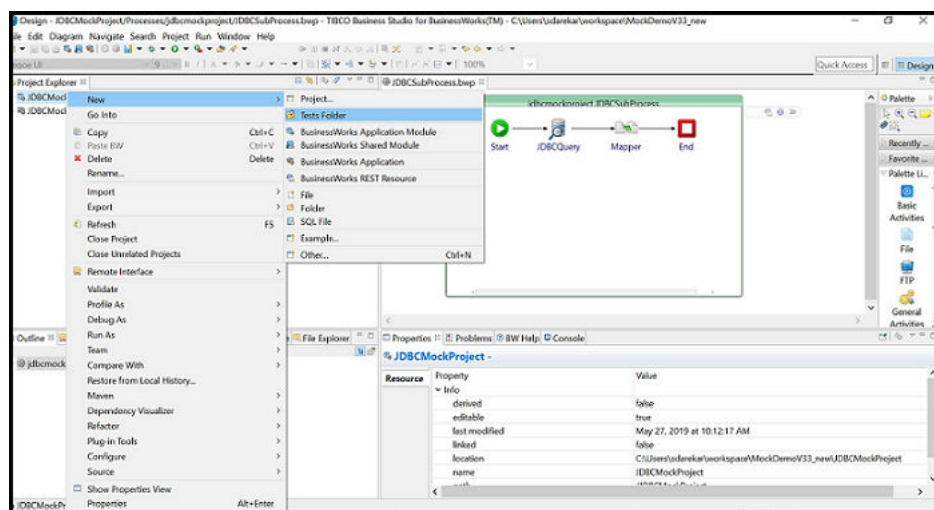
Make sure the demo project with the subprocess that has the activities to be mocked is created.

To add mock output to an activity in TIBCO ActiveMatrix BusinessWorks™, follow these steps:

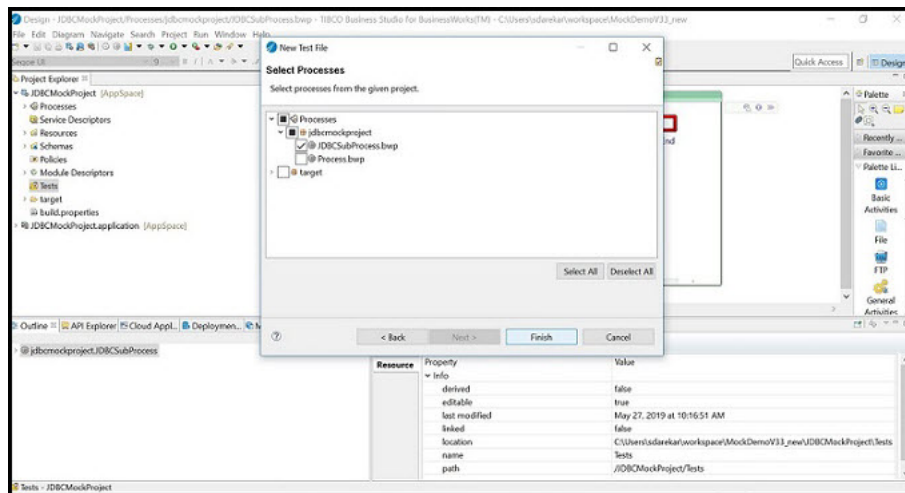
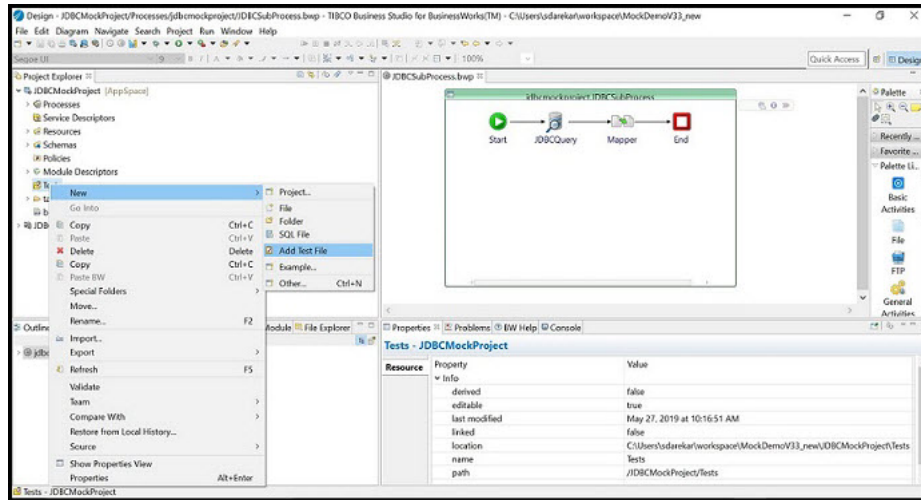
### Procedure

1. Right-click on module project and select **New > Tests Folder**.

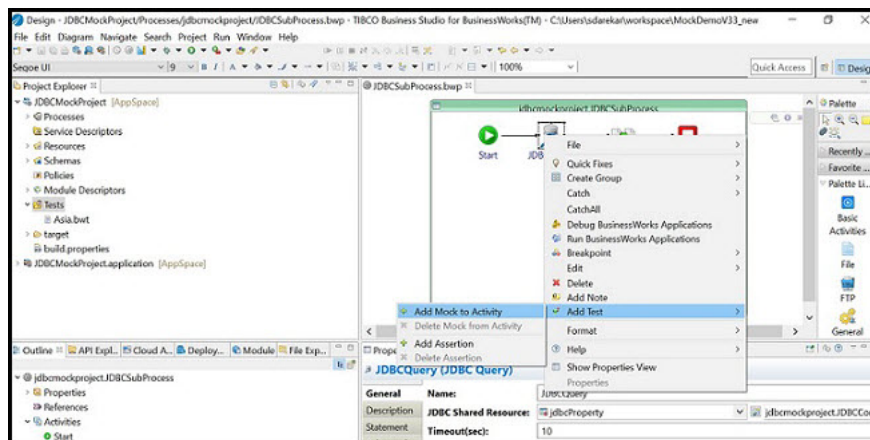
This will add the **Tests** folder in the module project.



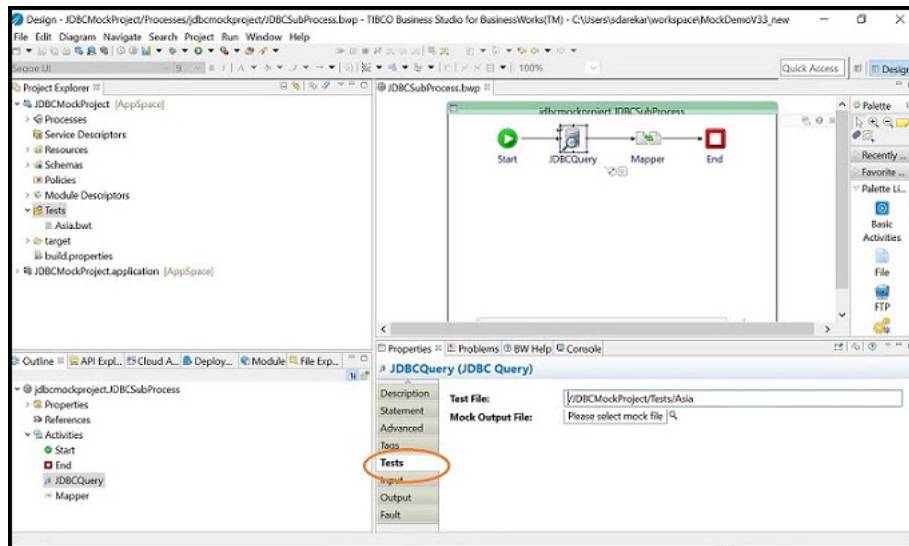
- In Project Explorer, right-click on the Tests folder and choose **New > Add Test File**. If needed, change the name of the Test File and Click **Next**. This displays the wizard with a list of subprocesses. Select the subprocess having the activities to be mocked.



- Right-click on the activity to mock and select the **Add Mock To Activity** option.



- The new **Tests** tab is added in the property section of the activity. The new **Tests** tab has file selector to select the Output File. Select the output file using File Selector.



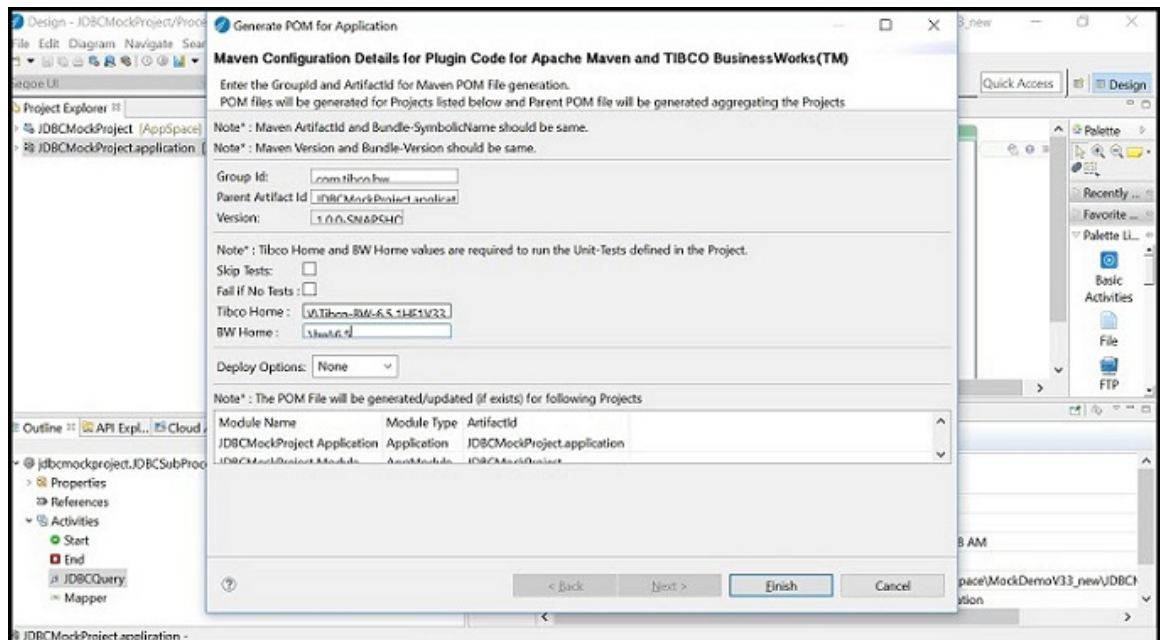
5. In the **Mock Output File** field, provide the relative mock output file.  
The mock output file should be present in the **Tests** folder of the project. The relative path will have the value like "Tests/fileName.xml". It is mandatory to provide the **Tests** folder name also in relative path.

## Running Unit Tests in TIBCO Business Studio™ for BusinessWorks™

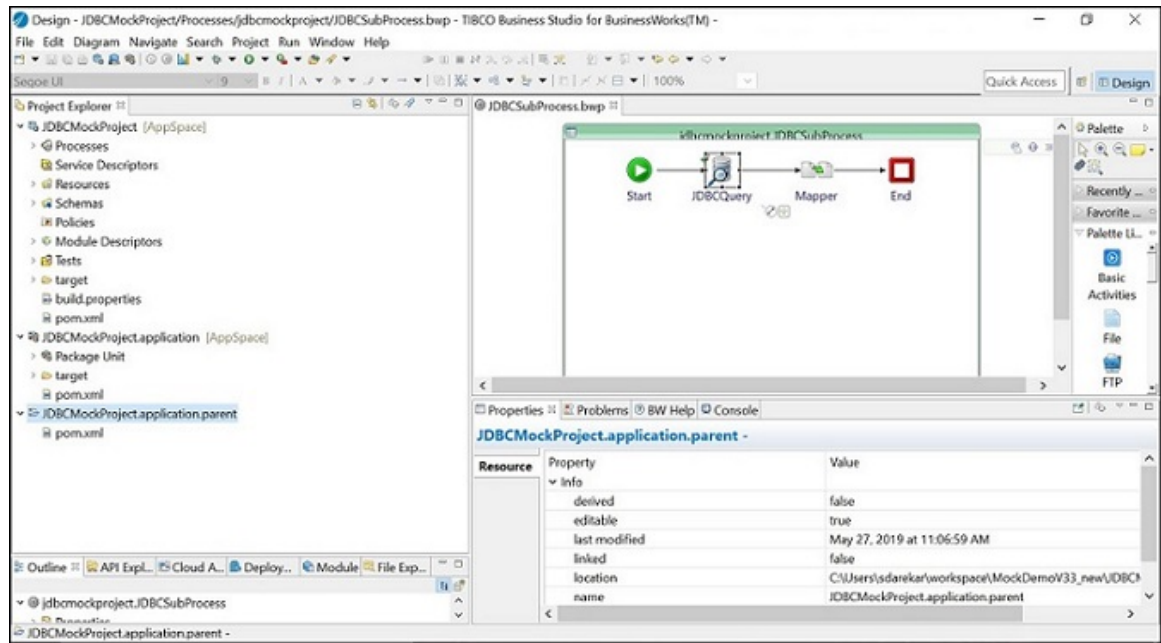
Follow the steps to run unit tests in TIBCO Business Studio for BusinessWorks

### Procedure

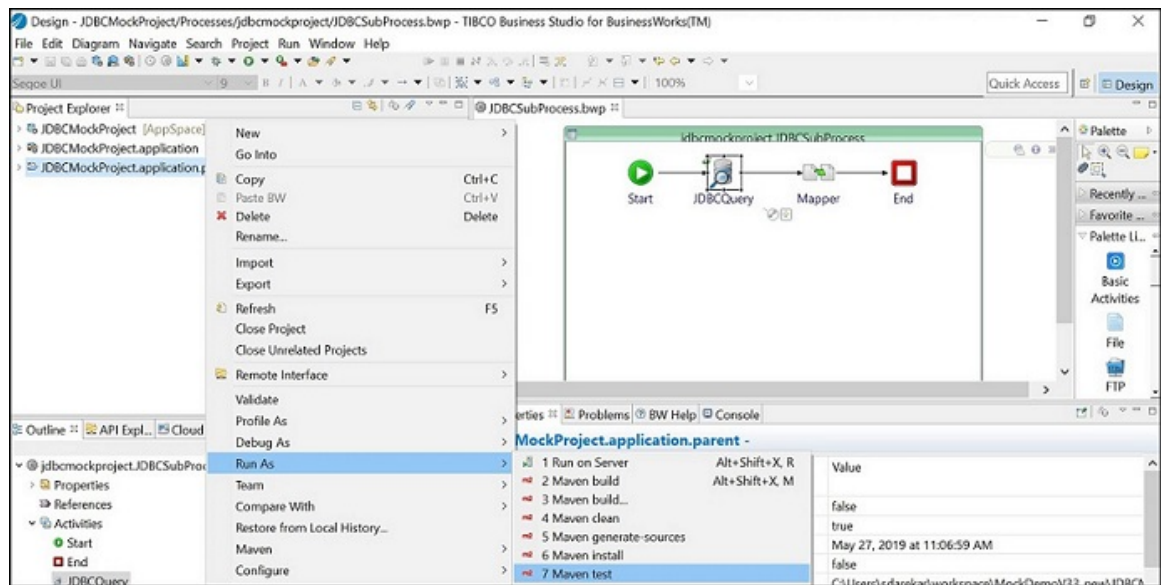
1. In TIBCO Business Studio for BusinessWorks, right-click on .application file and select the **Generate POM for Application** option. Set Tibco Home as the TIBCO Home for your BW installation with no trailing slash, for example, C:\tibco\bw651 for Windows. Set BW Home as the relative path to the version-specific BW folder under TIBCO Home (with a leading slash and no trailing slash), for example \bw\6.5 for Windows and click **Finish**.



2. This will, now, convert the existing projects to Maven type and add a new project called \*.application.parent and create pom.xml files in all projects.



3. Right-click on the parent project and run "test" goal



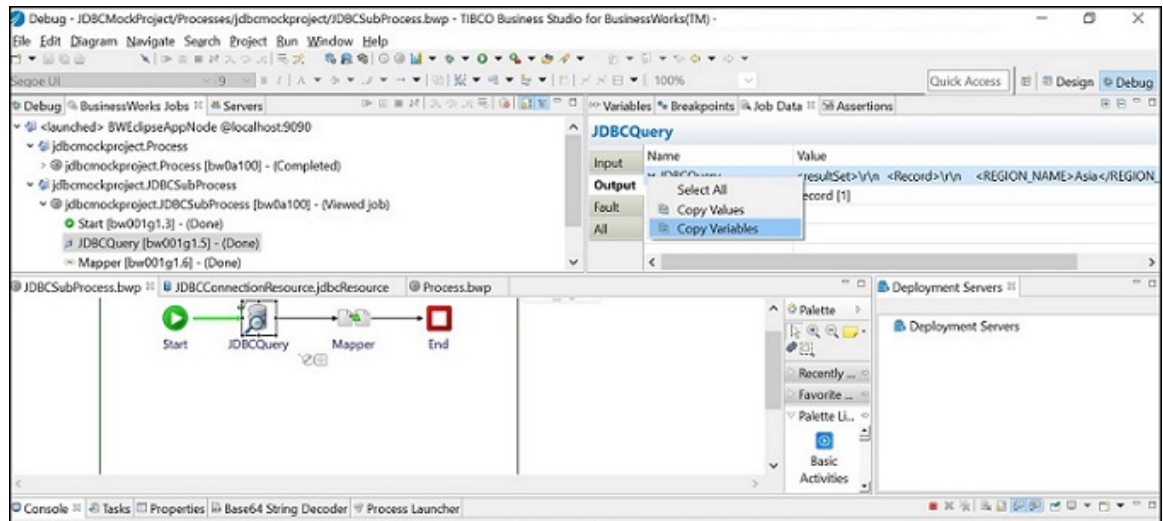
## Generating Mock Output File

To generate the mock output files in TIBCO Business Studio™ for BusinessWorks™, follow these steps:

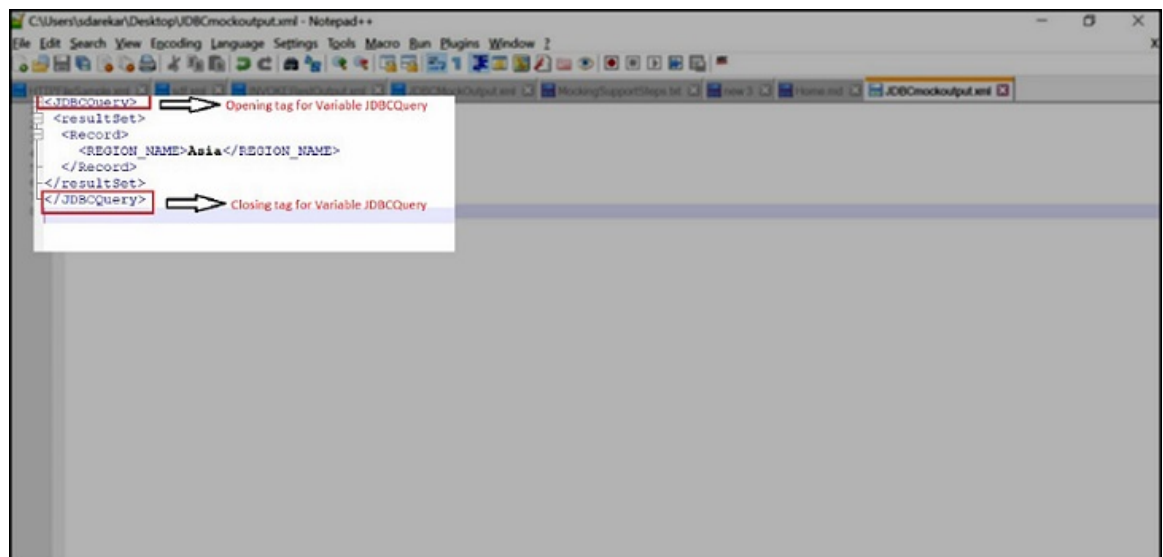
### Procedure

1. Run the application in debug mode from TIBCO Business Studio for BusinessWorks.
2. Select the **Output** tab from **Job Data** for an activity for which we need to generate the mock output file.
3. Right-click on the activity Name in **Output** tab and copy the data by selecting the **Copy Variables** option

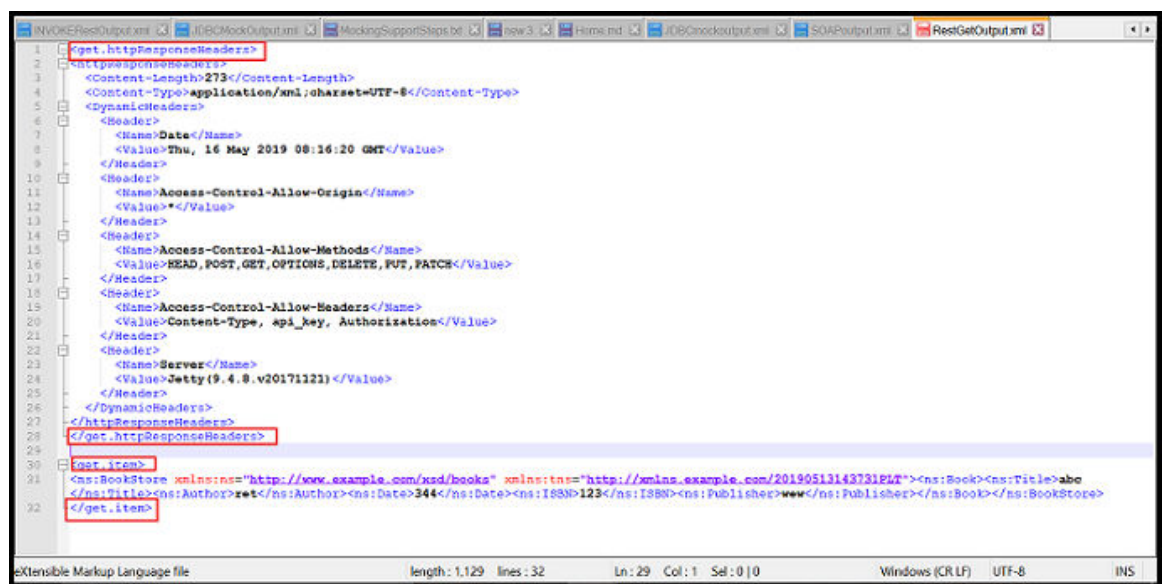




- Paste the copied data into the XML file and add opening and closing tag for variables.



- Services like REST and SOAP can have multiple variables. So in **Job Data**, the output will be shown for multiple variables. In this case, append the file for each variable data.



## Limitations for Mock Support

The following are the limitations for Mock Support in TIBCO Business Studio™ for BusinessWorks™.

- The top level processes are not supported for unit tests, only the sub processes are used for unit testing.
- The **Process Starter** and **SignIn** activities does not provide mocking support.
- TIBCO ActiveMatrix BusinessWorks™ needs to be installed on the same server where the tests are to be run
- Unit Tests can currently only be invoked with Maven

# Collaborative Application Development

Collaborative application development process helps multiple process designers to design a process simultaneously.

To keep the track of collaborative development work, configure TIBCO Business Studio™ for BusinessWorks™ with Git plug-in. For more information, see [Configuring TIBCO Business Studio™ for BusinessWorks™ with Git](#). Once you configure Git, you can commit special folders of an application module by adding .gitignore files. For more information, see [Generating gitignore files](#).

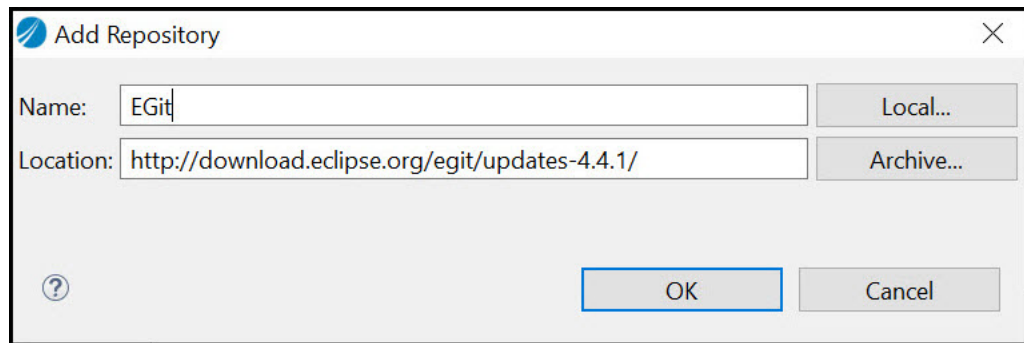
## Configuring TIBCO Business Studio™ for BusinessWorks™ with Git

### Prerequisites

Make sure that TIBCO Business Studio for BusinessWorks is open.

### Procedure

1. From the menu, select **Help > Install New Software...** to open Eclipse Update Manager.
2. In the Install dialog box, click **Add**.  
The Add Repository dialog box opens.
3. Add name and location as `http://download.eclipse.org/egit/updates-4.4.1` and click **Ok**.



TIBCO Business Studio for BusinessWorks is compatible with egit version 4.4.1 only.

4. From the list of available components, select the components you want to install and click **Next**.
5. In the Review Licenses dialog box, review the licenses, and click **I accept the terms of the license agreement**
6. Click **Finish** to start the installation of the plug-in.

After installing the software, restart TIBCO Business Studio for BusinessWorks. This restart is necessary for the software to install completely.

## Generating gitignore Files

By default, Git does not allow you to commit empty folders. To keep track of empty folders in the repository, Git allows you to add .gitignore file in such folders and then commit into the repository. In TIBCO Business Studio™ for BusinessWorks™ when you create an application module, it contains Service Descriptors, Resources, Schemas, and Policies as empty folders. Once you configure EGit plug-in with TIBCO Business Studio for BusinessWorks, you can generate .gitignore file.

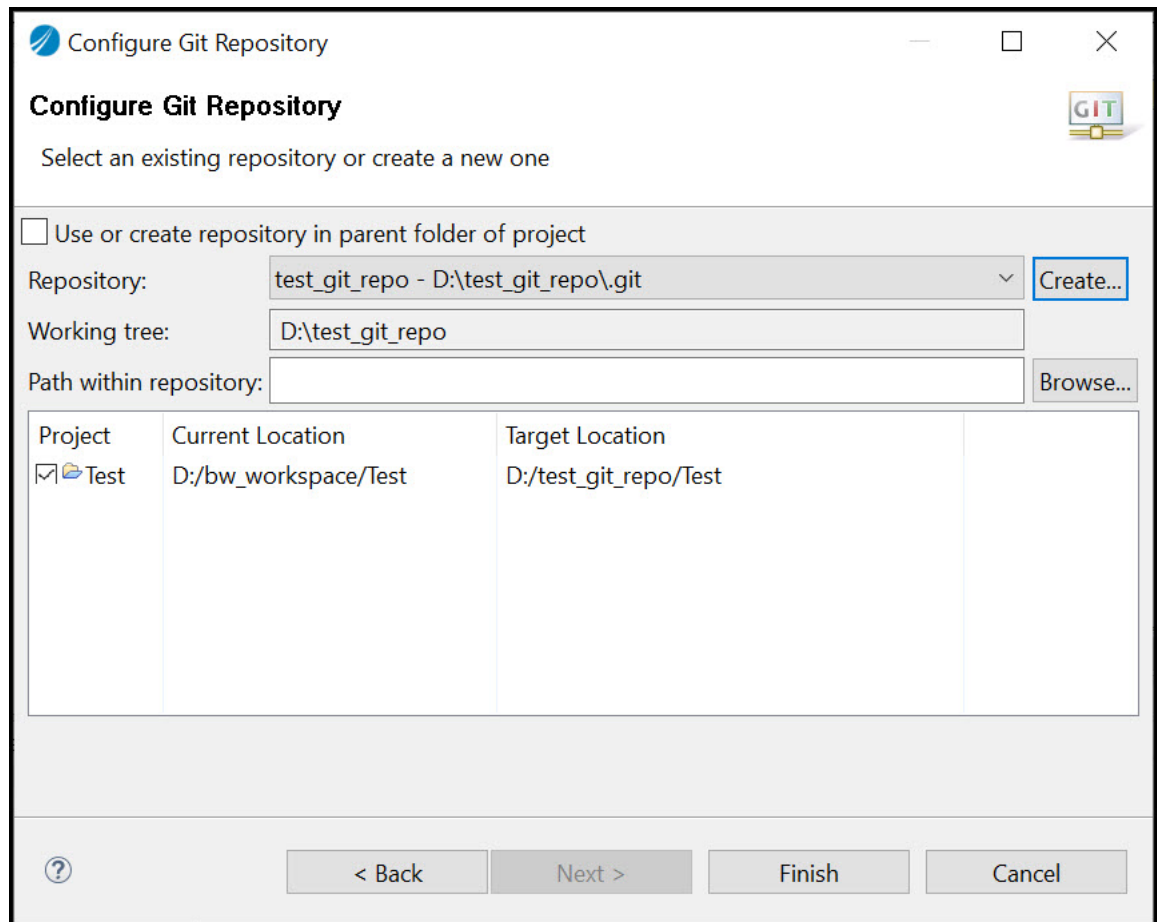
## Generating gitignore Files in Special Folders

### Prerequisites

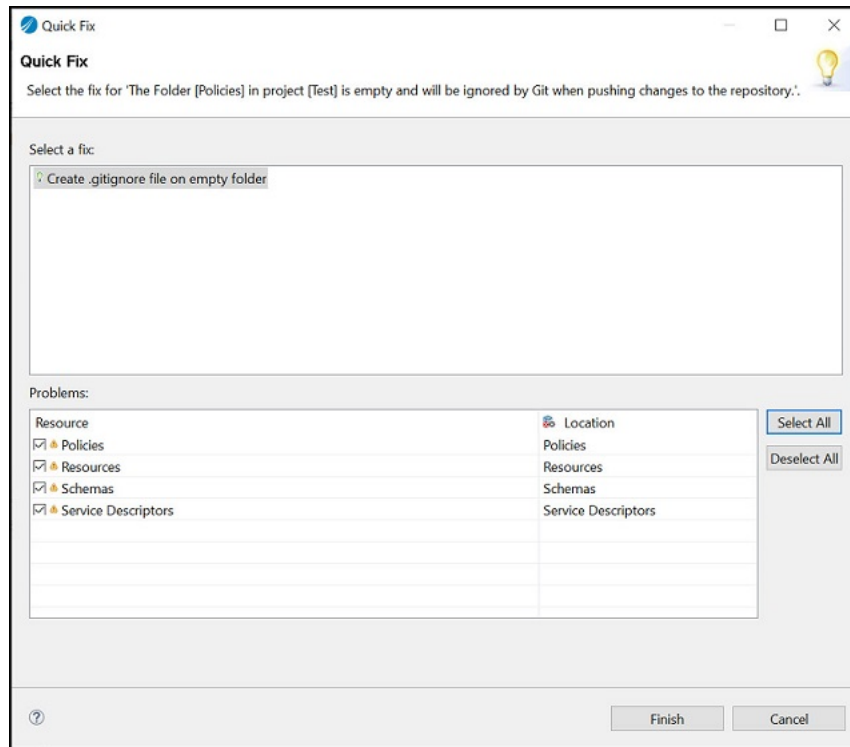
EGit plug-in is configured with TIBCO Business Studio for BusinessWorks and you created an application module.

### Procedure

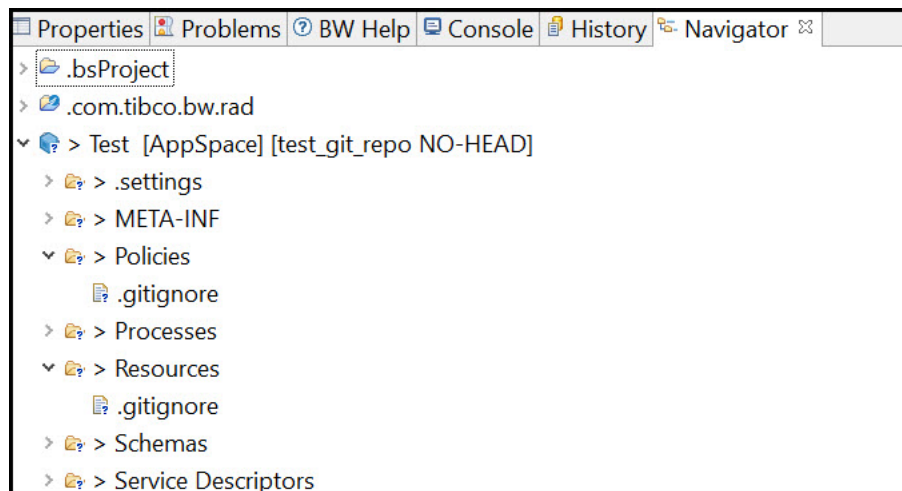
1. In the Project Explorer view, right-click on the project and select **Team > Share Project...**
2. In the Share Project dialog box, select **Git** and click **Next**.
3. In the Configure Git Repository dialog box, either create a new Git repository or select an existing Git repository from a drop down list.



4. Click **Finish** to share the project.  
As special folders are empty, you see warnings in the Problems tab.
5. Expand the warning header to display list of warnings. Select any one warning, right-click and select **Quick Fix**.  
The Quick Fix dialog box opens. In the **Select a fix** section, **Create .gitignore file on empty folder** option is selected. List of Resource folders is displayed along with the Location in the **Problems** section on the dialog box.
6. Click **Select All** to select all check boxes for the resources.



7. Click **Finish** to generate .gitignore files in the special folders.
8. The .gitignore files generated in special folders are visible in the **Navigator** view only. To open Navigator view, select **Window > Show View > Other...** Show View dialog box is displayed.
9. Select **General > Navigator** and click **Ok**.

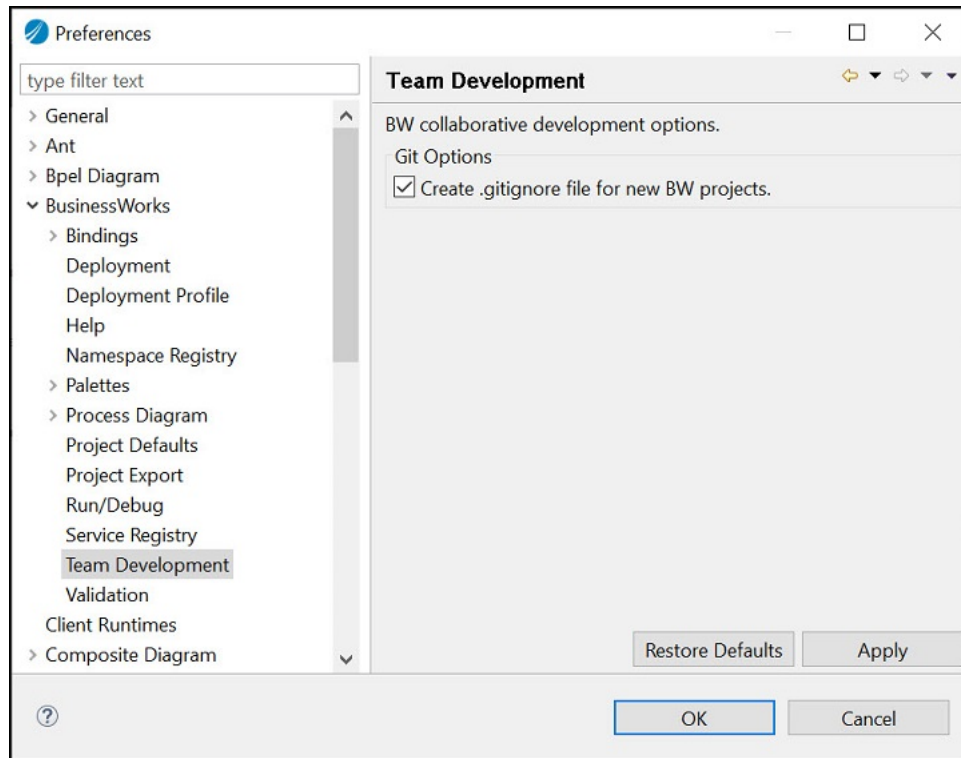


## Generating gitignore Files at Application Module Level

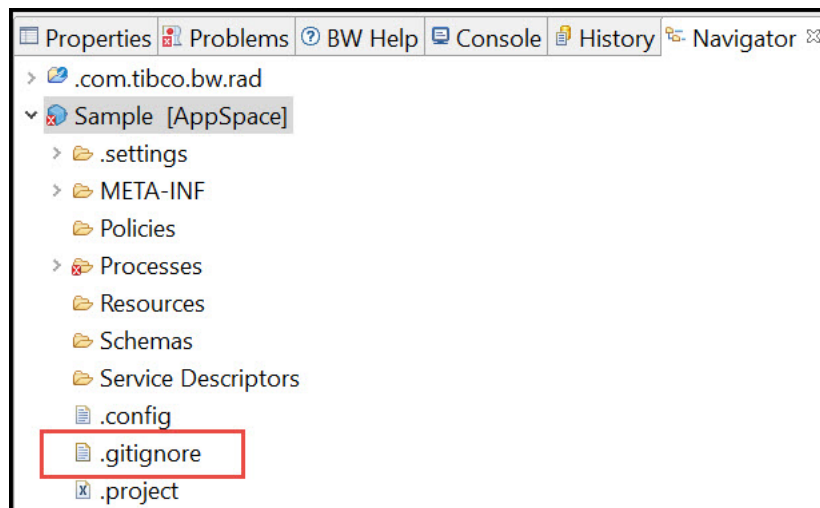
You can generate .gitignore file when creating a new application module.

### Procedure

1. Select **Window > Preferences > BusinessWorks > Team Development** menu.
2. Select the **Create .gitignore file for new BW projects.** check box. Click **Ok**.



When you create a new application module, in the **Navigator** view, select the application module name, right click and select **Refresh** to see the .gitignore file generated.



This method generates .gitignore file at the root level only.

## Synchronizing Module Properties

When an application module has dependent shared modules and if module properties are added, modified, or deleted in a shared module, push the shared module properties in a Git repository. When you perform a Git Pull operation, the application properties are synchronized.

Consider the following scenario:

There are 2 developers collaborating over a Git repository and they are authoring a shared module. Developer 1 adds, modifies or deletes module properties in a shared module in his or her workspace, and then pushes the changes to the repository. When Developer 2 pulls these changes onto his or her

workspace, TIBCO Business Studio™ for BusinessWorks™ detects those changes and synchronizes the application properties automatically.

## Using the bwdesign Utility

The **bwdesign** utility provides a command line interface for creating, validating, importing or exporting resources stored in a workspace.

### Prerequisites

1. To use the bwdesign utility, open a terminal and navigate to `BW_HOME\bin`.
2. Type: `bwdesign -data <TIBCO_BusinessStudio_workspace_absolutePath>`. For example, `bwdesign -data C:\myWorkspace`.

To view arguments and options for a command, open a terminal, navigate to the `BW_HOME\bin` folder, and type **bwdesign help** *command* at the command line.

| Command Name and Syntax                                                               | Description                                                                                                                                                                                                                        |
|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>cd</b><br>SYNTAX:<br><code>cd path</code>                                          | Changes the current working directory to the specified folder.<br>ARGUMENTS: <ul style="list-style-type: none"> <li>• path - The path of the new current working directory</li> </ul>                                              |
| <b>clear</b><br>SYNTAX:<br><code>clear</code>                                         | Clears the command line console.                                                                                                                                                                                                   |
| <b>diagram:gen_diagrams</b><br>SYNTAX:<br><code>diagram:gen_diagrams [project]</code> | Save each process diagram of a project in a .sgv format.<br>ARGUMENTS: <ul style="list-style-type: none"> <li>• outputfolder - Optional argument, It's used to save the diagrams in a given path.</li> </ul>                       |
| <b>edition</b><br>SYNTAX:<br><code>edition</code>                                     | Prints out the edition of this BW Studio                                                                                                                                                                                           |
| <b>execute</b><br>SYNTAX:<br><code>execute file</code>                                | Executes a batch script file containing a set of commands to execute in sequence.<br>ARGUMENTS: <ul style="list-style-type: none"> <li>• file - Script file which contains a set of commands to be executed in sequence</li> </ul> |
| <b>exit</b><br>SYNTAX:<br><code>exit</code>                                           | Exits the command line console.                                                                                                                                                                                                    |



| Command Name and Syntax                                                                                                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>generate_manifest_json</b></p> <p>SYNTAX:</p> <pre>generate_manifest_json [options] [ear_location] [manifest_location]</pre>          | <p>Creates manifest.json from an bw ear file.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>ear_location - The location of the BW EAR file</li> <li>manifest_location - The destination folder that will contain the created manifest.json file.</li> </ul>                                                                                                                                                                                                                                                                                                                                      |
| <p><b>ls</b></p> <p>SYNTAX:</p> <pre>ls [-f -p] [-a]</pre>                                                                                  | <p>List the projects in current workspace or the files in current working directory.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>a - List all the entities including hidden ones.</li> <li>f - List the files in file system</li> <li>p - List the projects in the current workspace</li> </ul>                                                                                                                                                                                                                                                                                                |
| <p><b>pwd</b></p> <p>SYNTAX:</p> <pre>pwd</pre>                                                                                             | <p>Prints the location of the current working directory.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <p><b>quit</b></p> <p>SYNTAX:</p> <pre>quit</pre>                                                                                           | <p>Exits the command line console.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <p><b>setedition</b></p> <p>SYNTAX:</p> <pre>setedition -name -t</pre> <p>EXAMPLE:</p> <pre>setedition -name test.application -t bwcf</pre> | <p>Converts projects from their existing editions to this edition of TIBCO Business Studio™ for BusinessWorks™.</p> <p>If the option <b>-name</b> is not selected this command sets the edition of all the projects in the workspace to the current edition of TIBCO Business Studio for BusinessWorks.</p> <p>Select the option <b>-name</b>, and provide the names of the projects to be converted.</p> <p>Provide comma separated values to convert multiple projects. The <b>-t</b> tag changes the edition to the specified edition. The values to be used for the editions are bwcf, bwe and bwcloud.</p> |

| Command Name and Syntax                                                                                                                                                                                                                                                                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>system:create</b></p> <p>SYNTAX:</p> <pre>system:create [options] [outputfolder]</pre> <p>Alternatively, you can use <b>create</b> command.</p> <p>EXAMPLE:</p> <pre>create application test2.application test</pre> <p>Note that this example generates test2.application for the test application module.</p> | <p>Creates resource(s) in the workspace.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>outputfolder - The destination folder that will contain the created resource in the workspace.</li> </ul> <p>Options:</p> <ul style="list-style-type: none"> <li>--help - Display this help message.</li> <li>application [name] [modules] -v [version] - Create an application project with the given name, including the given module(s).<br/>Optionally, specify the application version using the -v argument.<br/>Version format - major.minor.micro.qualifier e.g. '1.0.0.qualifier'.</li> </ul> |

| Command Name and Syntax                                                                                                                                                                                                                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>system:export</b></p> <p>SYNTAX:</p> <pre>system:export [options] [projects] [outputfolder]</pre> <p>Alternatively, you can use <b>export</b> command.</p> <p>EXAMPLE:</p> <pre>export -ear test2.application -removeunused D:\Samples</pre> | <p>Exports BW artifacts from the specified projects in the workspace to a folder. The artifacts can be ZIP or EAR files.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>• projects - The name of the project(s) to export, separated by commas, e.g. project[,project]*. Must specify at least one project. BW Applications can be exported as EAR files.</li> <li>• outputfolder - The destination folder to contain the exported module(s). Defaults to local folder.</li> </ul> <p>Options:</p> <ul style="list-style-type: none"> <li>• -e, -ear - Export application as a deployable ear file (default). Can be used with application projects. Cannot be used with module projects.</li> <li>• force - Export the BW Application as an EAR file even though there are validation errors. By default, erroneous Applications can be generated as ear files.</li> <li>• -bin, -binary - Export shared model as binary shared module. Can be used with -zip option. Cannot be used with -ear option.</li> <li>• -name [name] - Use the supplied name for the exported module</li> <li>• -pf, -profile name - Export the named profile of the given module.</li> <li>• -removeunused - Exclude unused resources from the application when creating the EAR.</li> <li>• -removediagraminfo - Removes process diagram information when creating .EAR file.</li> <li>• --help - Display this help message.</li> <li>• -z, -zip - Export model as zip file. Cannot be used with -ear option.</li> </ul> |

| Command Name and Syntax                                                                                                                                                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>system:import</b></p> <p>SYNTAX:</p> <pre>system:import [options] files</pre> <p>Alternatively, you can use <b>import</b> command.</p>                                 | <p>Imports flat or ZIP projects into the current workspace.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>files - The names of the folders which contain the target flat projects to import. All the flat projects found in the specified folders will be imported. The folders are separated by commas. By default, zip files will be ignored. If the items to import are zip archives, use <code>-z</code>, <code>-zip</code>, <code>-fz</code>, <code>-fzip</code> options.</li> </ul> <p>Options:</p> <ul style="list-style-type: none"> <li><code>-fz</code>, <code>-fzip</code> - The specified items to import are zip archives located in the folders specified by the arguments. All the zip projects in these folders will be imported, while flat projects will be ignored. Multiple folders are separated by commas.</li> <li><code>-z</code>, <code>-zip</code> - The specified items to import are zip archives specified by the arguments. Multiple zip files are separated by commas.</li> </ul> <p>Output:</p> <p>file status</p> <ul style="list-style-type: none"> <li>file - Name of the project</li> <li>status - Result of the import, either "imported", "ignored", or "failed {message}"</li> </ul> |
| <p><b>system:importpreferences</b></p> <p>SYNTAX:</p> <pre>system:importpreferences [options] file</pre> <p>Alternatively, you can use <b>importpreferences</b> command.</p> | <p>Imports preferences set in the preferences file.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>file - Absolute path of the preferences file to be imported.</li> </ul> <p>Output:</p> <p>file status</p> <ul style="list-style-type: none"> <li>file - Name of the project</li> <li>status - Result of the import, either "imported", "ignored", or "failed {message}"</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Command Name and Syntax                                                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>system:validate</b></p> <p>SYNTAX:</p> <pre>system:validate [options] [modules]</pre> | <p>Validates BW modules in the current workspace. If you don't provide any module name, by default, it validates all modules.</p> <p>ARGUMENTS:</p> <ul style="list-style-type: none"> <li>modules -<br/>The name of the module(s) to validate, separated by commas, e.g. module[,module]*. Defaults to all modules in the workspace.</li> </ul> <p>Options:</p> <ul style="list-style-type: none"> <li>-h,--help - Display help for this command.</li> <li>-d &lt;Directory path&gt;,--directory &lt;Directory path&gt;<br/>- Path of the directory to store validation result.</li> </ul> |

# Best Practices

---

As the business requirements become more complex, so do the business processes that are designed to implement them. TIBCO provides some best practices to help design processes that are readable, reusable, and manageable.

## Control Visibility with Scopes

A scope is similar to a block concept in programming languages and is useful to isolate or encapsulate process variables, thus avoiding conflicts with variable names used elsewhere in the process. Use of scopes helps reduce the number of module properties needed for the entire application, which must be unique for all lexical scopes. When designing or viewing a process in TIBCO Business Studio for BusinessWorks, scope constructs can be collapsed to enhance readability of the process and reduce clutter.

## Promote Reuse with Sub-processes

A sub-process is similar to a sub-routine in programming languages and is useful to keep a block of code small and maintainable. Sub-processes, if declared public, can be called from other processes, thus enabling the logic to be reused.

## Consolidate Literal Values

Keep the number of literal values in process logic and activity configurations to a minimum by consolidating them in the Process Properties tab at the process level. This makes it easier to view and maintain the literal values. In addition, the process properties can be promoted to module properties, which can then be controlled at the application level.

## Externalize with Module Properties

Configuration parameters can be externalized as module properties. At runtime, the values from the module properties are injected into process and activity configuration parameters upon application startup. This allows environmental specific application properties to be set at the time of deployment or in some cases, post deployment. Database password is a good example of a module property.

## Use Profiles for Staging

You can group module properties with the current set of property values into a named profile. An application can have multiple profiles, each having its own set of property values. At run time, you can deploy the same application and stage it multiple times using different profiles.

## Defining Service Contracts

When designing complex business processes, ensure that the service contracts on the interfaces are well-defined.

## Avoid XML Collisions

Avoid defining schema (XSD) or WSDL components with the same qualified names in the same module. Doing so may result in XML collisions at the module level.

If, for some reason, you need to define schema or WSDL components with the same qualified names, then define the schema or WSDL components in separate shared modules. Additionally, configure the process to have unique namespace by specifying the location of the schema document in the **Dependencies** section of the process.

## Close Unnecessary Projects in Workbench

Keep the number of open projects in your Eclipse workbench to a minimum by closing the unnecessary projects. Having too many ActiveMatrix BusinessWorks projects open in the Eclipse workbench may adversely affect the UI performance.

## Use Project Clean

Sometimes TIBCO Business Studio for BusinessWorks reports incorrect validation errors that are not related to design or development issues. It is recommended that you clean your project as it forces Eclipse to discard all build problems and states, and rebuild the projects from scratch. This option can be accessed from the menu **Project > Clean**.

## Manage TIBCO Business Studio for BusinessWorks Workspaces

If you are working with multiple major, minor, or service pack levels of the product, use different workspaces for different versions.

## Increase Log Levels

When debugging issues at design-time, increasing the log levels can provide additional information on the issues. You can customize the log levels for configurations like Debug and Run by editing the respective `logback.xml` configuration files.

The logging configurations are accessible from **Run > Debug Configuration > Advanced > Logging Configuration**. Permissible log level values are `INFO`, `TRACE`, `DEBUG`, `WARN`, and `ERROR`. These levels can be applied to activities, shared resources, bindings, engine, and so on.

## Change the Namespace or Name of a WSDL or XSD Definition

Renaming WSDL definition:

- Right-click the `.wsdl` file, and click **Refactor > Rename WSDL Definition namespace...**

Renaming XSD definition:

1. Right-click the `.xsd` file, and click **Refactor > Rename XSD Schema namespace...**
2. Right-click the `.xsd` file, and click **Refactor > Repair BusinessWorks Projects...**, select the **Refresh Project Cache and do Project Clean** option, and then click **OK**.

## Use Refresh (F5) and Project > Clean

Select the required or all the projects in the Project Explorer view by pressing **Ctrl** + clicking the project folder, and press **F5** on the keyboard to refresh the projects. Or select and right-click the required projects and click **Refresh**. In the Menu bar, click **Project > Clean**.

## Moving Resources

Avoid dragging and dropping the ActiveMatrix BusinessWorks resources that are used in SOAP binding from one place to another.

## Workspace Triggers a Rebuild Process after any Resource is Saved

It is a best practice to allow the rebuild operation to complete before making any additional project changes. This is important when modifying the XSD or WSDL files, because TIBCO Business Studio for BusinessWorks updates all processes that refer the affected files. Making the changes during this progress may lead to workspace corruption and hang issues.

**Project > Clean is Recommended for XSD or WSDL Modifications**

It is recommended to perform the **Project > Clean** operation in case of changes in the XSD or WSDL files.

**The Support for Undo-Redo Operations is Limited**

It is suggested to avoid multiple recursive Undo-Redo on the resources like the WSDL and XSD files. Recommended approach is to save the files (**Ctrl+S**), so you can close and reopen them.

**Project > Build Automatically Option should be Enabled as and when Feasible**

When a resource is changed, the project builders can perform cascading changes right away to update the related resources when the **Build Automatically** option is selected.

**Support for Copy/Paste Actions on ActiveMatrix BusinessWorks Activities and Processes is Limited**

To reuse the Copy/Paste functionality for ActiveMatrix BusinessWorks activities across different modules, consider recreating the activities or using the **Call Process** activity.

**Resolving Errors through Quick Fix Option**

Right-click the errors in the Problems tab to check if the errors can be resolved through a Quick Fix option. This helps to resolve errors faster than manually fixing them.



# Troubleshooting

This section provides information on how to solve some commonly observed issues when working with ActiveMatrix BusinessWorks.

## Mapping and Transforming Data

Some mapping issues and possible resolutions are explained below. This list is not complete but provides examples of messages that might be returned.

For a complete list of error codes, see the *TIBCO ActiveMatrix BusinessWorks™ Error Codes* guide.

- **Issue 1:** "The expression refers to a variable name, `variableName`, that is not defined in the static context".

**Resolution:** Delete the mapping and re-map. If the XSLT function `Copy-Contents-Of` is mapped on the right hand side of the mapper, delete the mapping and re-map.

- **Issue 2 :**

```
"Caused by: org.genxdm.exceptions.GenXDMEException: The prefix 'tns' is already bound to http://NamespaceTest.com/Example Caused by: org.genxdm.exceptions.GenXDMEException: The prefix 'tns' is already bound to http://NamespaceTest.com/Example and cannot also be bound to http://xmlns.example.com/20150212141103"
```

**Resolution:** Select the element on the right side of the activity's mapper, navigate to the Edit Statement panel and clear the **copy-namespaces** check box. See the following image.

