TIBCO BusinessWorks™ Container Edition

Application Development

Version 2.9.2 | August 2024

Contents

Application Development Overview

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

The *TIBCO BusinessWorks Container Edition 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.

> **Note:** 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 needs before the next level of details. This

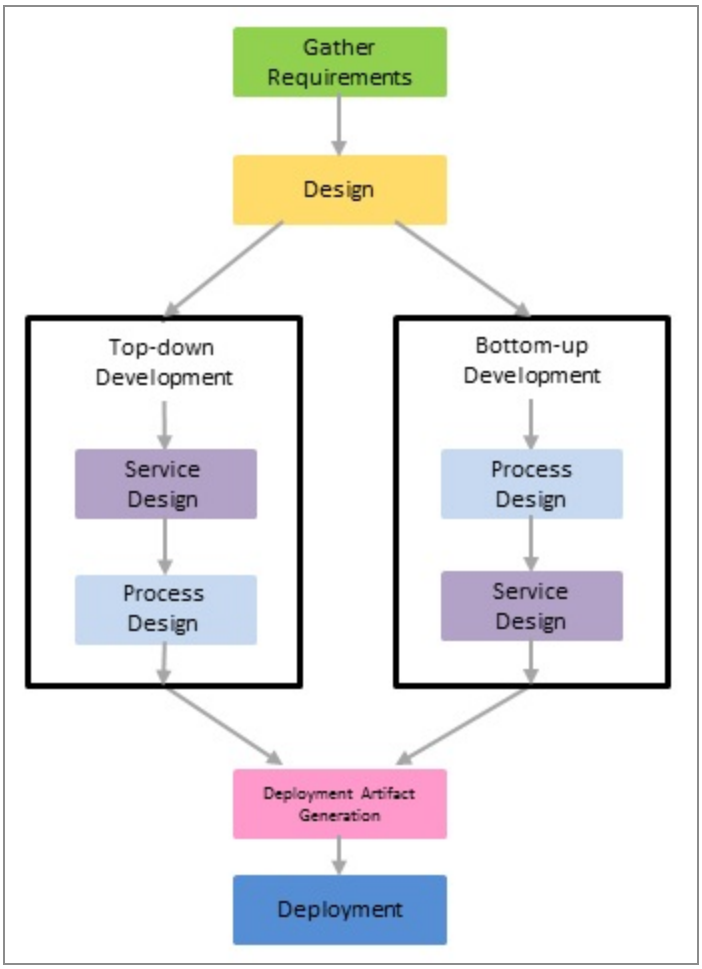
process is repeated until the most granular pieces are designed and implemented. The application is then ready for testing. Top-level services and processes can be designed and developed first before moving to the lower levels.

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.



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 details, see TIBCO BusinessWorks™ Container Edition Application Monitoring and Troubleshooting 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 following table 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, which then run sequentially.

A service-oriented style exposes multiple operations available in a process and each of the operations must be called directly. These operations are not related and can run independently.

Choosing the Modularity

An application module is the smallest unit of resources that is named, versioned, and packaged as part of an application. You can then run it in the TIBCO BusinessWorks Container Edition 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. It can be used by other modules that are part of the same application. Shared modules export their functionality (processes, shared resources, and schema namespaces) to application modules or to other shared modules. When creating a 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 run by the runtime 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 run by the runtime unless used 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. 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, TIBCO BusinessWorks Container Edition 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 TIBCO BusinessWorks Container Edition.

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 toolings such as source folders, JRE libraries the **Use Java Configuration** checkbox 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 is shared by activities across operations that executing as part of the same job.	Uses conversations to enable correlation.
Stateless Processes	Does not maintain state.	Data is not shared.	No conversations.

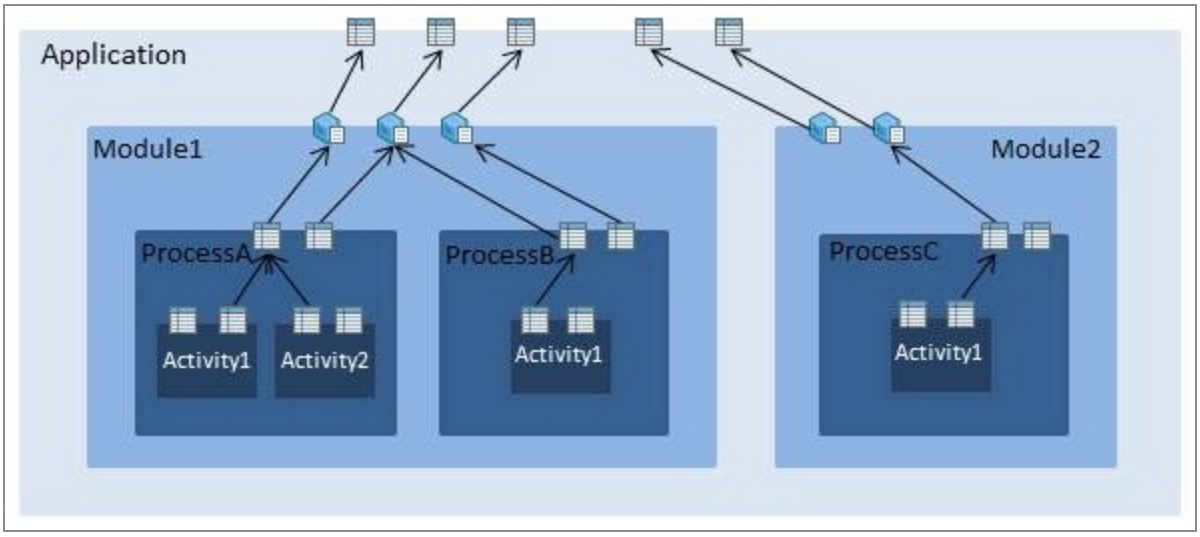
Choosing Between Properties and Variables

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

Property	Scope/Visibility	Datatype	Values	Additional Information
				application level.
Module Properties	• 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.	• 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.
Shared Module Properties	• 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	Literal or a shared resource reference.	• Literal or a shared resource reference. • Private module property values cannot be edited from the Admin UI.	• 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	Scope/Visibility	Datatype	Values	Additional Information
	from the Admin UI.			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.	• 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 BusinessWorks™ Container Edition 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 errors or faults thrown at the selected level.

Note: 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.

Note: 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.

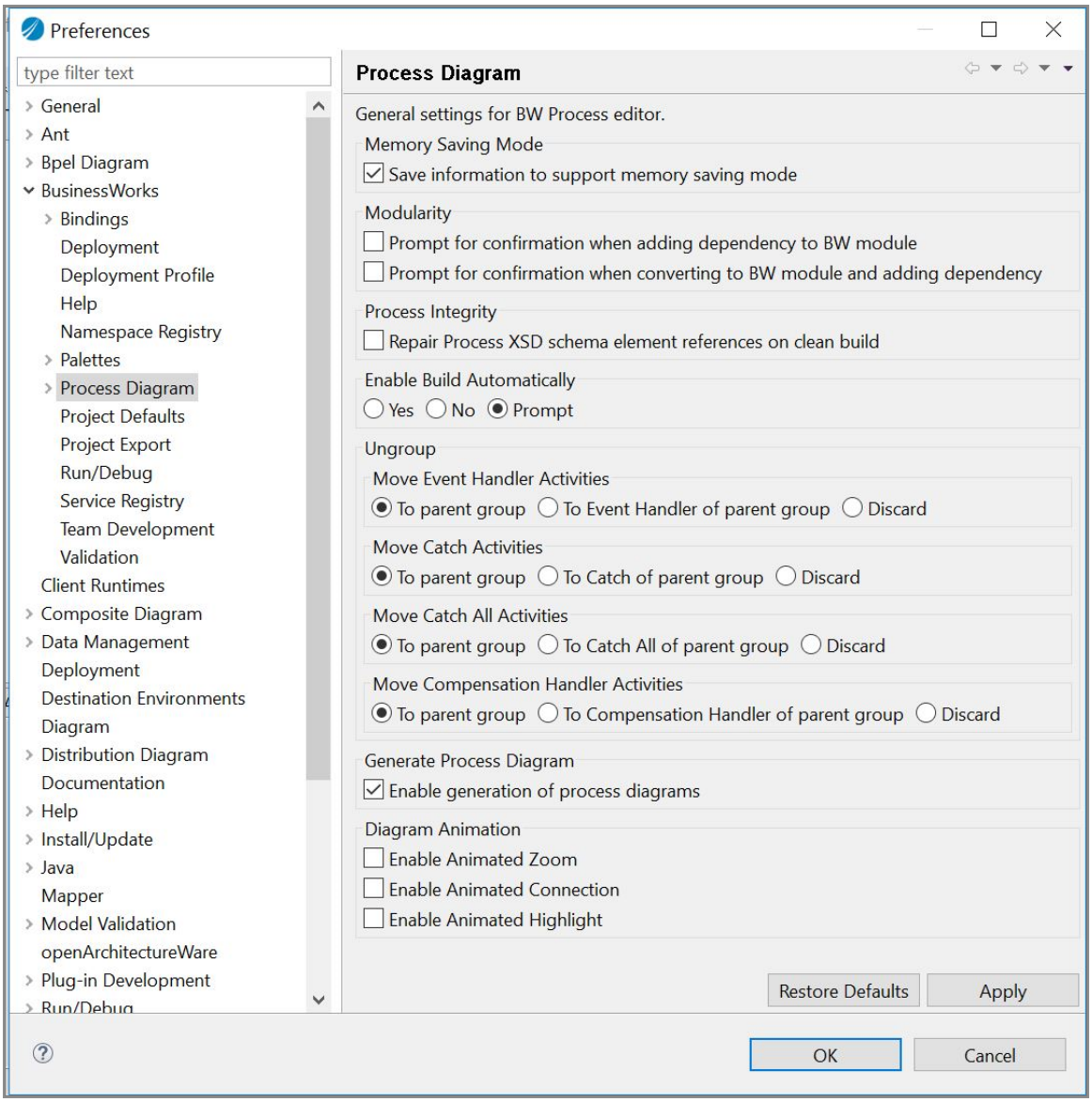
Service	Data Abstraction	State Information	Overhead of Additional Parameters (Headers or other SOAP elements)
REST Services	Resources	Stateless	Less
SOAP Services	Operations	Stateful	High

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. By default, the Memory Saving Mode is enabled. You can disable it by unselecting the **Save information to support memory saving mode** checkbox 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.



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

TIBCO Business Studio™ for BusinessWorks™ Essentials

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

Note:

- If you are familiar with the TIBCO Business Studio for BusinessWorks Essentials UI, skip to the section Developing a Basic Process.
- TIBCO BusinessWorks™ Container Edition 2.8.0 supports Java 11 on the Windows, Mac, and Linux platforms.

Starting TIBCO Business Studio for BusinessWorks

To start TIBCO Business Studio for BusinessWorks on Windows, select **Start > All Programs > TIBCO_HOME > TIBCO Business Studio for BusinessWorks 1.0 > Studio for Designers**. On Linux or Mac OS, select the TIBCO Business Studio for BusinessWorks executable located at `<TIBCO_HOME>/studio/4.0/eclipse/`.

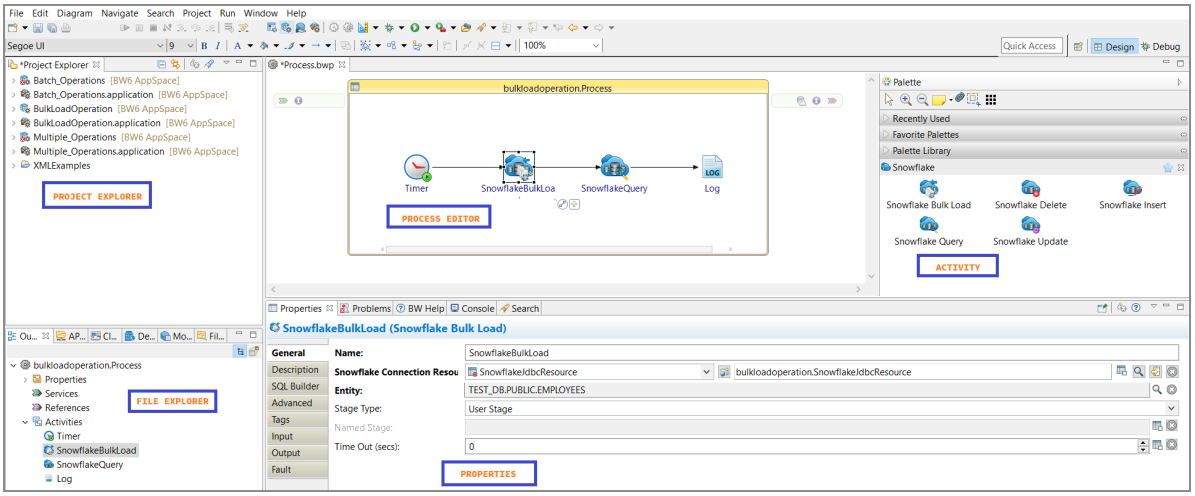
On the Workspace Launcher dialog, accept the default workspace or browse to create a new workspace, and then click **OK**. TIBCO Business Studio for BusinessWorks starts and the default development environment, called a *workbench*, displays. A welcome screen is displayed in the window when a workspace is opened for the first time.

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

- Install the JavaHL libraries. For more information, see http://subclipse.tigris.org/wiki/JavaHL.
- Update the SVN interface to use SVNKit instead of JavaHL. Select **Window > Preferences** and in the Preferences dialog, 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 is an Eclipse-based integration development environment that is used to design, develop, and test ActiveMatrix BusinessWorks applications. The studio provides a workbench in which you can create, manage, and navigate resources in your workspace. A *workspace* is the central location on your computer where all data files are stored.



The following table introduces the workbench UI elements highlighted in the image:

UI Element	Description
Menu	Contains menu items such as File, Edit, Navigate, Search, Project, Run, Window, and Help.
Toolbar	Contains buttons for frequently used commands such as: • New  • Save  • Enable/Disable Business Studio Capabilities  • Create a new BusinessWorks Application Module  • Debug  • Run 
Perspectives	Contains an initial set and layout of views that are required 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	Lists the resources and helps you navigate within 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_nameview_name*** from the main menu.
Editors	Provides 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.

UI Element	Description
Palette	Contains a set of widgets and a palette library. A *palette* groups activities that perform similar tasks, and provides quick access to activities when configuring a process.

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

Designing a Process

Design a process in TIBCO Business Studio for BusinessWorks to implement the business logic. For more information, see Developing a Basic Process.

Testing and Debugging an Application

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 the toolbar.

To execute and debug the application, select **Run > Debug** from the main menu, or click  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.
- Arguments such as the target operating system, target architecture, engine properties, and so on.
- 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.

Project Explorer

The Project Explorer allows you to view your project structure by displaying the artifacts that the project is made up of in a tree-like structure.

You double-click the process in the Project Explorer to open it in the Process Editor.

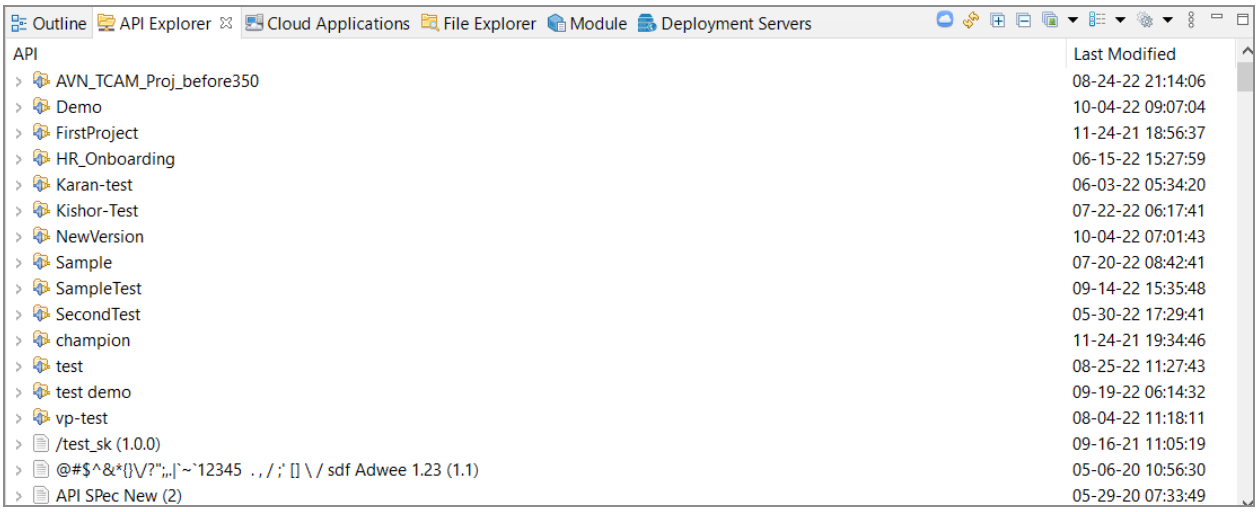
Outline

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.

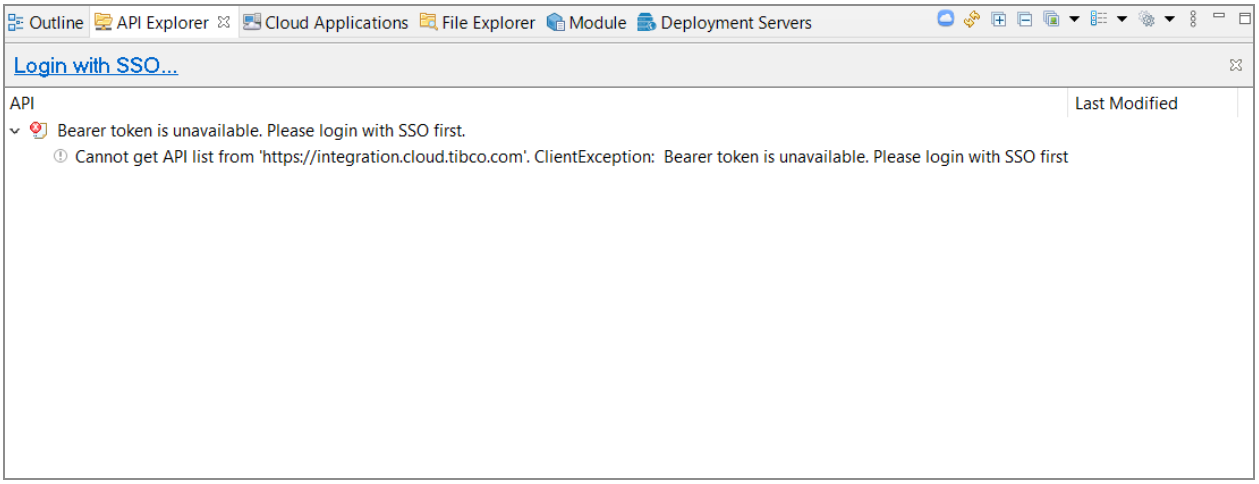
Use this view while you are actively editing a process to select an artifact and see its properties right away.

API Explorer

Displays a connected view of the TIBCO Cloud™ Integration API Modeler residing in the cloud. This view shows abstract APIs that are created in an 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, select the **API Explorer** tab that displays a login view.



Click the SSO Login hyperlink (Login with SSO...) or SSO Login icon ( ) and enter your credentials to authorize the default-selected region.

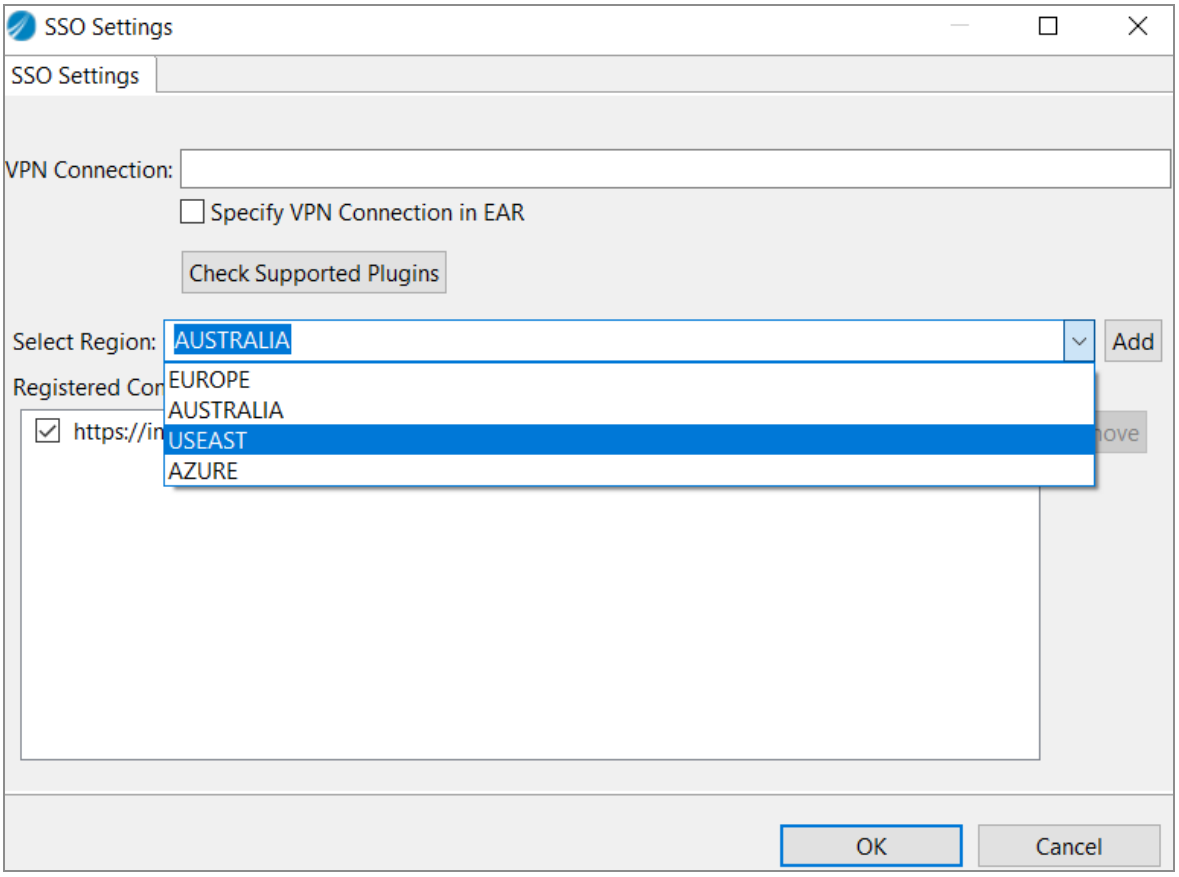
Adding a registry

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

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

To create a registry:

1. Open the **SSO Settings** dialog.
2. Select the appropriate Registry/Region from the dropdown list and click **Add**.

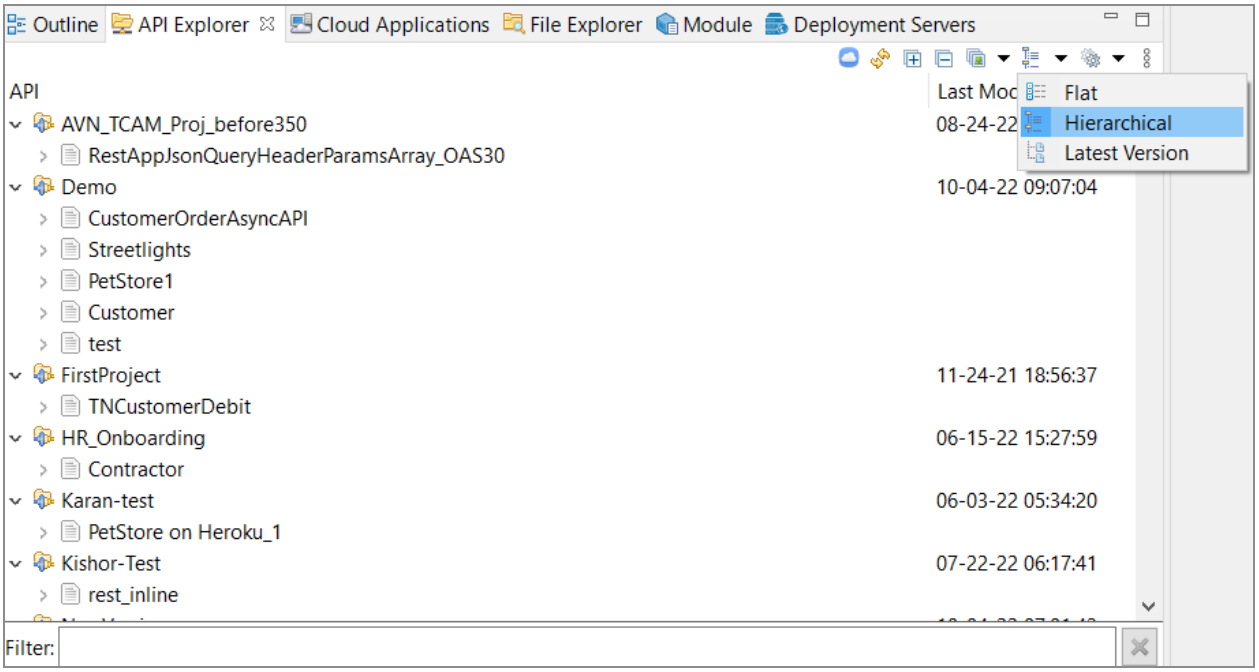


3. Select the specific **Registered Configuration(s) (Registry/Region)** checkbox.
4. Click **OK**.

Note: Once you select the new registry/region, you must Log in again.

Setting the presentation of the APIs

In this dialog, 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. Each API's version number is displayed next to its name in parentheses. If there are multiple versions of the same API, each version is shown as a separate API. Multiple APIs are displayed with the same name but different version numbers.
- **Hierarchical** - displays every API as a hierarchy of API name label, with a 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.

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 that 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:

- SSO Login
- Refresh

- Expand All
- Collapse All
- API Presentation
- View Menu

Note: The TIBCO Cloud API Modeler (TCAM) APIs are now available on api.cloud.tibco.com. TIBCO ActiveMatrix BusinessWorks™ leverages to make the API specs from the TIBCO Cloud API Modeler discoverable in the **API explorer**.

API Presentation

Configure how you want your API to appear in this view. The three types of presentations available are:

- **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** - If one or more APIs in your registry has multiple versions, selecting this option shows only the latest version of the API and hides the older versions.

Other Configurations

Group by API registry - Groups the APIs according to the registry from which they were discovered.

Show API Registry URL - Displays the URL of the APIs next to the registry name.

Check Supported Plug-ins - This button refreshes the supported list of plug-ins from TIBCO BusinessWorks Container Edition. When you import an existing project that uses plug-ins, you can validate that the plug-ins used in the project are supported in TIBCO

BusinessWorks Container Edition by clicking this button. A message is displayed indicating that the supported plug-ins are synchronized.

This list represents the plug-ins that is available to your projects in TIBCO BusinessWorks Container Edition during runtime. In order to use a plug-in during design time, you must have the plug-in installed locally on your machine. If your project uses a plug-in that is not supported in TIBCO BusinessWorks Container Edition, an error message is displayed while pushing the project to the cloud.

Developer Hub

TIBCO Developer® Hub is the Developer Portal for the TIBCO® Control Center. It is a one-stop portal that has a centralized catalog, helps developers restore order to microservices and infrastructure, and ships code quickly.

The Developer Hub tab in TIBCO Business Studio for BusinessWorks configures the TIBCO Developer Hub with TIBCO BusinessWorks Container Edition. For more information, see "Configuring TIBCO Developer Hub" in the TIBCO® Control Center.

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.

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 TIBCO BusinessWorks Container Edition 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. For the W3C definition of NCName, see http://www.w3.org/TR/xmlschema-2/#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, - (hyphen), _ (underscore).
- Cannot include the following characters: @, :, $, %, &, /, +, ,, ;,), 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]
```

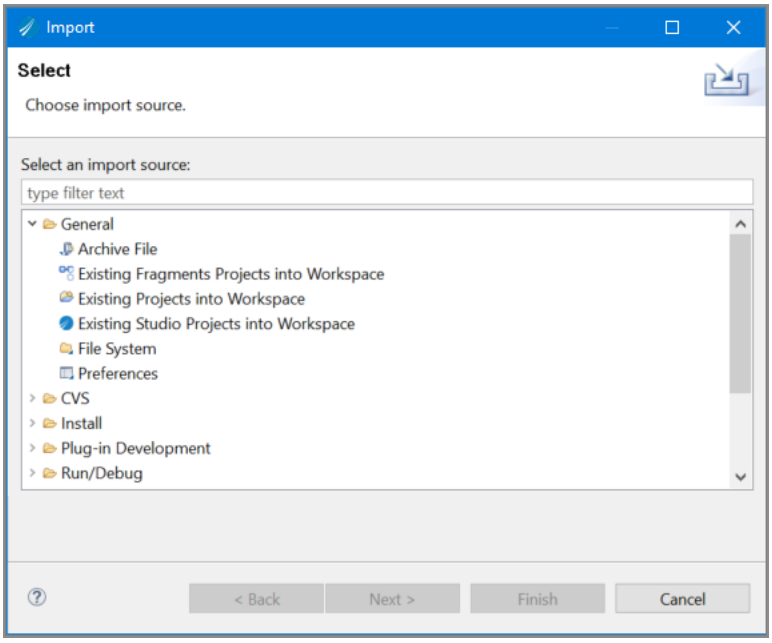
Importing an Existing Project into Workspace

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

Procedure

1. Go to **File > 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 **Next**.

   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.

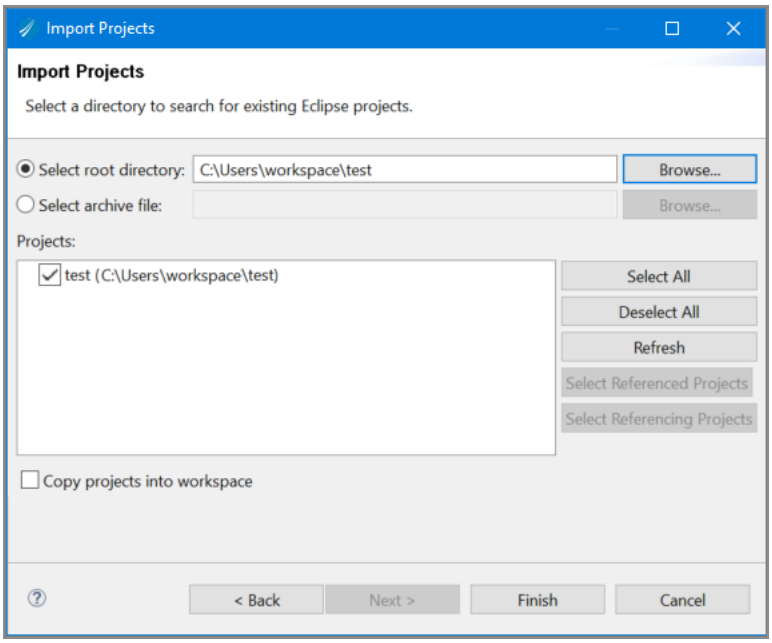
   Browse For Folder wizard is displayed.

   > **Note:** 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 projects to import.
9. Select the **Copy projects into workspace** option.

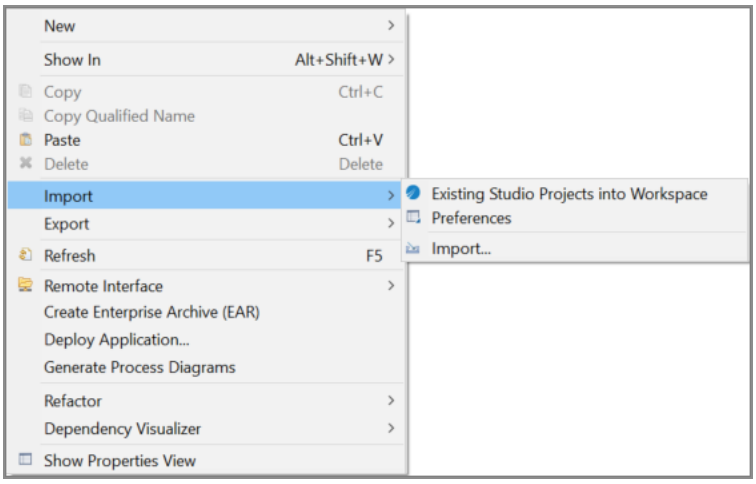


10. Click **Finish**.

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

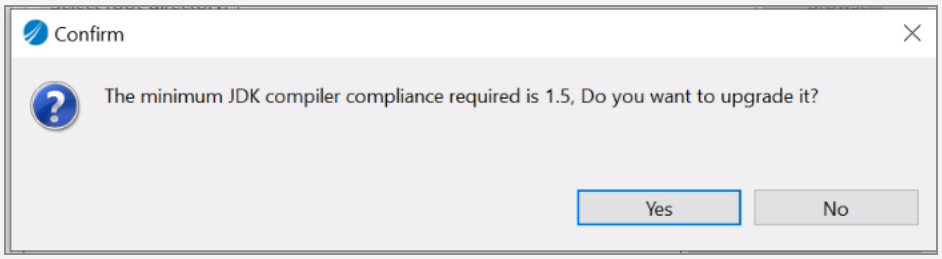
11. Optionally, you can import the projects by right-clicking the **Project Explorer** and navigating to **Import > Existing Studio Projects into Workspace**.

    The **Import** wizard is displayed with the **Import Projects** page, and the **Select** page is skipped.

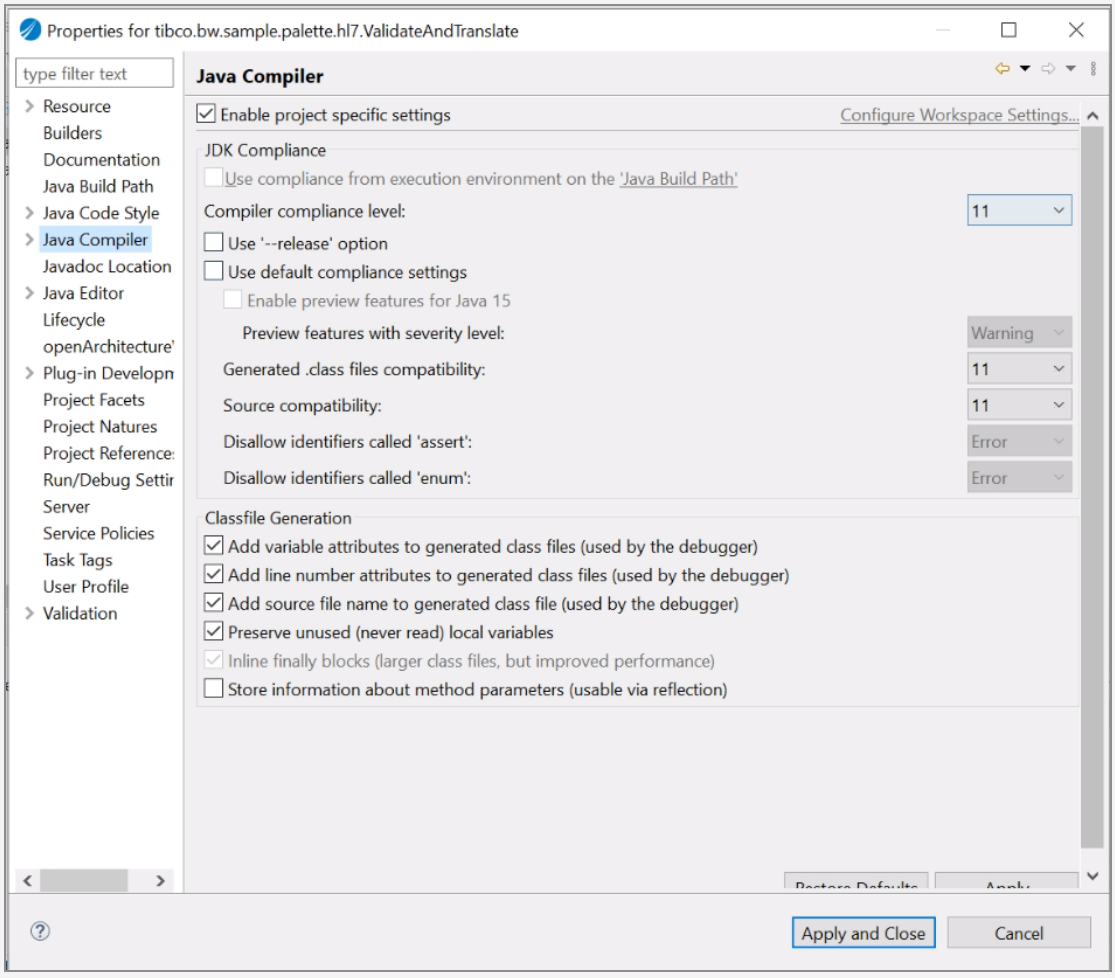


Note: The import functionality is also available from the command line interface. For more information, see Using the bwdesign Utility.

Important: The minimum JDK compiler compliance required is 1.5. If the project that you are importing with Java nature has JDK compiler compliance less than 1.5, TIBCO Business Studio for BusinessWorks imports the project as it is and you may see Java-related problems in your project. You can then choose to upgrade the JDK compiler compliance version to the one greater than 1.5 by clicking the **Yes** button on the **Confirm** window. When you click **Yes**, the default Java version of the environment is set to 11.



By default, the Java Compiler settings, available on right-clicking the project in the project explorer, has the JDK Compliance set to 11.


Properties for tibco.bw.sample.palette.hl7.ValidateAndTranslate
type filter text
Resource
Builders
Documentation
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
Lifecycle
openArchitecture'
Plug-in Developn
Project Facets
Project Natures
Project Reference
Run/Debug Settir
Server
Service Policies
Task Tags
User Profile
Validation
Java Compiler
Enable project specific settings
Configure Workspace Settings...
JDK Compliance
Use compliance from execution environment on the 'Java Build Path'
Compiler compliance level:
11
Use '--release' option
Use default compliance settings
Enable preview features for Java 15
Preview features with severity level:
Warning
Generated .class files compatibility:
11
Source compatibility:
11
Disallow identifiers called 'assert':
Error
Disallow identifiers called 'enum':
Error
Classfile Generation
Add variable attributes to generated class files (used by the debugger)
Add line number attributes to generated class files (used by the debugger)
Add source file name to generated class file (used by the debugger)
Preserve unused (never read) local variables
Inline finally blocks (larger class files, but improved performance)
Store information about method parameters (usable via reflection)
Apply and Close
Cancel

Importing and Exporting the Studio or Eclipse Preferences

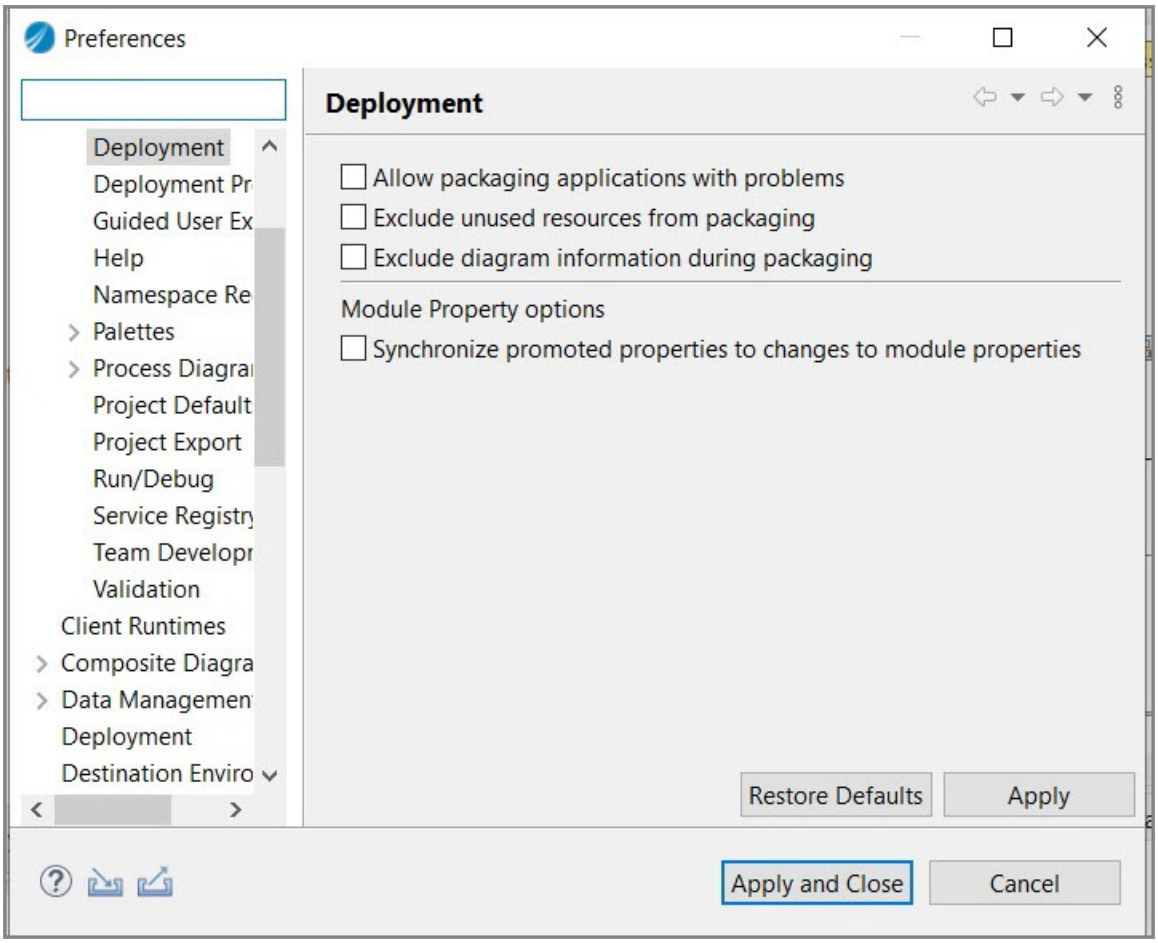
You can also import the Studio or Eclipse preferences set in the other user's workspace and export the Studio or Eclipse preferences set in your workspace. These preferences are stored in a file with an extension `.epf`. You can select any preferences that you want to import or export.

Exporting the Studio or Eclipse Preferences

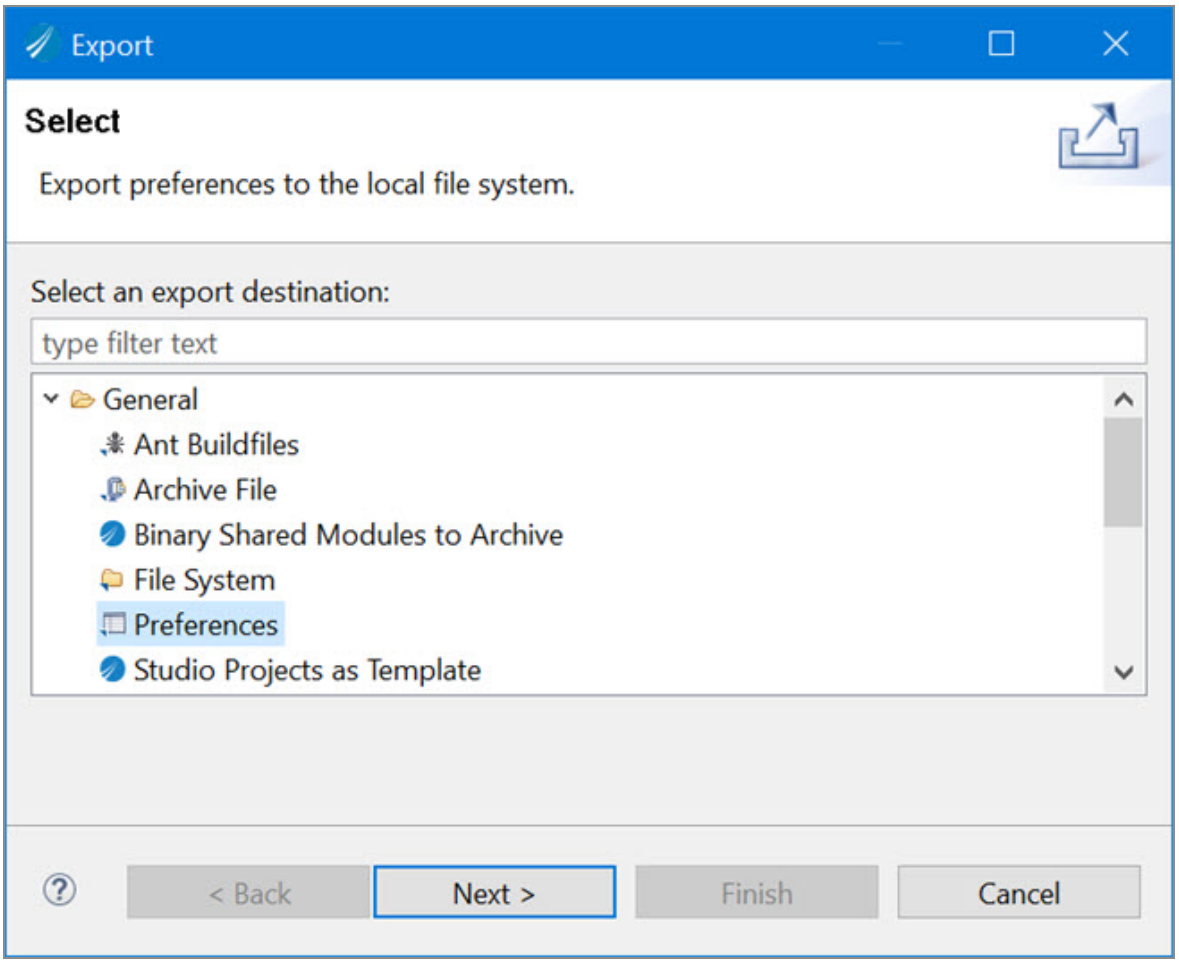
The procedure to export the *Deployment* preference present under *BusinessWorks* category is as follows:

Procedure

1. Open TIBCO Business Studio for BusinessWorks, navigate to **Window > Preferences**. The Preferences wizard is displayed.
2. Expand **BusinessWorks** and select **Deployment**. Select or clear the checkboxes to set the preferences as required, click **Apply and Close**.



3. Navigate to **File > Export**. The Export wizard is displayed. Or click  at the bottom left of the Preferences wizard to export an `.epf` file.



4. Select **General > Preferences** and click **Next**.
5. Select or clear preference checkboxes as required.
6. Click **Browse** to save the preferences in the `.epf` file, enter the name for the file to be exported, and click **Save**.
7. Click **Finish**.

Result

On successful export, the `.epf` file is created in your local file system at the given path.

Importing the Studio or Eclipse Preferences

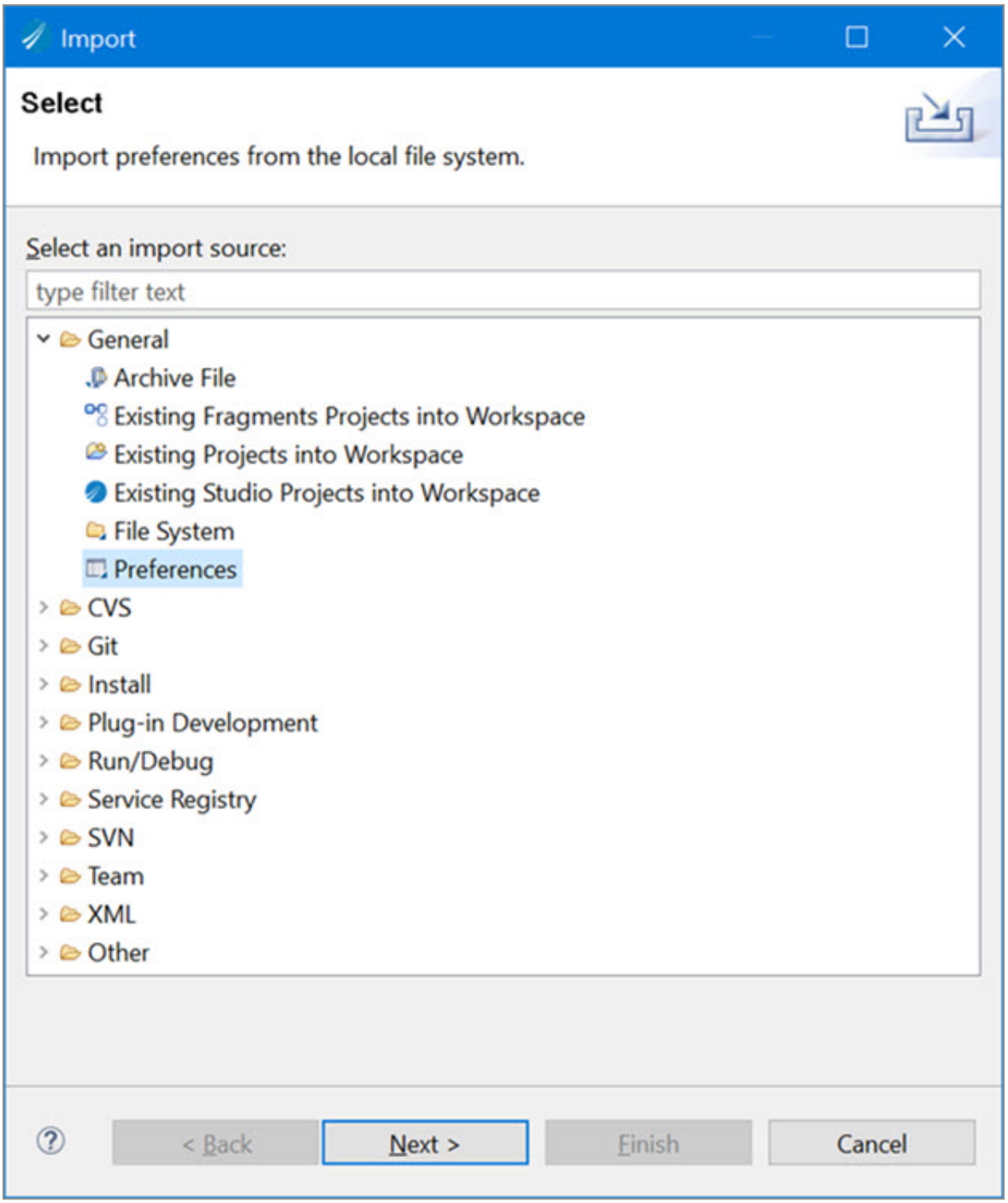
Following are the steps to import preferences file:

Before you begin

Ensure that the `.epf` file to be imported is available in your local file system.

Procedure

1. Open TIBCO Business Studio for BusinessWorks, navigate to **File > Import**. The Import wizard is displayed. Or click  at the bottom left of the Preferences wizard to import an `.epf` file.



2. Select **Preferences** and click **Next**.
3. Click **Browse** to select the `.epf` file to be imported.
4. Select or clear preference checkboxes as required.

5. Click **Finish**.

Result

On successful import, you can see the preferences from the imported `.epf` file in the TIBCO Business Studio for BusinessWorks **Window > Preferences**.

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. Configuring a Process 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.

Note: Conceptual information about processes and their use is provided in the TIBCO BusinessWorks Container Edition *Concepts* guide.

Creating an Application Module

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

The New BusinessWorks Application Module wizard helps create an application module. There are multiple ways to open 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 checkbox and browse to select the location to be used.
3. **Version**: Version of the application module.
4. **Deployment Target**: Select one or more required deployment platforms.

   > **Note:** Optional, you can set the default deployment profile to create applications, and migrate the existing TIBCO BusinessWorks Container Edition projects with the set preference. Navigate to **Window > Preferences > BusinessWorks > Deployment Profile**.

5. Depending on the deployment platform you select, the target platform name follows the project name. For example, **tibco_bw_sample_palette_http_requestresponse [Container, TIBCO Cloud, BW6 AppSpace]**.
6. **Create empty process**: Selected by default to create an empty process with the specified name (default: Process). Clear the checkbox 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 checkbox 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**.

Note: You can add identical package names in two 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 checkbox 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).

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

Note: 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 BW6 AppSpace and uses SharedModule1

Application2 : Configured to BW6 AppSpace and uses SharedModule1

SharedModule1 : Configured to BW6 AppSpace

If you configure the deployment target platform for Application2 as Container, then SharedModule1 is configured to both BW6 AppSpace and Container.

5. Depending on the deployment platform you select, the deployment target names follow the project names.

   For example, **tibco_bw_sample_palette_http_requestresponse [Container, TIBCO Cloud, BW6 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

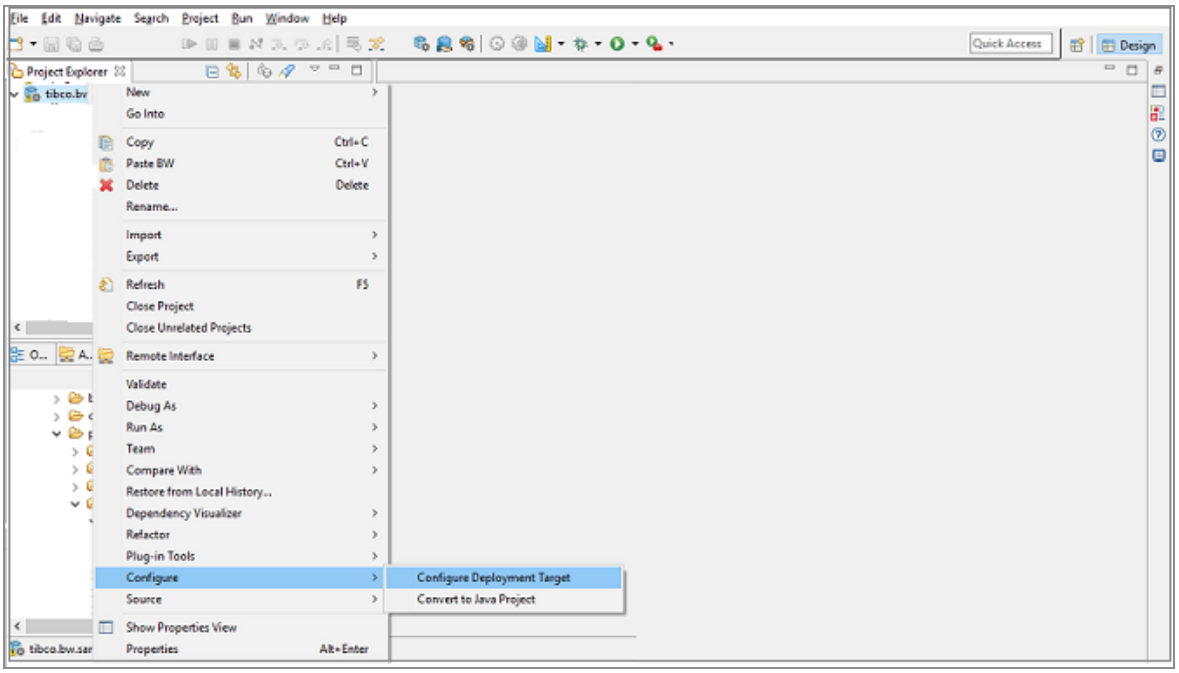
A shared module with the specified name is created and opened in the workbench.

Reconfiguring Deployment Target

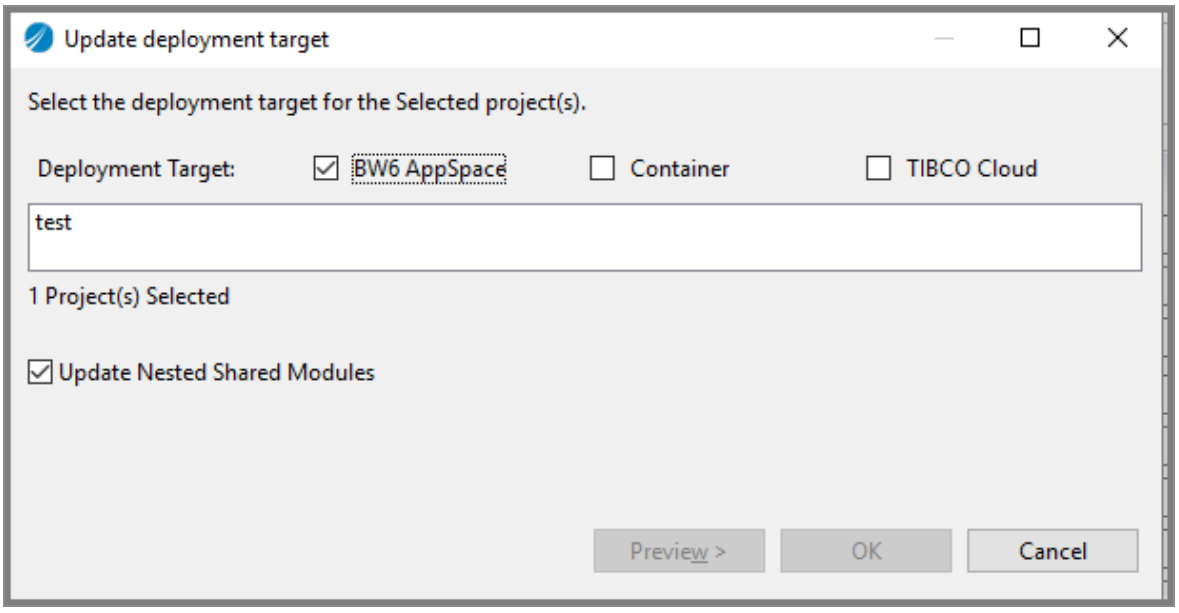
Applications can be reconfigured to run on different platforms or can be configured to more than one platform at the same time. Using the **Configure Deployment Target** option, applications can be developed and run on the Enterprise edition (BW6 AppSpace), Container, or the TIBCO Cloud edition (TIBCO Cloud).

Procedure

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



2. In the **Update deployment target** window, select one or more target platforms to configure or reconfigure the application.



3. The **Update Nested Shared Modules** checkbox 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 project, only the palettes and activities supported by one or more target platforms configured for a project are displayed.

> **Note:**
> - You can also 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**
> - If the deployment target is set to **Container** then **Push to Platform** option is visible on right click of an `.application`.

Generating an EAR

Application archives are enterprise archive (EAR) files that are created in TIBCO Business Studio for BusinessWorks. An EAR file is required when deploying an application.

Procedure

1. In the Project Explorer view, right-click **project.application**.
2. Select the **Create Enterprise Archive (EAR)** option.

   The Create Application Archive (EAR) is displayed.
3. Enter the EAR location.
4. Check the **Use Default EAR file name** checkbox for the default EAR file name or specify a name by selecting the **EAR File Name** checkbox.
5. To export the EAR without the `.substvar` files, select the **Export EAR without profiles** checkbox .
6. Click **Finish**.

A `manifest-bwce.json` file is generated in the EAR is for a TIBCO BusinessWorks Container Edition project. The "_SlAsH_" character present in the appProperties section of this file is replaced with the actual "/".

Generating the manifest.json File Using the bwdesign Utility

In order to push an application created in TIBCO BusinessWorks™ Container Edition 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:

   > **Note:** 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 BusinessWorks Container Edition or TIBCO Cloud™ Integration to VMware Tanzu, the application manifest (`manifest.yml`) file is required. In TIBCO BusinessWorks Container Edition the `manifest.yml` file can be created from the Context menu option, **Create Manifest YML** for an application in the **Project Explorer** view.

Manifest File Editor

The **Application Details** tab has the following fields:

Application Details

Field	Description
Name	The name is auto populated when the manifest file is created. This is an editable field. When you rename the application module, then the name is updated in the `Manifest.yml`.
Memory	The memory field, to specify the memory limit for all instances of an app, is auto populated when file is created. The default value is 512M. This attribute requires a unit of measurement: M, MB, G, or GB, either in uppercase or in lowercase followed by the numeric value
Timeout	The timeout field is auto populated with default value as 60. Only numeric values are allowed for timeout.
Build Pack	The build pack name needs to be specified by the user. By default, it is empty.

Environment Variables

To select environment variables, click the variable chooser button on the right hand side of the table. Environment variables can still be used if there are no default environment variables listed.

Services

The services can be selected from the chooser feature. The services that are present in VMware Tanzu environment is displayed in the chooser. The **Add** button is also used to define the services. Specify the service name once a new service is added.

> **Note:** The files can be edited from the source view. If there is an error while editing the file in the source view, the simplified editor becomes read-only until the error is resolved in the source view.

Exporting the EAR file

The `Manifest.yml` file is exported when you create an EAR file using the **Create Enterprise Archive** option. The `Manifest.yml` file is copied to the same directory as the EAR file and it is not be present in the Jar file inside the EAR file. If a `Manifest.yml` file already exists in that directory, then it is overwritten. The EAR path attribute is updated automatically in the exported manifest.yml file.

Exporting the EAR using bwdesign CLI

The `Manifest.yml` file can also be exported through bwdesign CLI using the export command, in the manner similar to the Create Enterprise Archive command on the Tibco BusinessWorks Studio GUI.

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.

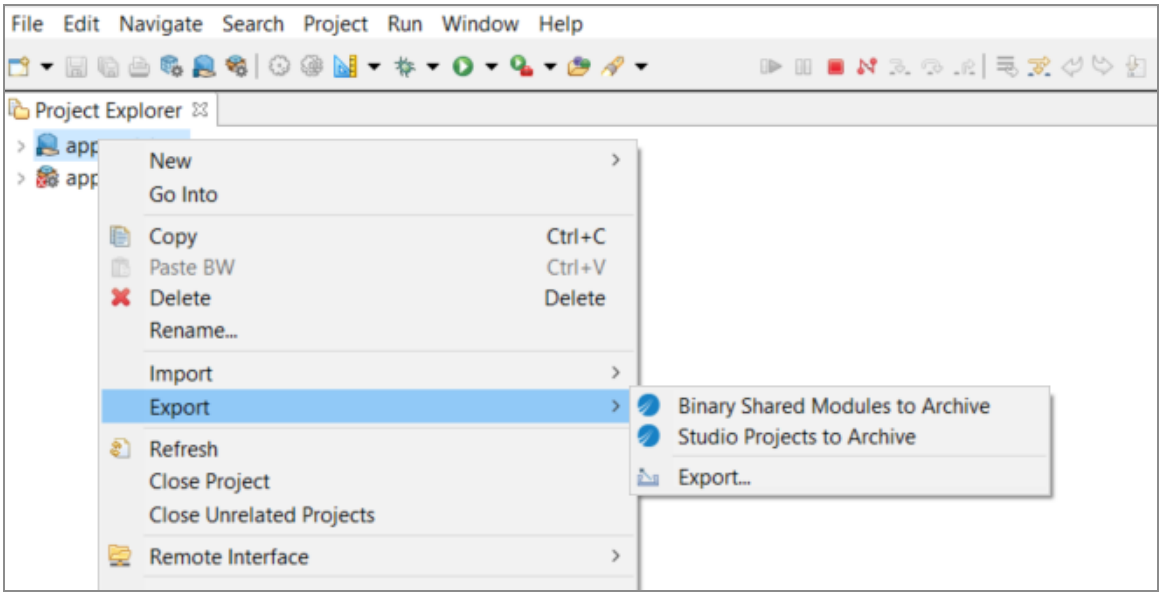
To export a shared module as a binary shared module, begin by implementing the process you want to share. The process must have a descriptive name and a description. Next, test the process by calling it from a test application. Once satisfied, you create a zip archive file for the project which contains 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.

Tip: 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

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 checkbox 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

Before you begin

- 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] -path

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, import 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. 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

  > **Note:** Because the schema files are in plain text, you can modify them. Whenever you import a newer version of the module into your workspace, your modifications to the schema files are 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 displays the Overview page only. All other fields are disabled

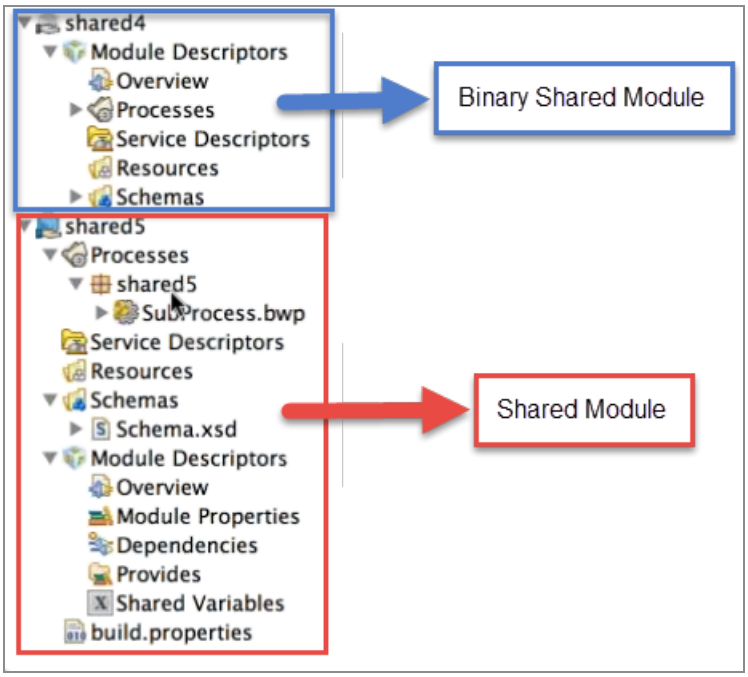
You can implement a **Call Process** activity that invokes the functionality in the binary shared module. When deploying your application, the binary shared modules are included in the application like any other shared module.

Difference between a Shared Module and a Binary Shared Module

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

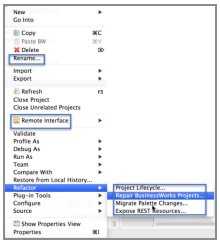
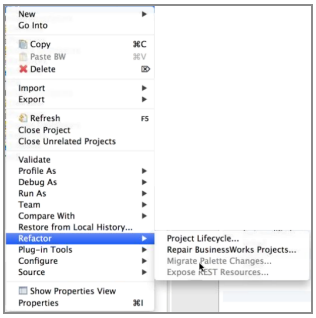
In Project Explorer

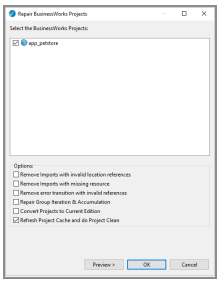
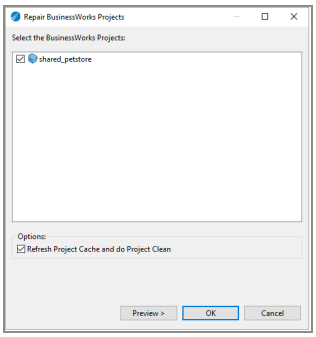
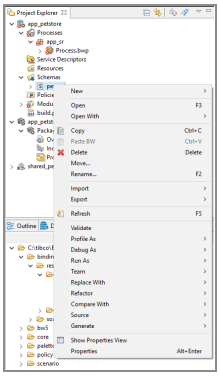
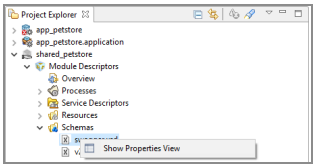
The image below shows you the difference between a shared module (shared5, in the image below) and a binary shared module (shared4). Notice that almost all the editable artifacts (such as Module Properties, Dependencies and Shared Variables) are missing from the binary shared module tree. This is one 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
At the project level: Right-click menu from process name		

Context Menu	Shared Module	Binary Shared Module
Repair BusinessWorks Project dialog		
Context menus at the following levels: • Processes • Service Descriptors • Resources • Schemas		

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** checkbox and click **OK**.

Importing Binary Shared Modules

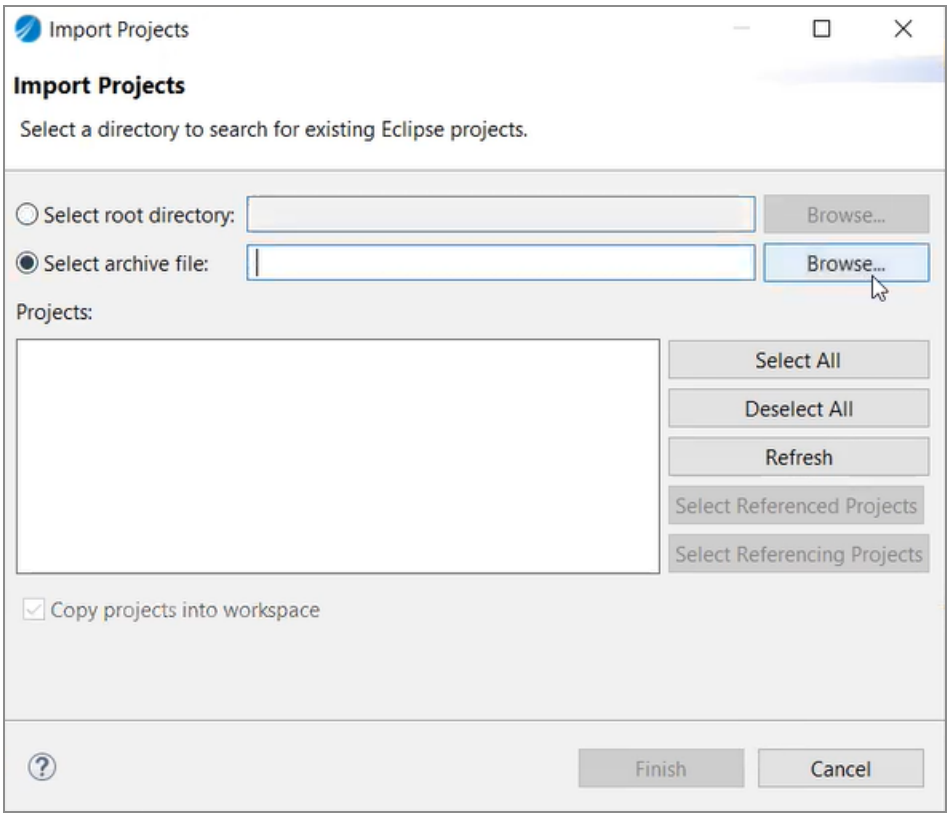
A binary shared module can be imported to TIBCO Business Studio for BusinessWorks by using the following methods:

- Using Import Projects Option
- Using the Dependencies Tab on Application Module

Using Import Projects Option

Procedure

1. In the Project Explorer window, right-click and select **Import > Existing Studio Projects into Workspace**.



2. In the Import Projects dialog, select the **Select archive file** option and click **Browse**.
3. Navigate to the archive location of the binary shared module and select the binary shared module archive file to be imported. Then click **Open**.

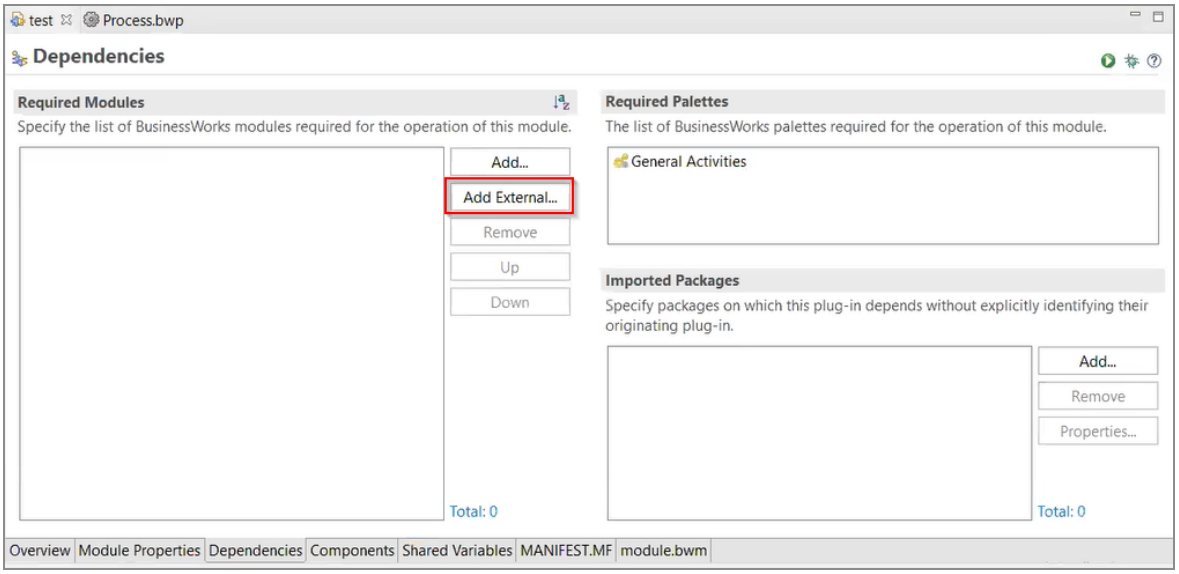
4. Click **Finish**.

   The binary shared module is imported in your project.

Using the Dependencies Tab on Application Module

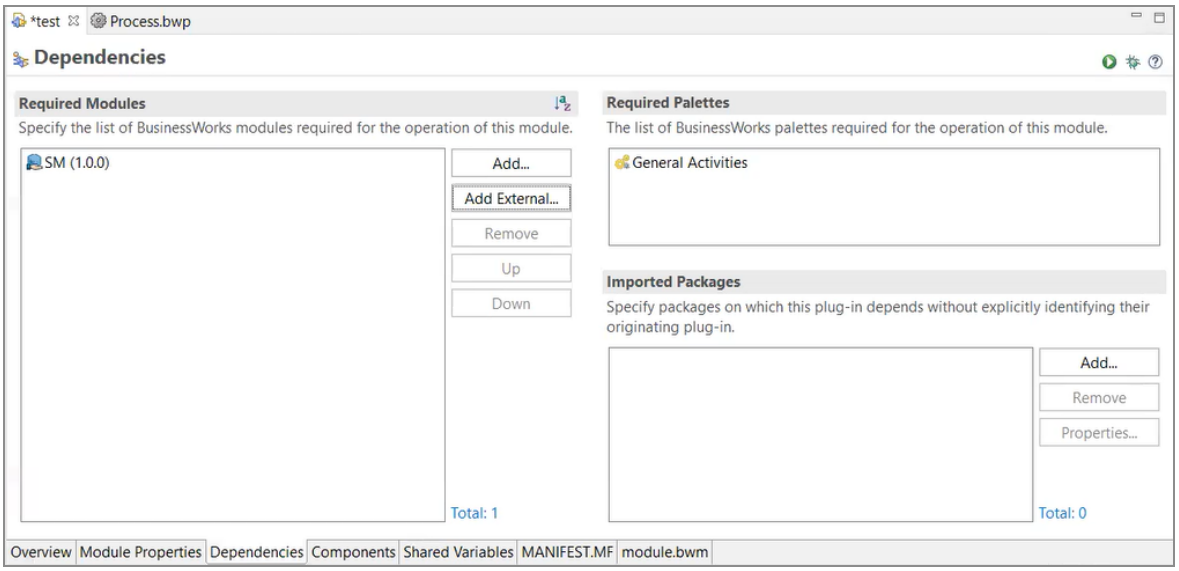
Procedure

1. In an application module, go to the **Module Descriptors** section.
2. Double-click the **Dependencies** option.



3. In the Dependencies pane, click **Add External**.
4. Navigate to the archive location of the binary shared module and select the binary shared module archive file to open. Then click **Open**.

The binary shared module is imported and it is listed under the Required Modules tab.



5. Click **Save**.

   The binary shared module is displayed on the **Project Explorer** tab.

Referencing Shared Modules

You can keep the archive files of the shared modules at an external or a shared location where other users can access the archive files. You can import the archive files in TIBCO Business Studio™ for BusinessWorks™.

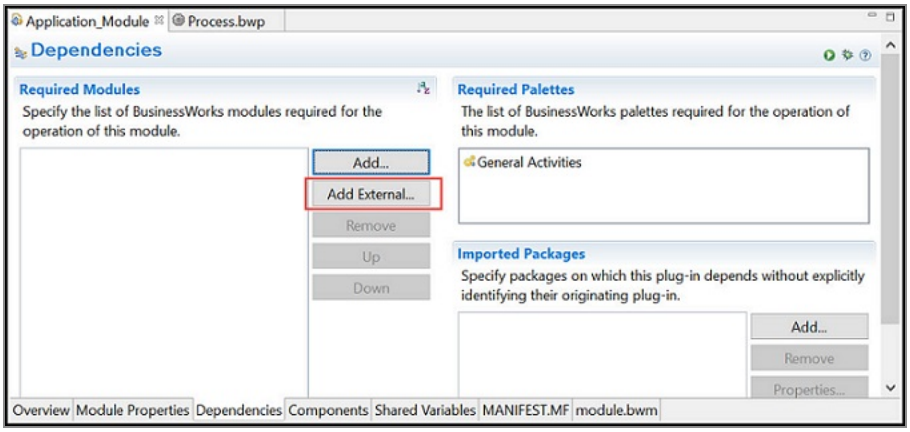
Using Dependencies Tab

TIBCO recommends to import an external shared module using the **Dependencies** tab of an application module.

Procedure

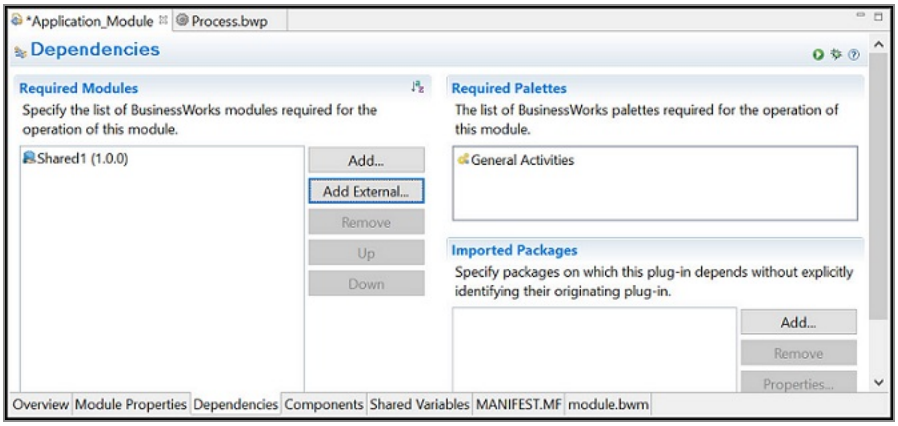
1. In an application module, navigate to **Module Descriptors** section.
2. Double-click on the **Dependencies** option.

   The Dependencies pane is displayed.



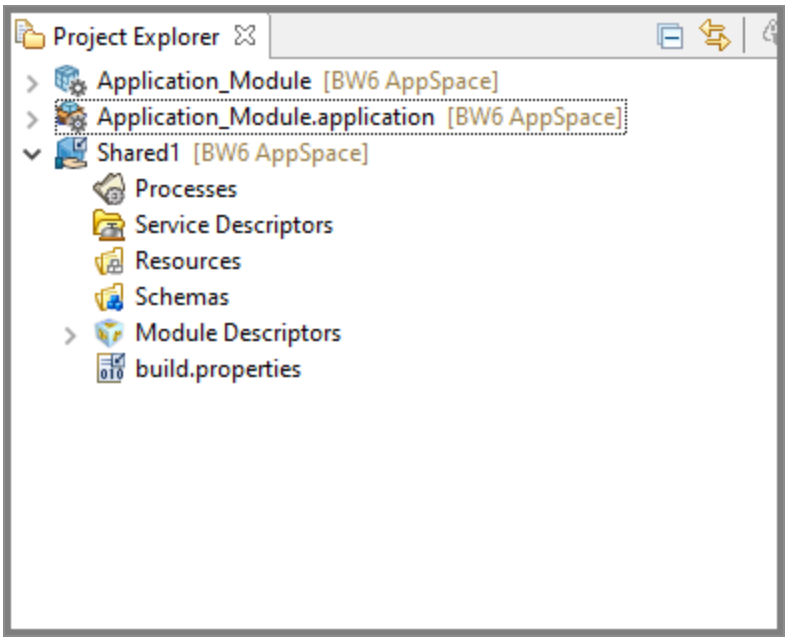
3. Click the **Add External** button.
4. Navigate to the archive location of shared module and select a shared module archive file to open.
5. Click **Open**.

   A shared module is imported and it is listed in the Required Modules tab.



6. Click **Save**.

   The shared module is displayed in the **Project Explorer** view.



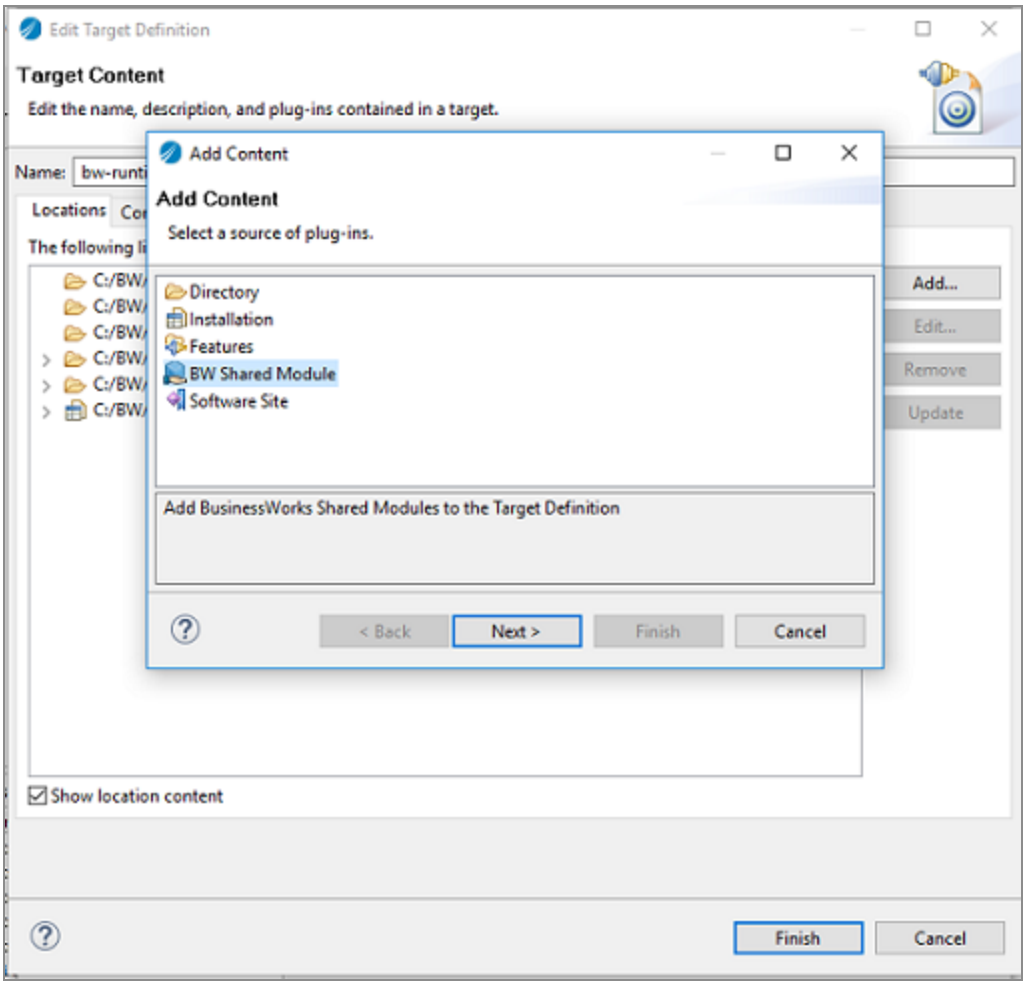
> **Note:** Processes, module properties, shared resources, and other artifacts in a shared module are read-only.
>
> You can not add dependencies in the external shared module.

Using Target Platform

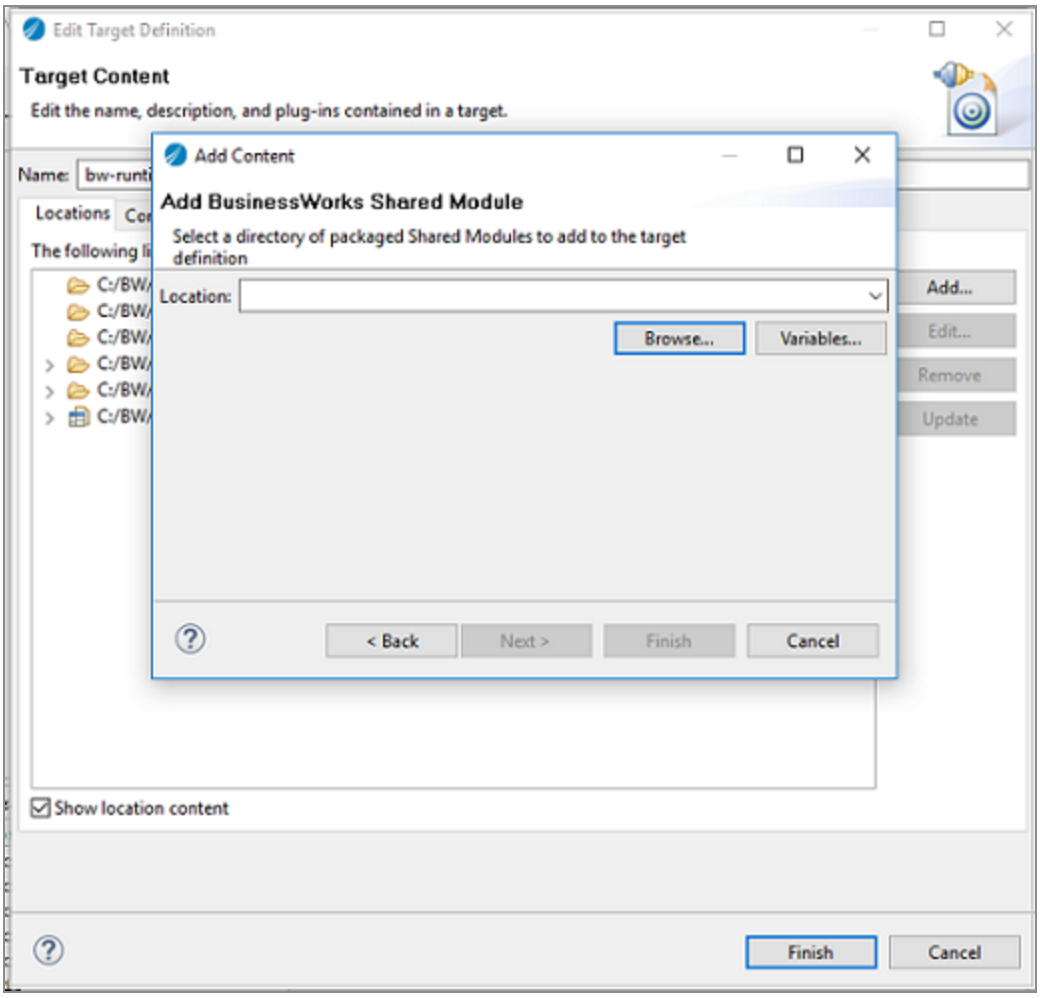
Optionally to reference an external shared module use the **Target Platform** option.

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, click **Add**.
3. On the Add Content dialog, select the option **BW Shared Module** and click **Next**.



4. On the Add BusinessWorks Shared Module dialog, 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.

Before you begin

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 the name `Process`. There are multiple ways to open 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	The name of the module and the Process special folder where the process is located. You can add multiple folders in **Project Explorer** and then update this field to select the new folder. For example: bw.test.app/Processes.
Package	The 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: bw.test.app.main.
Process Name	Name of the new process. For example: MainProcess
Modifiers	Designate whether the process is public or private. You can change this preference later.
Patterns	Choose the pattern **Empty Process** when creating a process. **Note:** To create a subprocess, choose the pattern **Subprocess**. For more information, see Creating Sub-Processes on details for creating a subprocess.

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 Configuring a Process
- Add activities to the process as described in Adding Activities

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 is 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.

Field	Description
Process Name	Name of the subprocess.
Modifiers	Designate whether the process is public or private. You can change this option later.
Interface Mechanism	Select either **Direct** or **Service** • **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: ◦ **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 is 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 is public or private. You can change this option later.

Field	Description
Patterns	Select **Standard Patterns > Process** and then select one of the following options: • Direct Subprocess • Service Subprocess See the preview of the selected subprocess.

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.

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.

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 opens a **Rename Package** dialog. Change the package name using the **Rename Package** dialog.
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. Change the process name using the **Rename Process** dialog.

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: • 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.
Mode	Mode defines whether the process depends on the engine to maintain its

Property Name	Description
	state: • **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	Activation mode for a process defines the way in which processes are activated at runtime. • **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 is activated and any check pointed data can be recovered. **Note:** This feature requires the engine persistence mode to be set to group and the database and group provider to be configured.
Activity Error Variable	By default, this checkbox 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.

Property Name	Description
	Clear the checkbox 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.
Namespace Registry	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. 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. 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. 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. 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 **Window > Preferences > BusinessWorks > Namespace Registry**.

Process Properties

Add or delete process properties variables in the following format:

Property Name	Description
Property Name	Provide a property name.
Data Type	Supported Data Types are:

Property Name	Description
	• Boolean • DateTime • Integer • Long • Password • String • Data Format • FTP Resource • HTTP Client • HTTP Connector • Identity Provider Resource • JDBC Connection • JMS Connection • JavaGlobalInstanceResource • KeyStoreprovider Resource • LDAP Authentication • Notify Configuration • Proxy Configuration • Rendezvous Transport • SMTP Resource • SSL Client • Subject Provider Resource • TCP Resource • ThreadPool Resource Data types may vary depending on the additional plug-ins installed.
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. You can create simple or complex type of variables.

The ErrorVariable type changes depending on the number of transitions to the **End** activity. Whenever there are multiple transitions to the **End** activity, a new variable with an optional error report type is created and that variable, *_errorOptional*, takes precedence over the normal error variable, *_error*.

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 resolves a specific Element. If you choose the Element, the appropriate namespace import is then added to make sure that the element resolves.

Configuring a Process

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.

Before you begin

Open a process in TIBCO Business Studio for BusinessWorks if it is not already open and go to the Properties view.

Procedure

1. Configure the general properties for a process by selecting the **General** tab on the Properties view.

Property Name	Description
Package	Displays the name of the package containing the package. This field is not editable.
Name	Name of the process. This field is not editable.
Target Namespace	Target namespace for the process. You can specify a different target namespace for the process.
Modifiers	Modifiers define the visibility of the process outside its package: • **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.
Namespace Registry	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. 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.

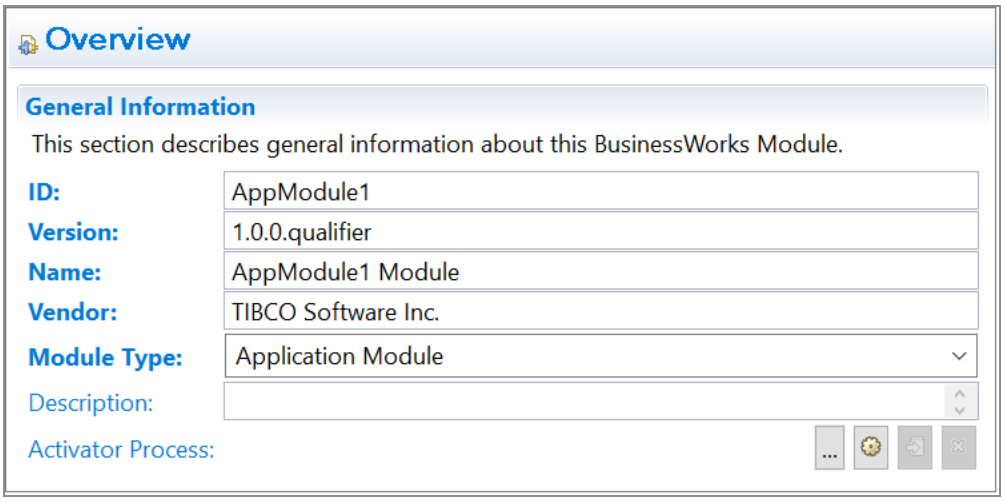
Property Name	Description
	Note: If you have defined both, Process level and Module level configurations for a namespace, the Process level registry takes precedence over the Module namespace registry. When namespace registry is applied, prefixes in the activity input bindings are updated using the prefixes defined in the namespace registry where the namespaces are referred to. 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**.

Creating an Activator Process

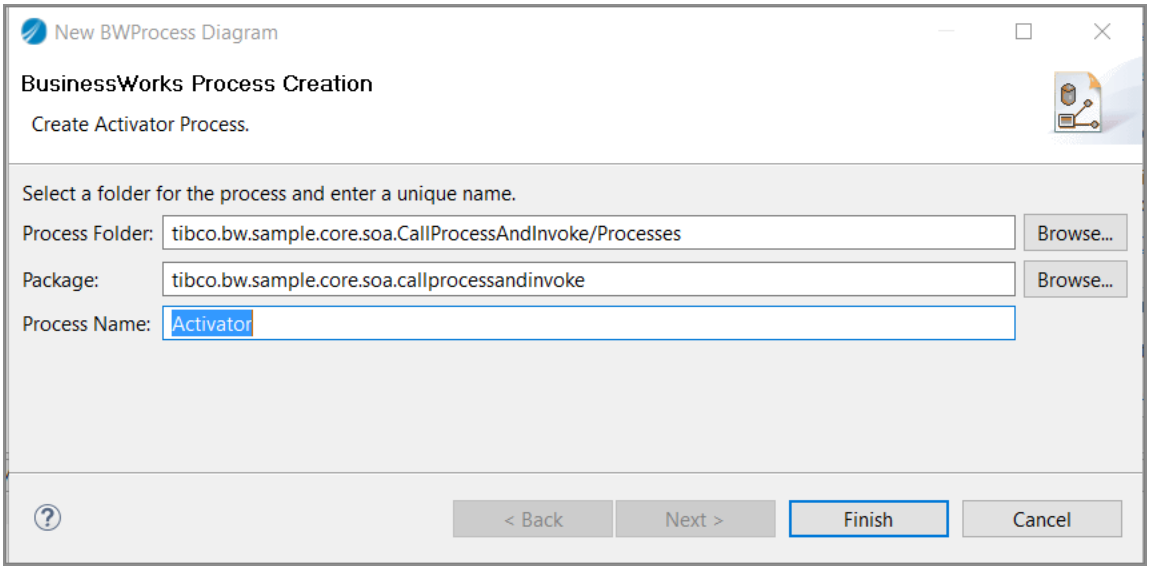
An activator process consists of two service operations, On StartUp and On ShutDown, which can be used to perform tasks when an application starts or after an application stops.
An application module can contain only one activator process. The following steps describe how to create an activator process 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.

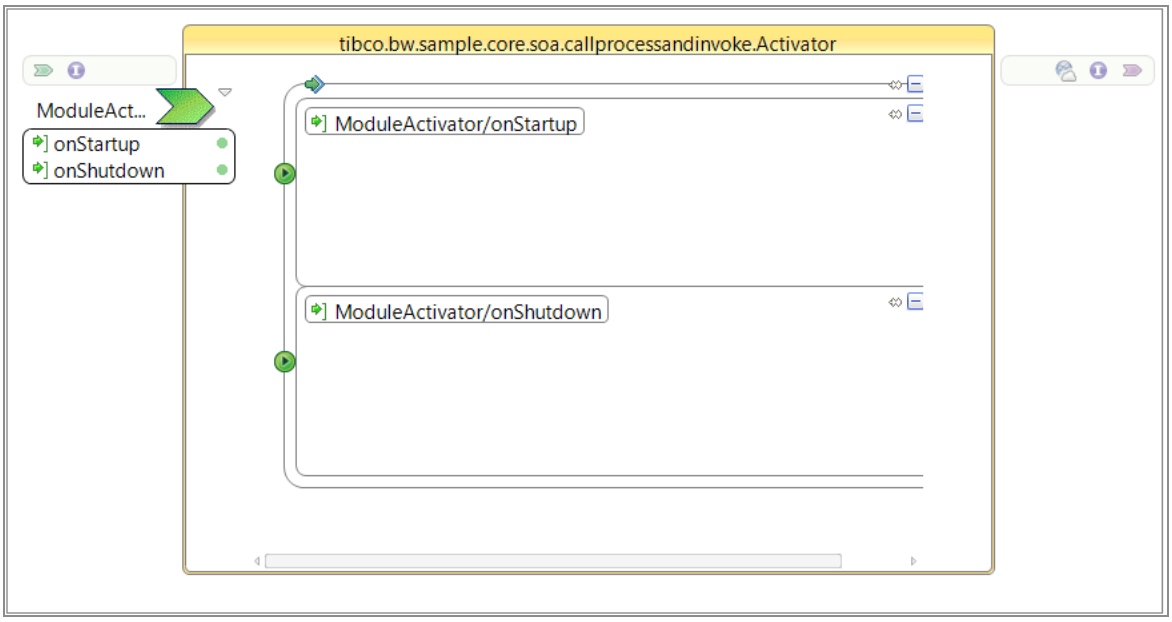


2. Review the fields in the **Create Activator Process** wizard and click **Finish** to create an activator process. The **New BWProcess Diagram** wizard is displayed as follows.



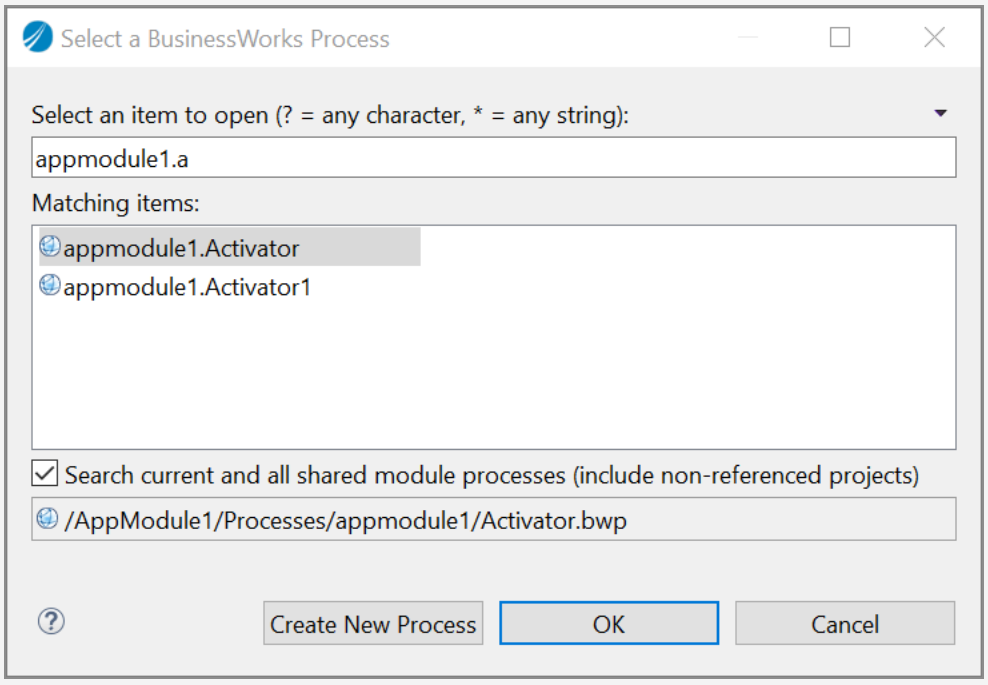
Result

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


tibco.bw.sample.core.soa.callprocessandinvoke.Activator
ModuleAct...
onStartup
onShutdown
ModuleActivator/onStartup
ModuleActivator/onShutdown

Note:

- 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**, , 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.

Note: **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**,,**Fault**, and so on.

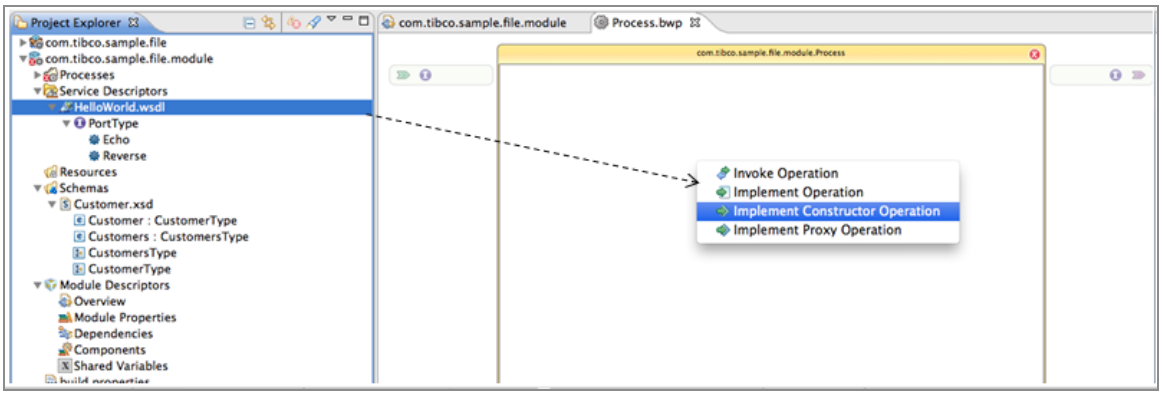
Adding Activities From the Project Explorer

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 `Echo.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. For more information, see Working with Transitions.

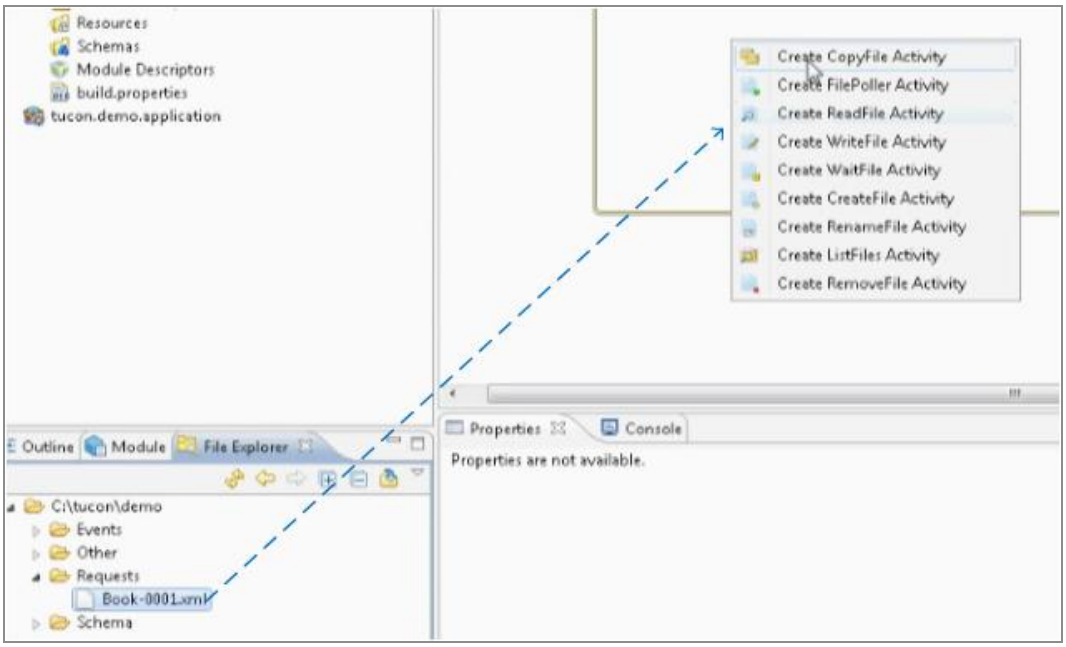
Adding Activities From the File Explorer

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 it 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 is available in the diagram. You can change this label 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.

Note: 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.

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 information, 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 information, 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 information, 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 information, 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

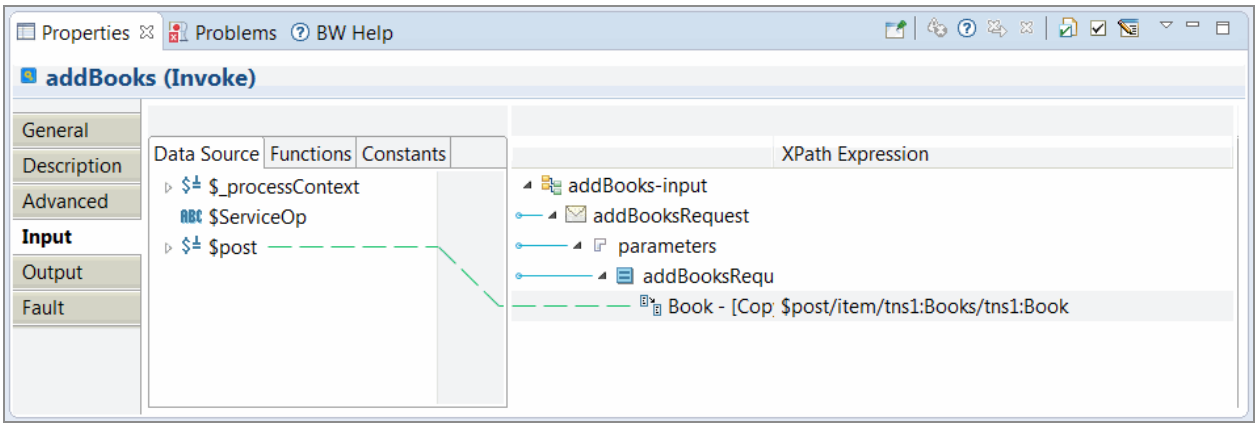
The **Input** tab is used to enter the data for an activity and the **Output** tab displays the output schema of an activity.

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



1. Click on the desired item in the available schema in the Data Source panel such as "name". Drag the item to the desired item in the Activity Input panel such as "Message".
2. To type in a constant or expression, click on the schema item ("Message") 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 **Statement** menu item contains several sub-items that are useful shortcuts for creating XSLT statements.

Option	Description
Surround with Choice	Choice statements enable you to conditionally specify the mapping based on an expression. Choice statements consist of a When clause to specify the condition you want to test, the mapping you want to perform if the condition is true, and an Otherwise clause to contain a mapping to perform if no conditions evaluate to true. An example of using a Choice statement is when more than one fault message is handled by the same Catch Fault task.
Surround with If	If statements enable you to specify a condition, and if the condition is met, then the specified mapping is output. When you choose this option, an If statement appears before the selected

Option	Description
	element, and you must place an XPath expression in the If statement that evaluates to a boolean. If the expression evaluates to true, the specified mapping is performed. If the expression evaluates to false, the mapping is not performed and no value is set for the item. Do not place an If statement around schema data components that are marked as required.
Surround with For-Each	For Each statements enable you to execute one or more statements once for each data element in a list. When you choose this option, a For Each statement appears before the selected data component, and you must place an XPath expression in the For Each statement that evaluates to a list of zero or more items. This is useful when you want to manipulate sequences or repeating elements.
Surround with For-Each-Group	A shortcut for moving the current element into a For-Each-Group statement and adding a Group-By grouping statement. The `current-group()` is not provided on the 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.

Instead of specifying a constant or an expression for the schema item, you can first configure the sequence in which this message appears 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:

a. Schema type definition or creating a new type definition.

b. Number of Minimum Occurs (default is 1).

c. Number of Maximum Occurs (1 or unbounded).

d. Number of references to this resource (generated, in this case it is 0).

e. 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.

f. 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:

a. The schema type definition or a new type definition (default is anyType)

b. Number of Minimum Occurs (default is 1).

c. Number of Maximum Occurs (1 or unbounded).

d. Number of references to this resource (generated, in this case it is 0).

e. 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.

f. 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:

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. 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.)

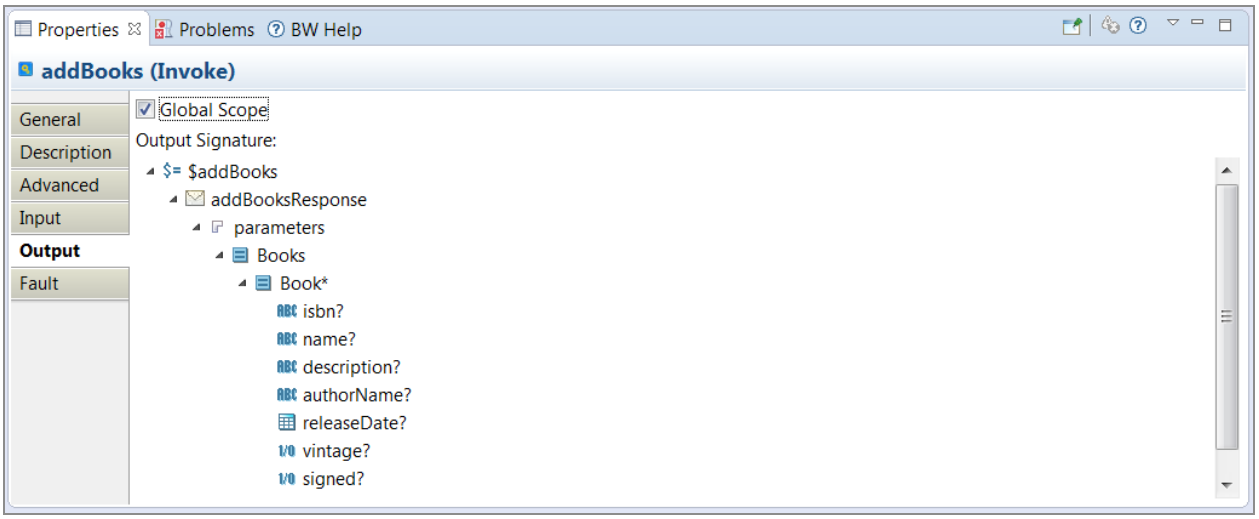
d. The remaining icons are **Go To** , **Accept Changes** , and **Delete** , which invoke the general editing tools.

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.

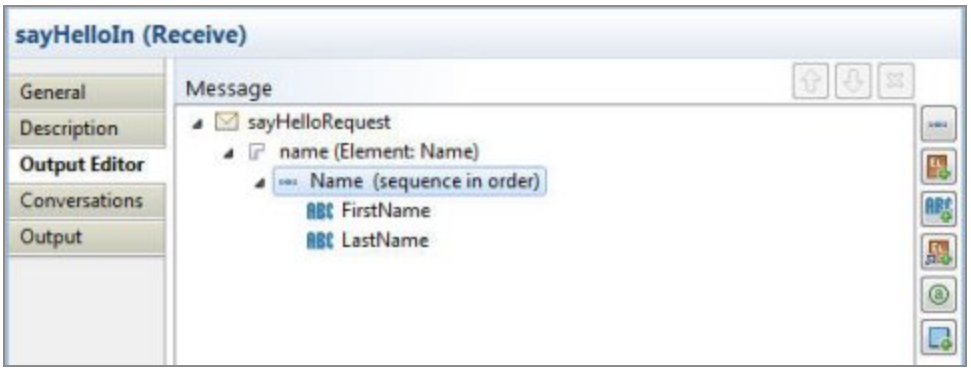
Output Tab



Configuring the Output Editor Tab

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

Output Editor Tab



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

Adding Custom Icons

You can set a custom icon of size 48 pixel by 48 pixel to activities in a process. The icon image file must be of type `.png`, `.jpg`, or `.jpeg` and the file size must be less than 512 KB. The following steps are shown for the **Call Process** activity. Perform the same steps to set a custom icon for the **Java Invoke** activity.

Procedure

1. Select the **Call Process** activity in a process.
2. Provide a **Custom Icon** path on the **General** tab of the activity.

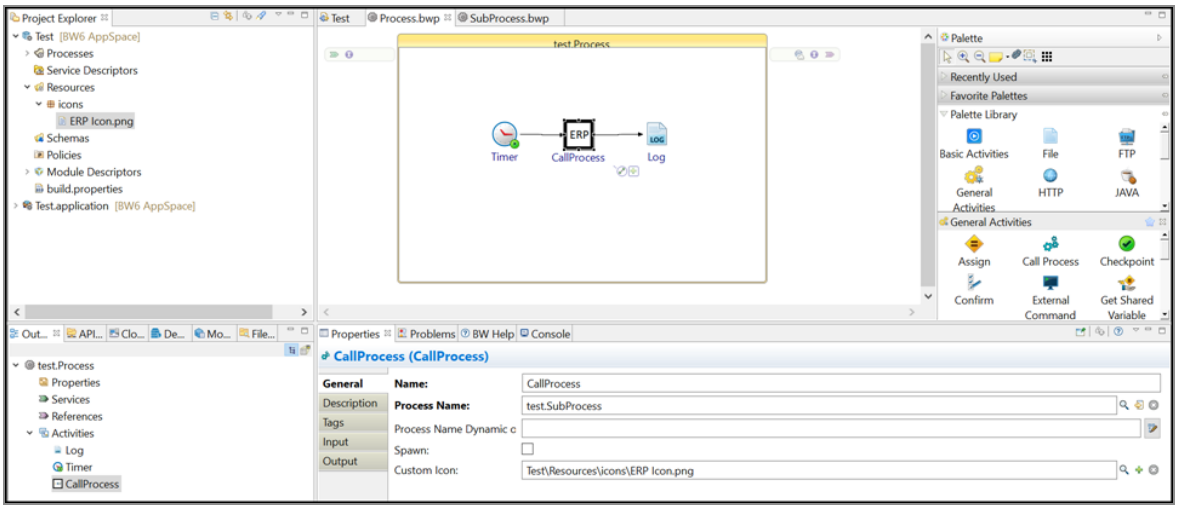
   You can either provide a custom path from your file system or path of the icon file present within a workspace.

   For more information, see "Call Process" and "Java Invoke" in *TIBCO BusinessWorks Container Edition Bindings and Palettes Reference*.
3. Save the process.

   The custom icon is set to the **Call Process** activity. The custom icon path is also changed to the path as per your process module location in the following form:

   ```
   <Module Name>\Resources\icons\<Icon File Name>
   ```

   The image is listed in the **Project Explorer** view. The new icon is also reflected in the **Outline** view.



Updating Custom Icons

The following steps show how to update a custom icon for the **Call Process** activity. Perform the same steps to update custom icon for the **Java Invoke** activity.

Procedure

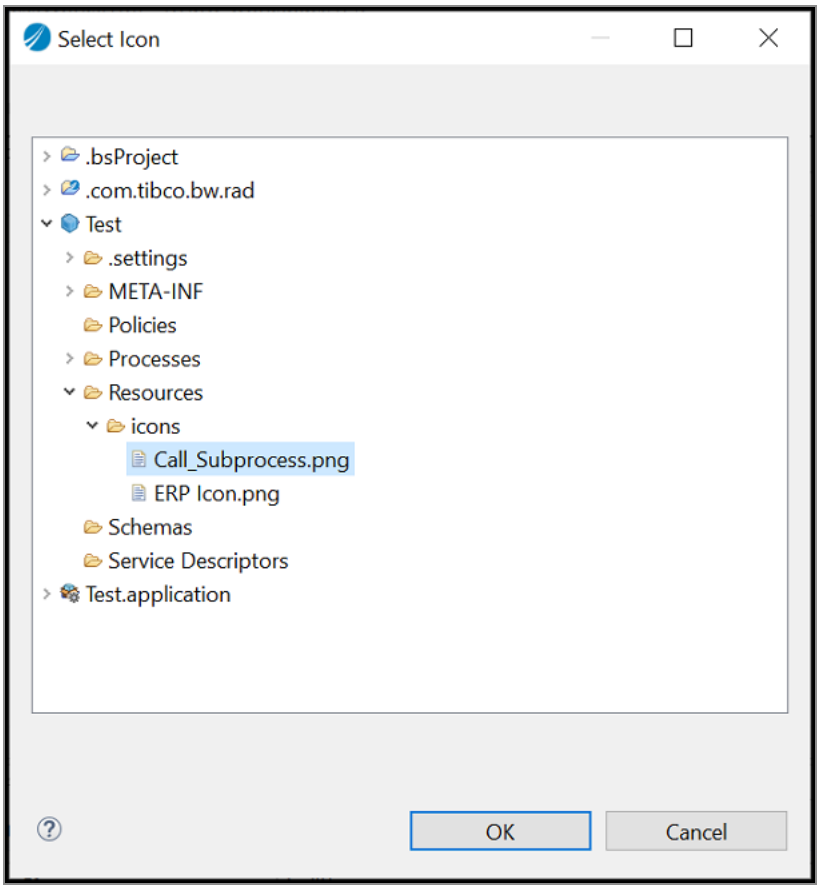
1. Select the **Call Process** activity.

2. Provide the **Custom Icon** path on the **General** tab of the activity.

   For more information, see Adding Custom Icons.

3. Optionally, to search for the icon file in your workspace, select the **Choose a custom icon** button .

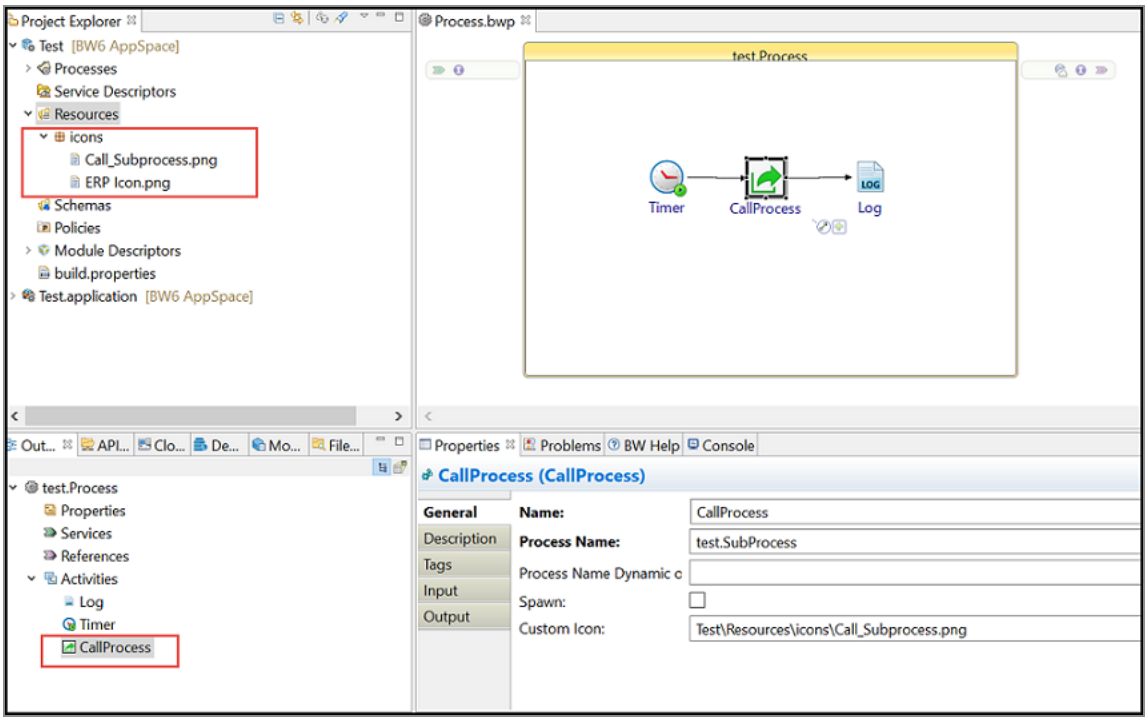
   The Select Icon dialog is displayed.



4. Select an icon image from any of the modules in the workspace and click **Ok**.

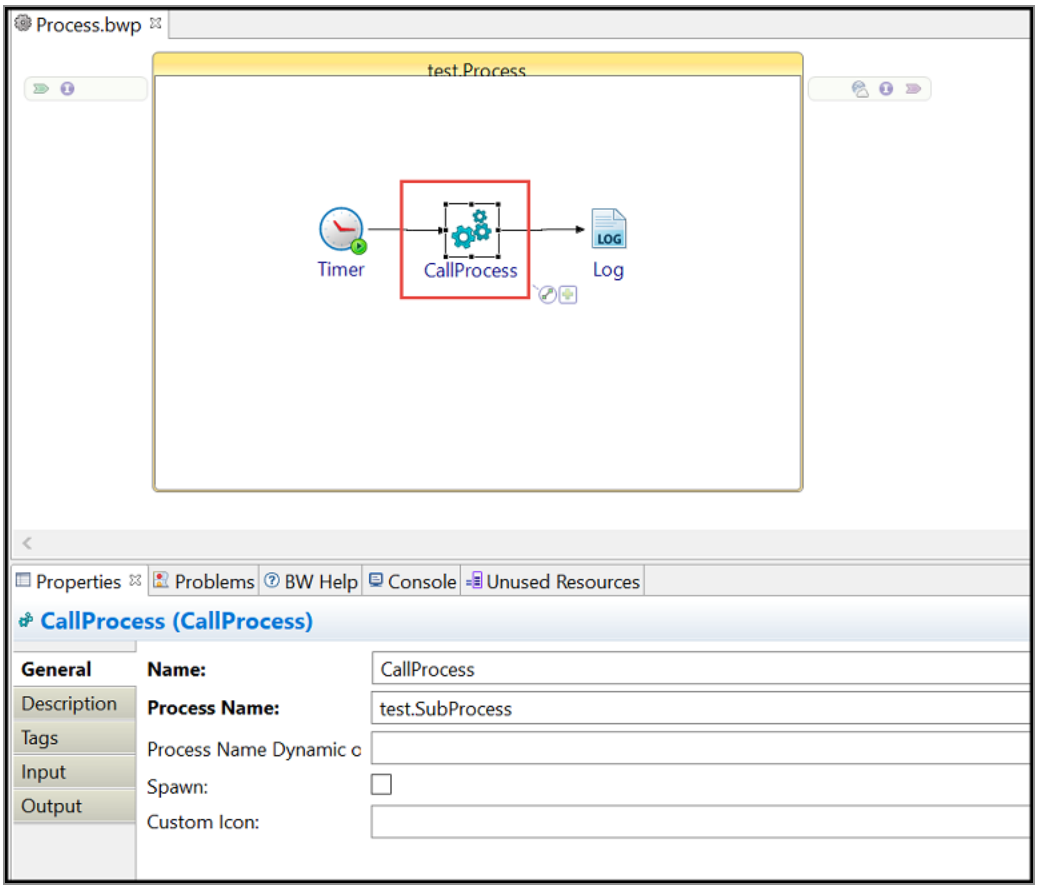
   The new icon is set to the activity in a process as well as in the **Outline** view. The icon is added at

```
<Module>\Resources\Icons
```



Note: Whenever you set a new custom icon, the previous icon persists in your workspace.

5. If you select the **Clear Value** button or delete the **Custom Icon** field value, then the default icon provided by TIBCO for the **Call Process** activity is set. The icon is also updated in the **Outline** view.



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.

> **Note:** If you enter a remote WSDL URL, and the WSDL contains dependencies, these dependencies are listed in the **Dependencies** section of the Import WSDL from URL wizard.

3. Optional. Clear the **Update import location attribute and include location attribute to reference WSDL and XSD files imported locally** checkbox if you do not the import location to be automatically updated with the relative locations of corresponding and already imported WSDL and XSD files.

Properties

Properties are used to define configuration. Depending on where and how they are defined and qualified, properties can be classified into application properties, module properties, shared module properties, and process properties. The values for all three kinds of properties can be of one of the six primitive types (Boolean, Integer, DateTime, Long, Password, or String) or one of the available default shared resource types. These values are static and cannot be changed once an application has started execution. These values can only be changed at design time or deployment time.

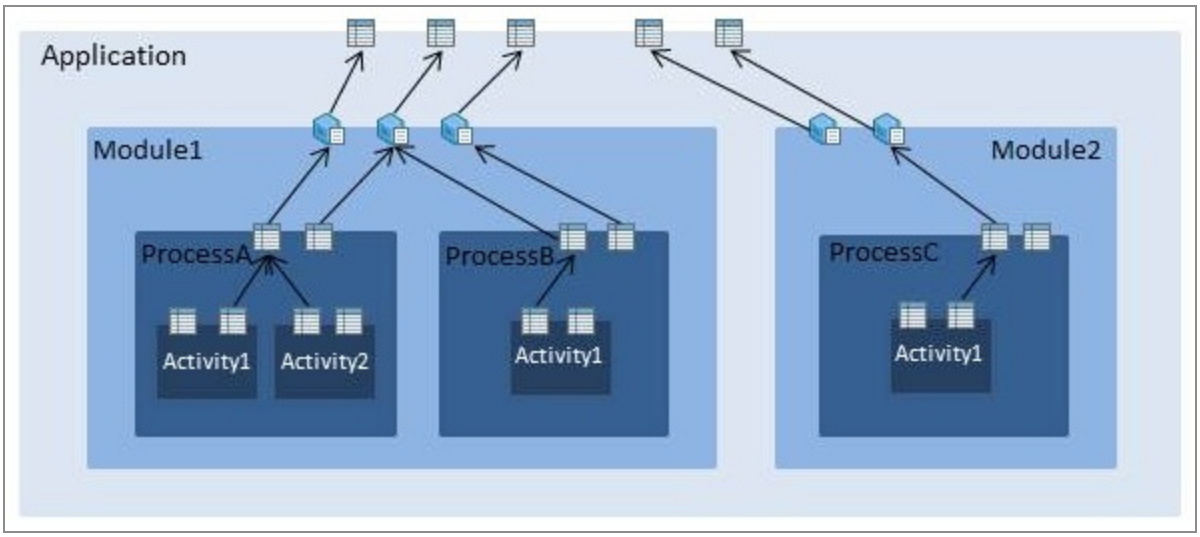
The three levels of properties are hierarchical: application properties are in the outermost scope, followed by module properties, followed by process properties.

Properties defined in the inner layer can reference a property defined at its parent layer. For example, a process property can reference a module property instead of providing a literal value. Similarly, a module property value can be defined by literal values or source from its parent scope application property.

Any process property or module property that you define is available both in the activity configuration page and is also available to use as an input to an activity (from the **Data Source** tab of the **Input** tab for the activity).

The following diagram illustrates the relationship between the different types of properties:

Relationship Between Properties



Features of Process, Module, Shared Module, and Application Properties

Property	Scope/Visibility	Values	Additional Information
Process Properties	Visible within a process.	Literal, module property reference, or a shared resource reference.	Literal values cannot be modified at the module or application level.
Module Properties	• Visible within the module.	• Literal or a shared resource reference.	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.
Shared Module Properties	• 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. • Private module property values cannot be edited from the Admin UI.	• 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	• Only available	• Literal.	• Overrides module

Property	Scope/Visibility	Values	Additional Information
	for an application and visible within the application.	• Profiles can be used to specify a new set of values for the same application.	properties, thus enabling you to use different values for the same module. • Cannot add new properties at application level.

Process Property

The process property is the most basic type of property. Process properties are defined locally for each process, for example a subprocess can have its own process property, processes in shared modules can have their own process properties, and so on.

Important:

- A process property can be assigned to multiple activities within the process. At design time, when you assign a process property to an activity, only the name and the type of the property is associated with the activity, not its value. The value of the property is stored in the process and remains a part of the process. At runtime, the value gets injected into the property. So, if a property is assigned to multiple activities, if you change the value of the property on one activity, keep in mind that the value of the property changes for all activities that use that property.
- Even though a process property is assigned to multiple activities, it can have only one value across all activities. You cannot use the same process property for multiple activities in a process but assign a different values for the property for each activity that it is assigned to.
- Multiple process properties within the same module can get their value from the same Module property.
- Process properties are also visible in the **Outline** view of a process.

Creating a Process Property

Process properties can be created from the **Process Editor** properties page. Follow these steps to create a process property:

Procedure

1. Click the open process in the Process Editor.

   The **Properties** page opens.

2. Click the **Process Properties** tab in the **Properties** view.
3. Click the  icon to add a process property.
4. Click the property name to make it editable and edit the name of the property as desired.
5. Select the data type of the property by clicking **String** and selecting the type from the dropdown menu.
6. Click in the Default Value column and click the  button to specify if you want the property value to be a literal value or a module property.

   If you select a module property, you have the option to create a module property by clicking  or selecting an existing module property from the dropdown menu.

7. Save your project.

Editing a Process Property

A process property that is referenced by an activity can be edited in the **General** tab of the **Properties** view of the activity. Alternatively, it can be edited in the **Process Properties** tab of the Properties view of the process itself.

Important: If the process property you want to edit is assigned to multiple activities within the process, keep in mind that if you change the value of the property in one activity, the value of the property changes for all activities that use that property. When you assign a process property to an activity, only the name and the type of the property is associated with the activity, not its value. The value of the property is stored in the process and remains a part of the process. At runtime, the value gets injected into the property.

To edit the property from the **Properties** view of the activity, click the activity icon in the **Process Editor** to open its properties in the **Properties** view. You can edit the property value by clicking the  button or even select a new property from the drop-down menu.

To edit the process property from the **Properties** view of the process, follow these steps:

Procedure

1. Click on an empty space or on the border of the process in the **Process Editor**.

   The process properties page opens in the **Properties** view.
2. Click the **Process Properties** tab.
3. Click in the Property Name, Data Type, or Default Value column of a property to edit it.

> **Note:** You can create a module property and use the module property as the value for a process property. During runtime, the process property gets its value from that module property.

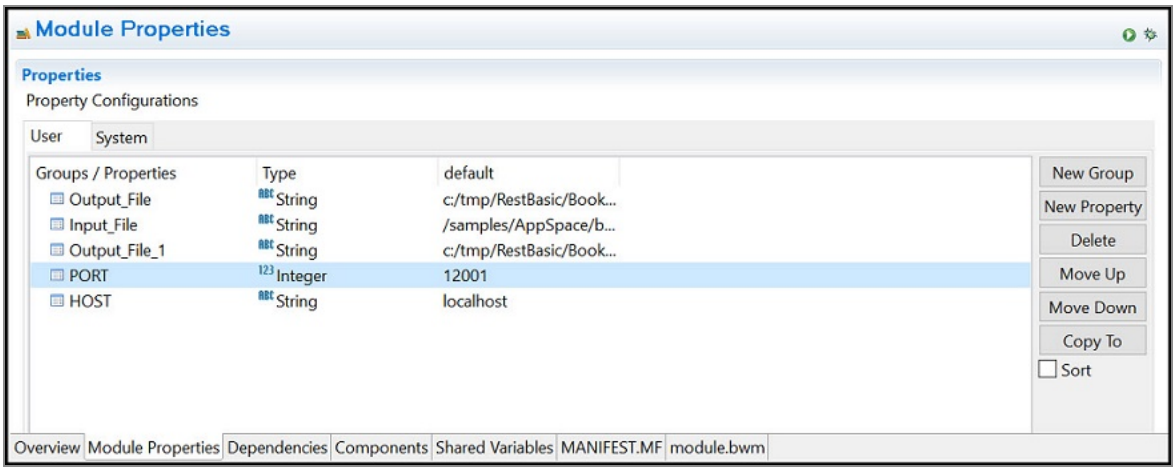
Module Property

Module property provides the default value for a module. Multiple process properties can source their value from a single module property.

Module properties are defined at the module level and can be referenced by various resources that are defined as a part of the module. Their values can also be sourced from application properties at deployment time.

> **Note:** A module property cannot be assigned directly to an activity. It must be assigned to a process property in order to be used in an activity. The process property inherits the value of the module property that is assigned to it. The process property can then be used in the activity. Module properties can be used directly only when configuring shared resources and policy resources.

The Module Properties editor can be used to create and manage module properties. You can add a new property or logical groups which you can use to organize module properties.



Hierarchy of Properties:

The TIBCO BusinessWorks Container Edition module properties can be tokenized which allows in externalizing the properties. These properties can be stored in either config or credential management systems like Consul, AWS Parameter store and so on.

The hierarchy in which the properties are replaced is as follows:

- Config or credential management systems take top priority and any properties coming from external config or credential management systems are used.
- If the tokenized property is not found in external config management systems, the environment variables are used. If the property value is provided as tokenized property then those properties are used.

Working with Module Properties

Module properties can be used to define configurations 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 the module property with the same name as the module property.

Creating a Module Property

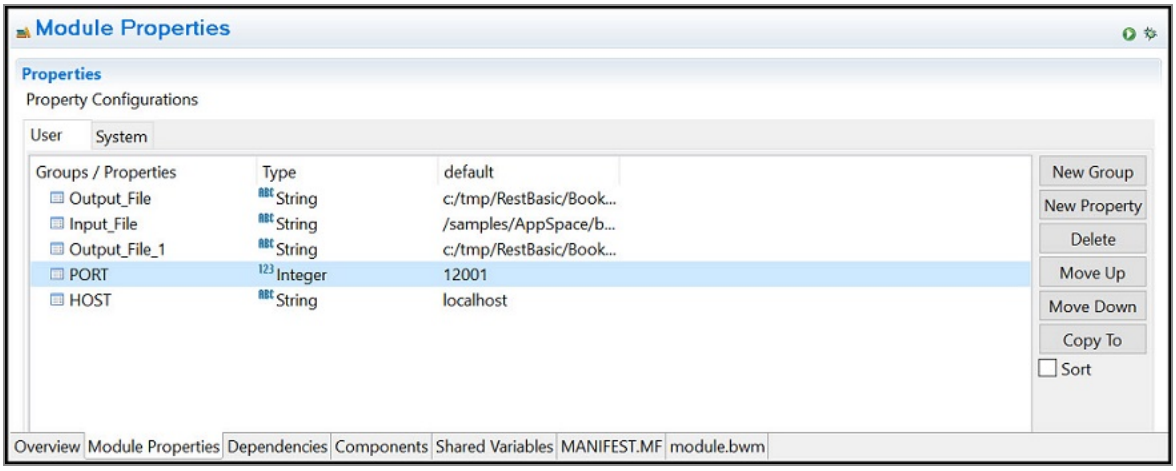
To create a module property, follow these steps:

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.



   The following data types are supported:

   - Boolean
   - DateTime
   - Integer

- Long
- Password
- String
- Data Format
- FTP Resource
- HTTP Client
- HTTP Connector
- Identity Provider Resource
- JDBC Connection
- JMS Connection
- JavaGlobalInstanceResource
- KeystoreProvider Resource
- LDAPAuthentication Resource
- Notify Configuration
- Proxy Configuration
- Rendezvous Transport
- SMTP Resource
- SSL Client
- SSL Server
- SubjectProvider Resource
- TCP Resource
- ThreadPool Resource
- TrustProvider Resource

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 is 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.

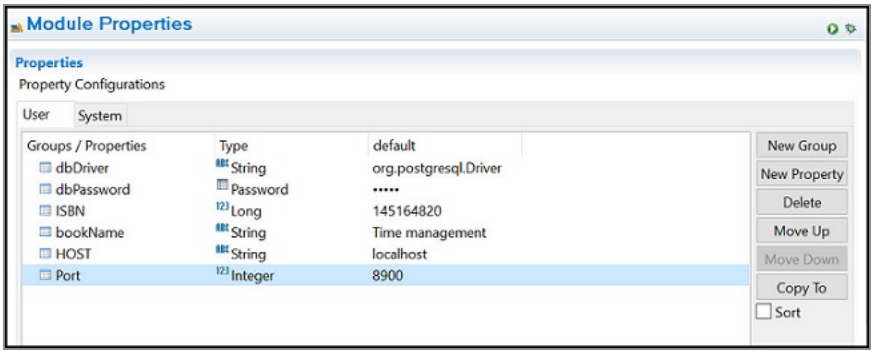
   The value of the property changes color from black to blue. If you edit the module property value in the Properties page accessed from the application, it overrides the original value of the property that was defined in the Module Properties page.

Sorting Module Properties

You can sort the module properties in ascending or descending order. Sorting the module properties helps you to locate them easily.
Follow the steps to sort the module properties.

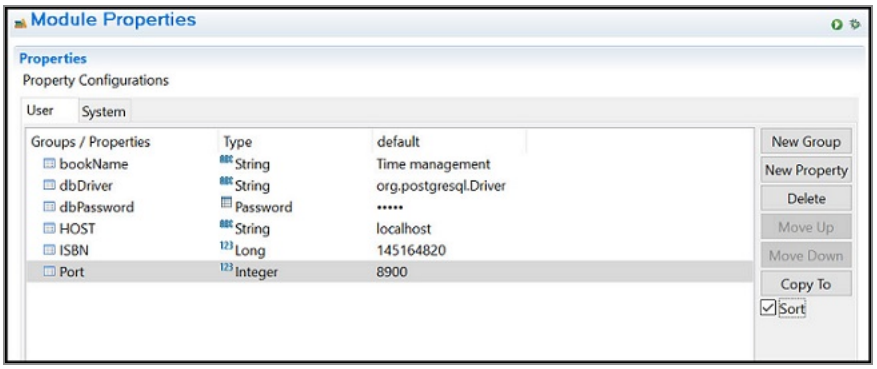
Procedure

1. Open the **Module Properties** view.

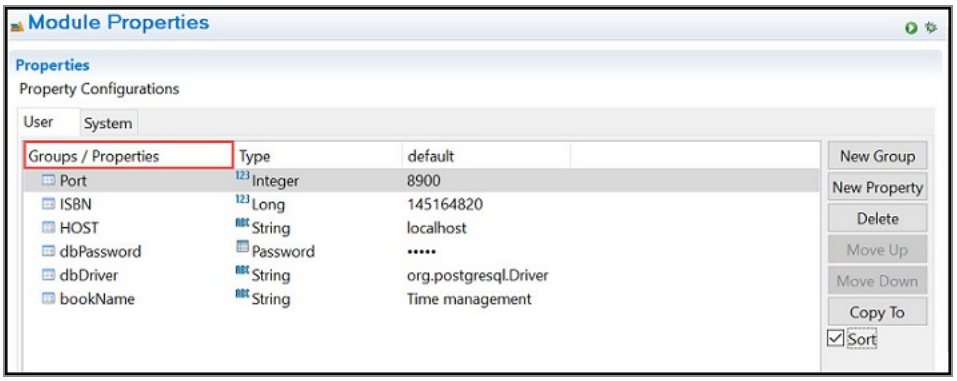


2. Select the **Sort** checkbox.

   By default, the module properties are sorted in the ascending order.



3. To sort the module properties in the descending order, click the **Groups/Properties** column header.



Deleting a Promoted Property

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

Copying module properties from one module to another in TIBCO Business Studio for BusinessWorks

A module property can be copied from one shared or application module to another in TIBCO Business Studio for BusinessWorks.

To copy module properties from one module to another, a new button **Copy To** is added in the Properties view for module properties. By default, this button is disabled.

Before you begin

To copy module properties, make sure the module property is selected in the **Module Properties** tab. The **Copy To** button is available only when the module properties are selected.

To copy a module property from one module to another, do the following steps:

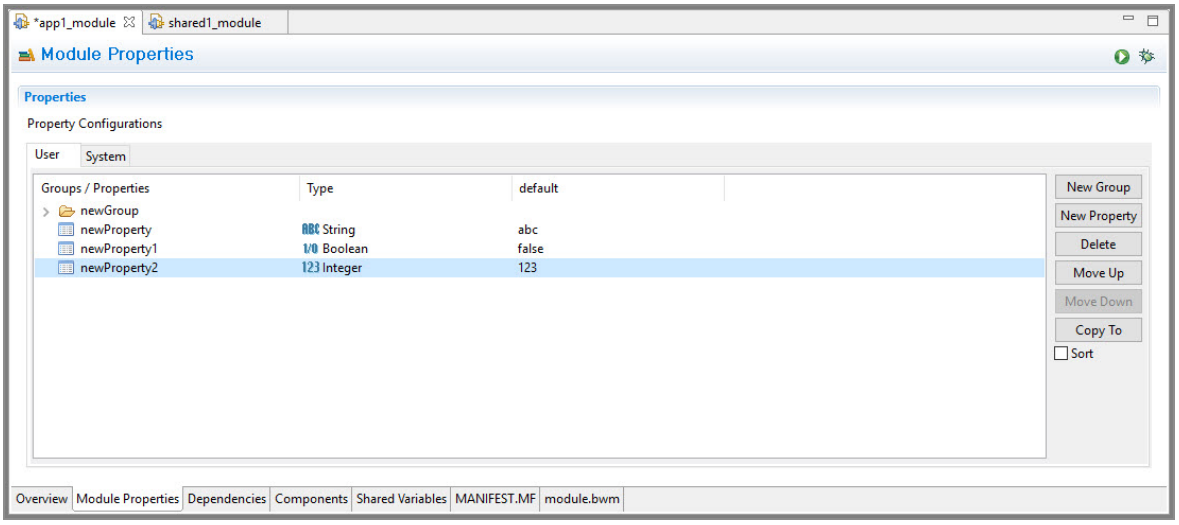
Procedure

1. In the **Project Explorer** view, select **Module Descriptors > Module Properties** of the module from where you want to copy the module properties.

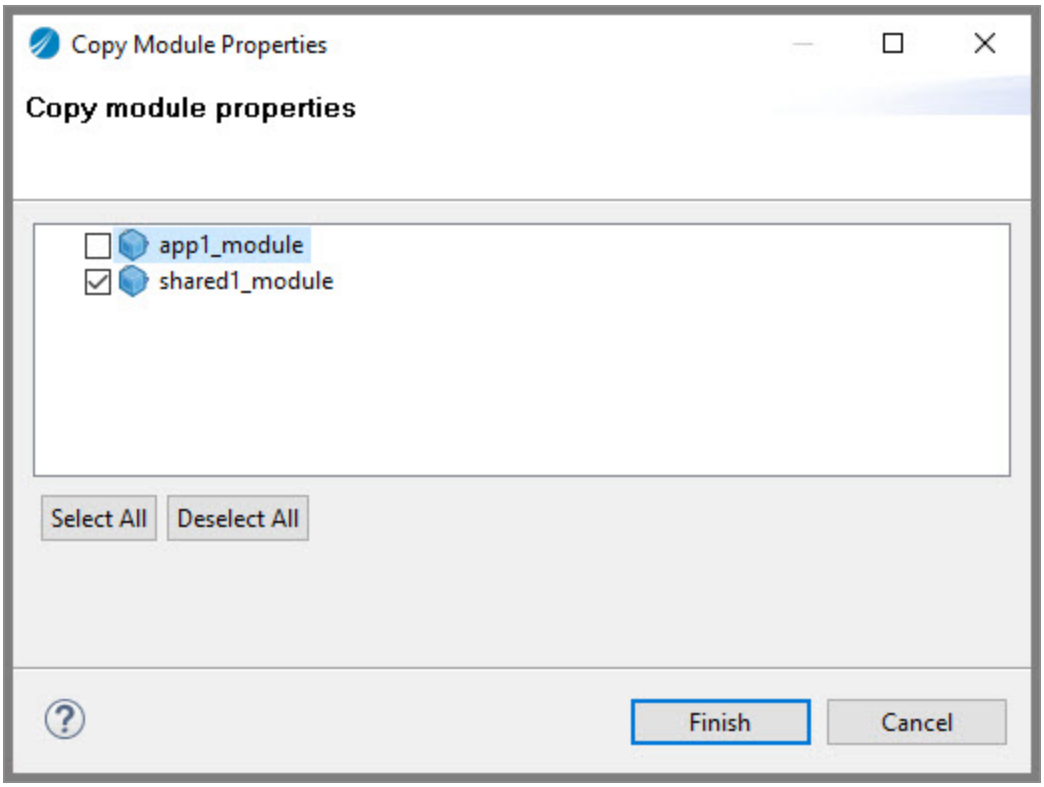
   The **Module Properties** tab opens.

2. In the **Module Properties** tab, select the module properties you want to copy and click on **Copy To** button.

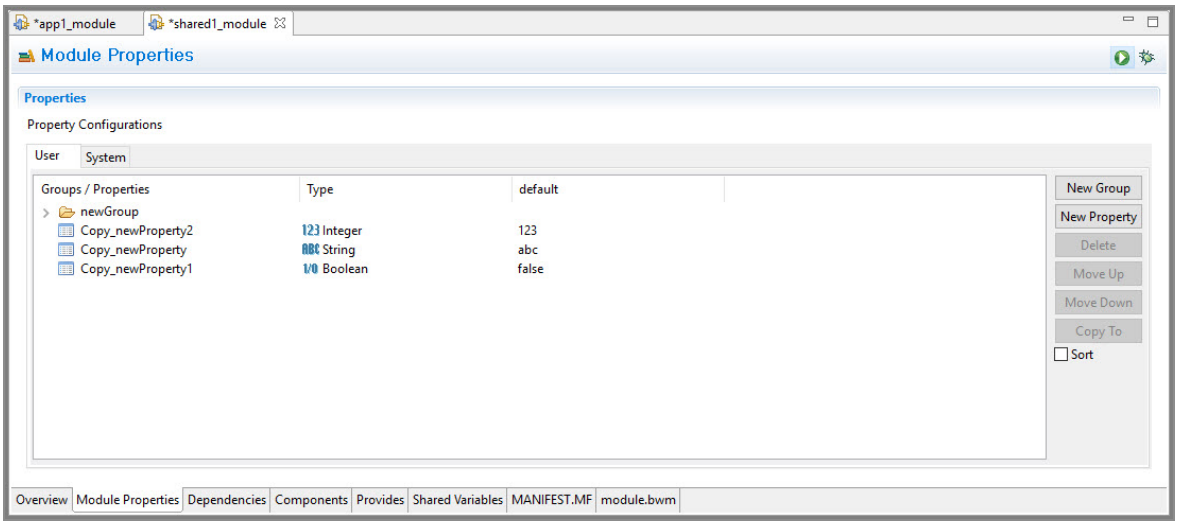
   The Copy Module Properties wizard is displayed.



3. In the Copy Module Properties wizard, select the target project, where you want to paste the module properties and click **Finish**. The **Finish** button is available only when the target project is selected. You can select one or more target projects at the same time.



The module properties are copied to the selected target project with the prefix `Copy_` new.



You can copy the module properties from an application module to a shared module or vice versa. It is also possible to copy the module properties from an application

module to another application module or the same application module and from a shared module to another shared module.

When the module properties are copied to the target project, the module property editors are in one of the following states:

- Closed: The projects are modified silently.
- Already open but not dirty: The changes are visible and not saved. The editor is marked dirty (*).
- Already open and dirty: The changes are visible and not saved. The editor is marked dirty (*).

After the modules properties are copied, they are also synchronized with the application properties and the newly pasted module properties are copied in the application properties as well.

The module properties can be copied even if they are within groups. When the copied module properties which are within groups, is pasted the same hierarchy path is also copied in the target folder. The new module properties have unique names that are generated as per the scope they are in.

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:

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 changes color to blue and enclosed in **%%** indicating that the property has been promoted. The property is 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 does 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.

   > **Note:** Whenever you promote any properties of a particular profile, either set that profile as **default** or promote the same property in the default profile so that you can modify it at runtime.

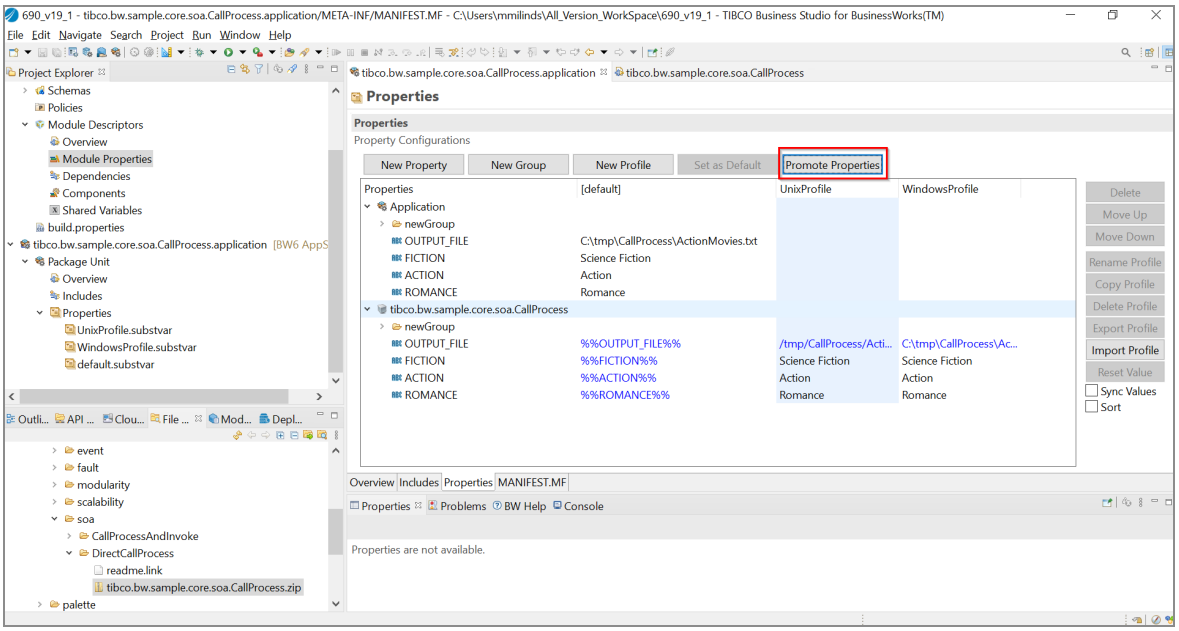
Promoting Module Properties in a Batch

You can use this feature to promote module properties in a batch.

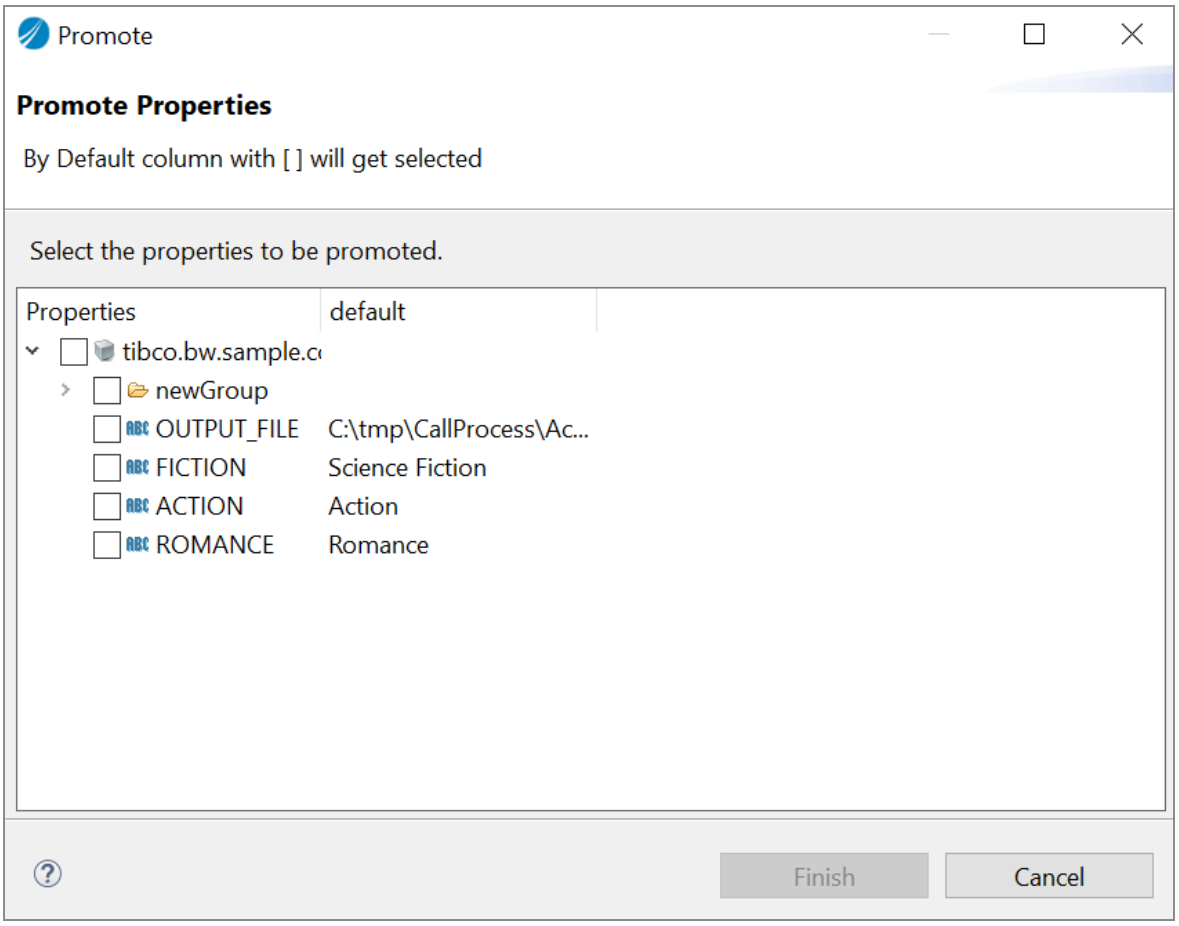
Follow these steps to promote application properties:

Procedure

1. In the Project Explorer, fully expand `<application_name>.application`.
2. From the **Package Unit** menu, double-click **Properties** to open the application properties in the right pane.
3. Select the profile to be promoted and click **Promote Properties**.



This opens a dialog with the properties from the selected profile that are yet to be promoted. By default, the property column with [ ] is selected.

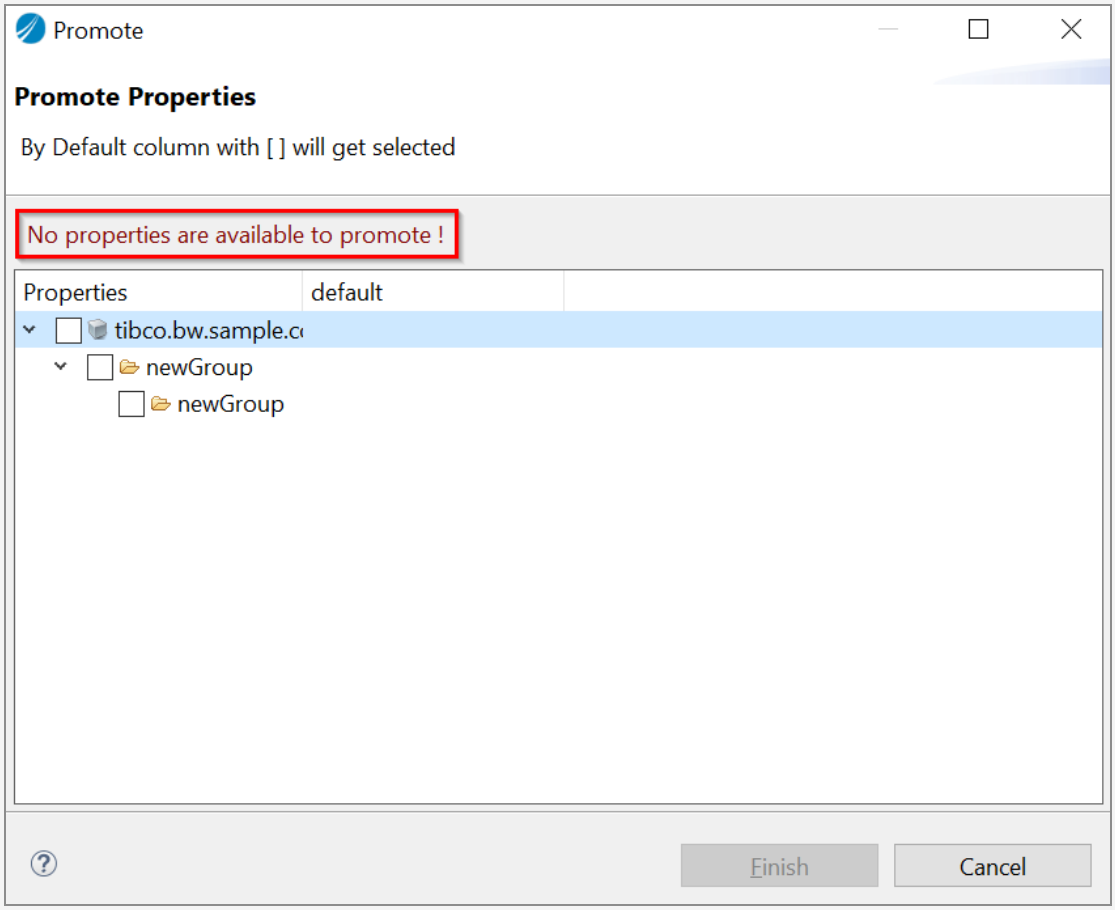


4. Select the parent checkbox to select all the properties to be promoted. Alternatively, you can select each property manually.
5. Click **Finish**.

   The selected application properties are promoted.

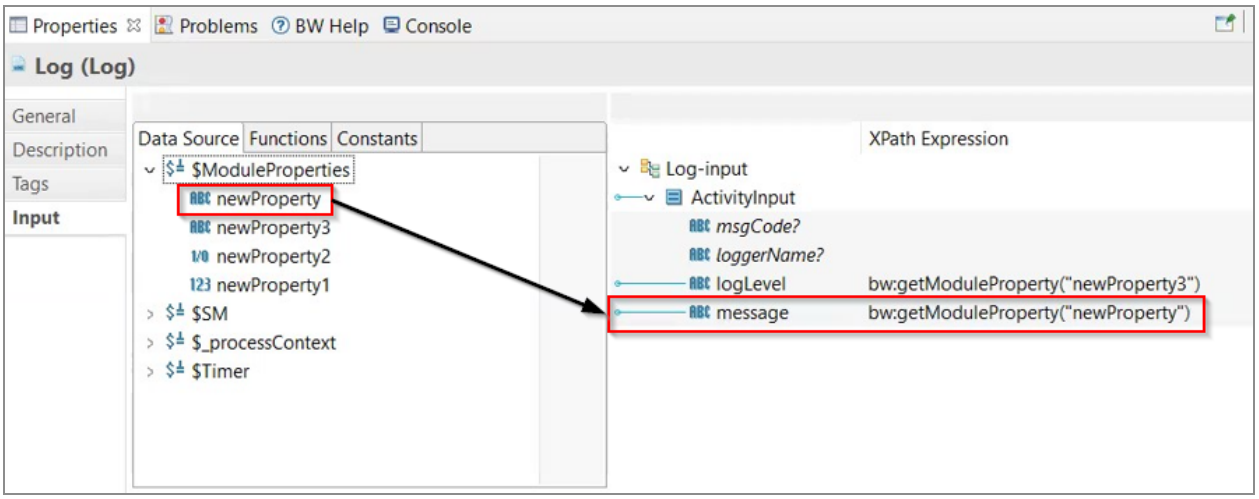
Note: If a property is already promoted, the following error message is displayed:

```
No properties are available to promote!
```



Mapping Module Properties to an Activity Input

The module properties can be mapped to an activity input without creating a process property. All module properties as well as the module properties from the Shared Module are available under the **Data Source** tab of an activity input pane. Select the property that is to be mapped, and drag and drop to an activity input element.



Note: When you drag and drop the module property, `bw:getModuleProperty()` gets autowrapped.

Application Properties

Application properties have the largest scope of all properties. Application properties are useful when you want to share the same value for an existing module property across multiple processes in an application or between an application module and a shared module. You can either promote an existing module property to the application level or create a new application property.

Important: If you choose to promote an existing module property to the application level, keep in mind that all activities using that property get the same value.

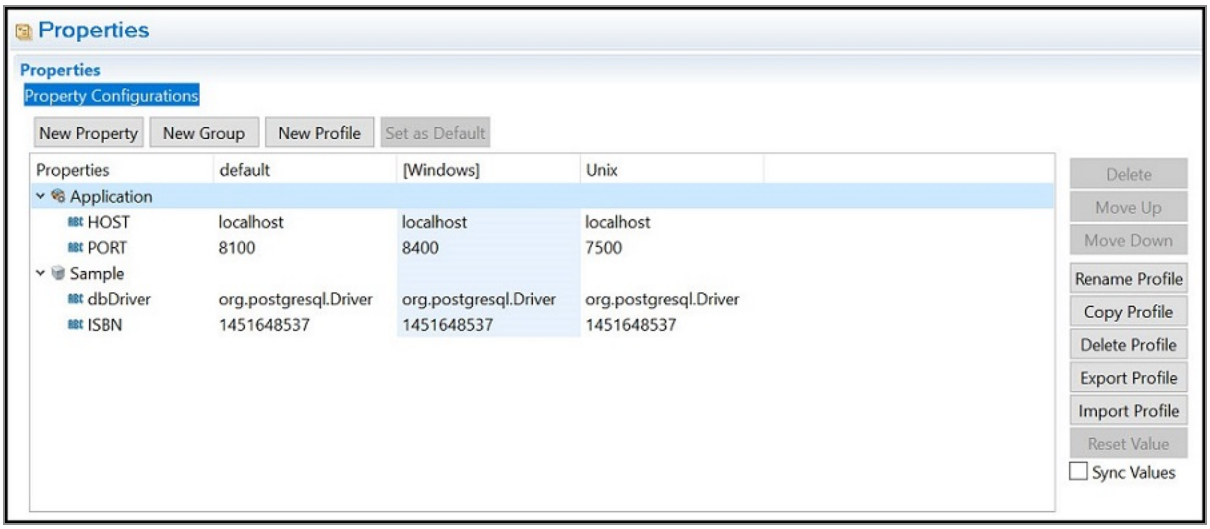
Refer to Promoting Module Properties for Visibility at the Application Level for steps on how to promote a module property and Creating an Application Property for steps on how to create a new application property.

You can use application profiles to store multiple values and switch profiles at deployment time. Refer to Creating an Application with Multiple Profiles for details on creating profiles.

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.



Supported Datatypes

The following datatypes are supported for an application property:

- Boolean
- DateTime
- Integer
- Long
- Password
- String

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 *TIBCO BusinessWorks Container Edition Samples*.
6. Save your application.

   For information on creating profiles, see Creating an Application with Multiple Profiles.

   When a property value is the 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** checkbox.

Using an Application Property

If you create a new application property, be sure to map a module property to it. Application properties cannot be assigned directly to an activity or used directly to configure a resource. To use an application property, you need to map a module property to the application property. A module property that is mapped to an application property takes its value from the application property to which it is mapped.
As a good housekeeping practice, do not create new application properties without mapping one or more module properties to it.

Follow these steps to map an existing module property to an application property:

Procedure

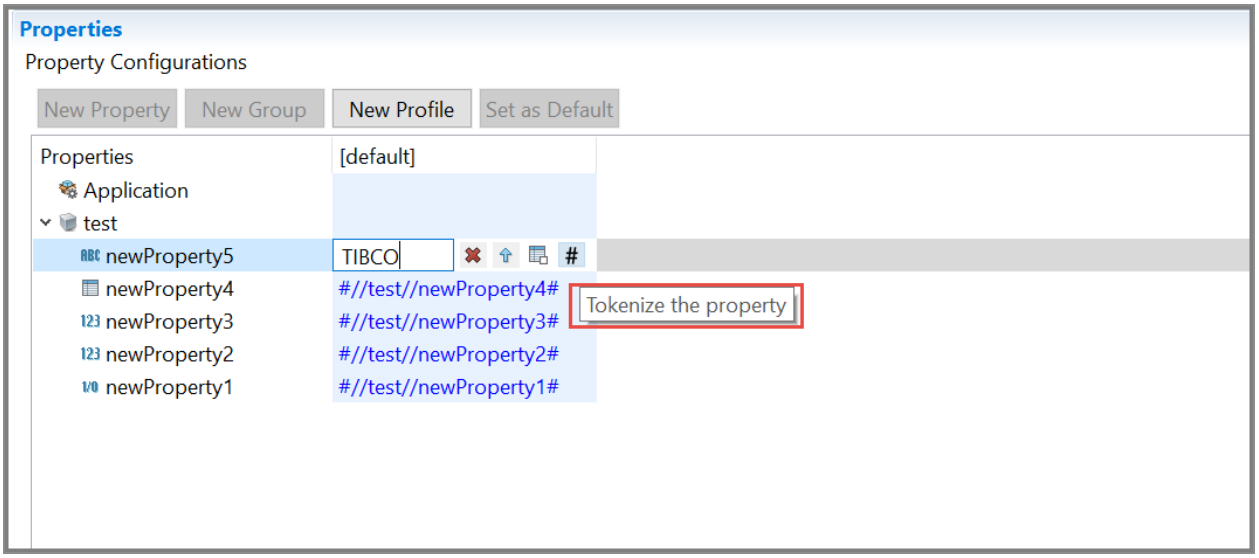
1. In **Project Explorer**, fully expand the **<*application_name*>.application**.

2. Double-click **Properties** under **Package Unit** to open the application properties page.
3. Click the on profile column (which shows the property value) of the module property that you want to map to the application property. If you have not created any custom profiles, then this is the **default** column.

   This puts the profile column in edit mode.
4. Enter **%%<*application_property_name*>%%** where *application_property_name* is the actual name of the application property to which you want to map the module property.

Tokenizing Application Properties

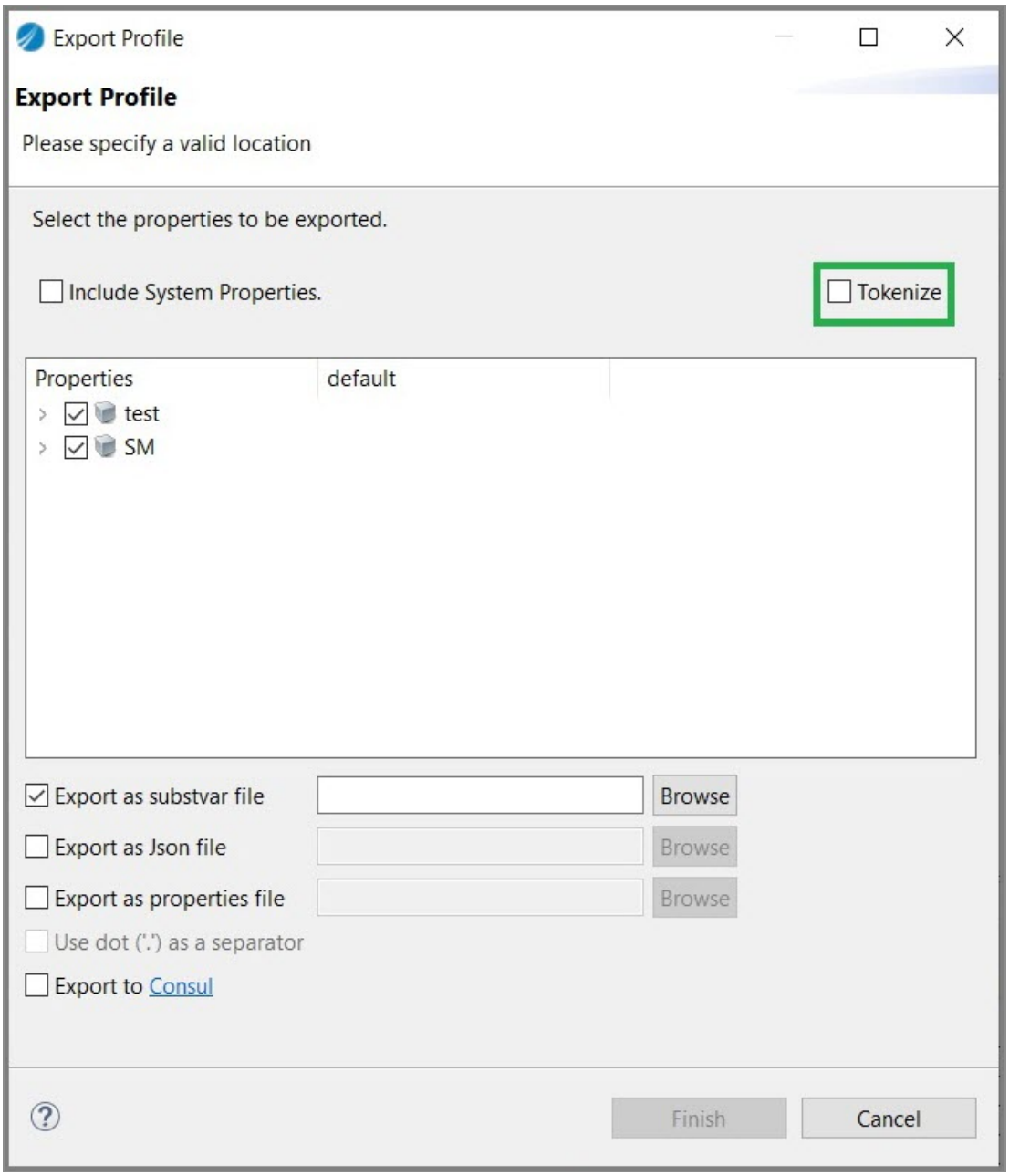
To tokenize application properties, a new button **Tokenize the property** is added in the Properties view for application properties. Tokenizing supports properties of type boolean, string, integer, long, and password.



After tokenization, the property value is set in the format //<ApplicationName>//<PropertyName>. Once the user tokenizes a property, the original default value for the property is lost.

Note: Tokenize the property button is available only for applications with Deployment Target set as Container.

By default, the **Tokenize** checkbox is unchecked. Select the **Tokenize** checkbox to auto-tokenize property value for the **Export as properties file** profile.



The following table lists the values of five different types of properties if the default tokenized values are not provided for all the properties and exported as properties file

Data Type	Property Name	Values Before Tokenization	Values After Tokenization/ Values Before Export	Values after Export	Values after Export in the properties file
		Default	**Default**	**Default**	**Default**
String	newProperty5		#//test//newProperty5#	#//test//newProperty5#	newProperty5
Password	newProperty4		#//test//newProperty4#	#//test//newProperty4#	a String: PASSWORD
Integer	newProperty3	0	#//test//newProperty3#	#//test//newProperty3#	0
Long	newProperty2	0	#//test//newProperty2#	#//test//newProperty2#	0
Boolean	newProperty1	false	#//test//newProperty1#	#//test//newProperty1#	false

The following table lists the values of five different types of properties if the default tokenized values are provided for all the properties and exported as properties file

Data Type	Property Name	Values Before Tokenization	Values After Tokenization/ Values Before Export	Values after Export	Values after Export in the properties file
		Default	**Default**	**Default**	**Default**
String	newProperty5	TIBCO	#//test//newProperty5#	#//test//newProperty5#	newProperty5
Password	newProperty4	***	#//test//newProperty4#	#//test//newProperty4#	a String: PASSWORD
Integer	newProperty3	1	#//test//newProperty3#	#//test//newProperty3#	0
Long	newProperty2	12345	#//test//newProperty2#	#//test//newProperty2#	0
Boolean	newProperty1	true	#//test//newProperty1#	#//test//newProperty1#	false

Note: For String type property in group, the values after export in properties file is `//test///newGroup/newGroup1/newProperty5=newGroup.newGroup.newProperty5`

The following table lists the values of five different types of properties if the default tokenized values are provided for all the properties and exported as properties file using dot (.)as a separator.

Data Type	Property Name	Values Before Tokenization	Values After Tokenization/ Values Before Export	Values after Export	Values after Export in the properties file
		Default	Default	Default	Default
String	newProperty5	TIBCO	#//test//newProperty5#	#test.newProperty5#	newProperty5
Password	newProperty4	***	#//test//newProperty4#	#test.newProperty4#	a String: PASSWORD
Integer	newProperty3	1	#//test//newProperty3#	#test.newProperty3#	0
Long	newProperty2	12345	#//test//newProperty2#	#test.newProperty2#	0
Boolean	newProperty1	true	#//test//newProperty1#	#test.newProperty1#	false

Note: For String type property in group, the values after export in properties file is `test.newGroup.newGroup1.Property5=newGroup.newGroup.newProperty5`

Exporting Properties to Consul Server

This section describes exporting default and tokenized properties to Consul Server

Before you begin

1. Select **Windows > Preferences > BusinessWorks > Service Registry**from the **Menu** bar.
2. Once the Service Registry wizard opens, add the URL as `http://<consul_server_host>:<port>` for the Consul Server in the service registry configuration section.
3. Click **Apply** and **OK**

To export tokenized properties to the Consul Server

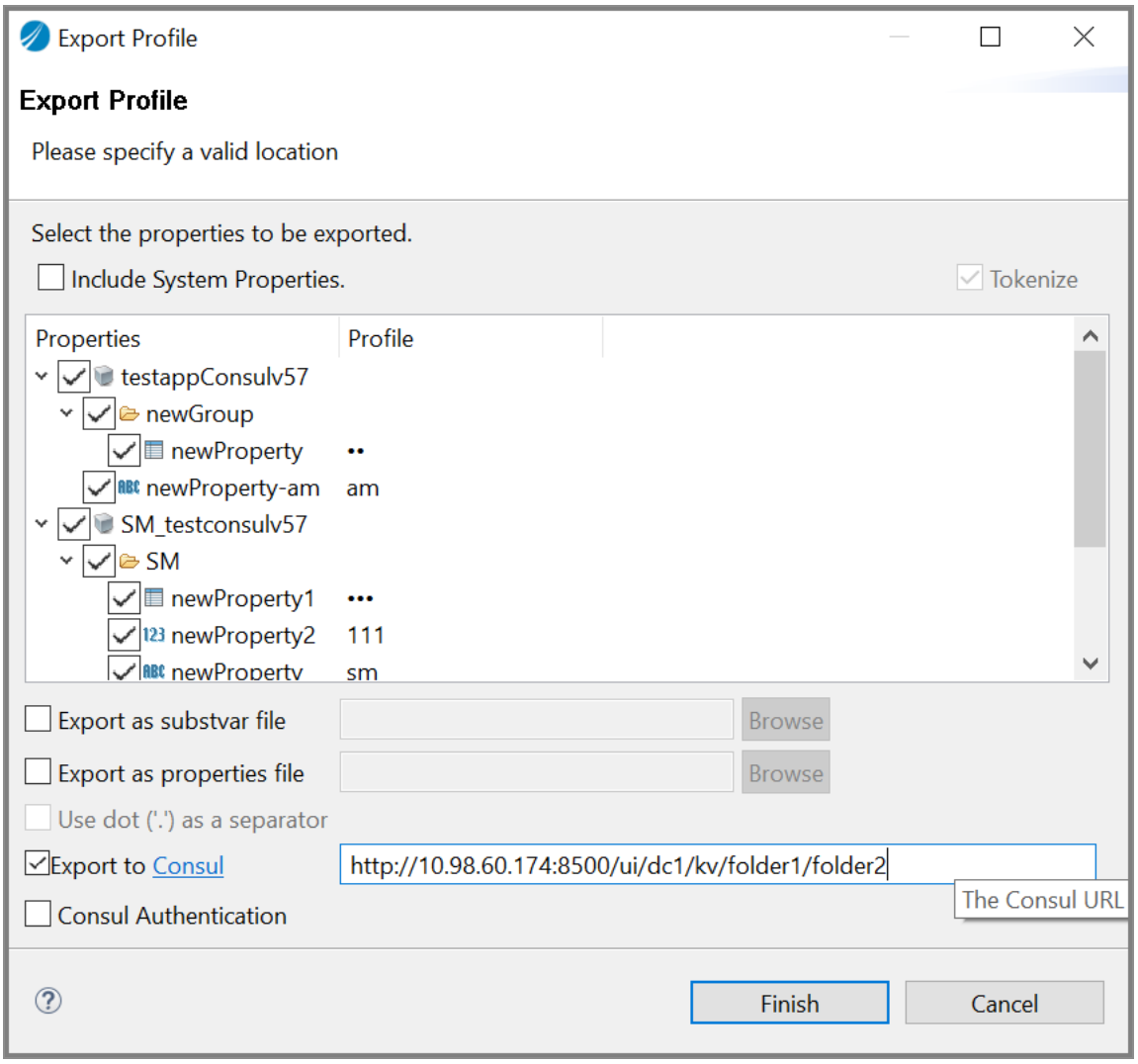
Procedure

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. To export to consul server, see select the **Export to Consul** checkbox and click **Finish**.
5. To enable authentication for the Consul Server, select the **Consul Authentication** checkbox and click **Finish**.

   This checkbox is available only when the **Export to Consul** checkbox is selected. When the **Consul Authentication** checkbox is selected, it is mandatory to provide a authentication token.

   Note:
   - In the properties view, the values of the properties in the profile can be anything, when the profile is exported as a properties file it changes to `#<PropertyName>#`
   - The authentication token that has write authorization only can be exported to Consul Server. For more information, see the Using Consul as a Configuration Management Service.



Note: It is now possible to export the profile properties to the specific custom folder under **Consul > Key-Value** by specifying its path in the **Export to Consul** checkbox.

The following table lists the values of five different types of properties if the default values are not provided for all the properties and exported to consul server

Data Type	Property Name	Values Before Export	Values After export	Values After Export in Consul Server
		Default	**Default**	
String	newProperty5		#newProperty5#	newProperty5
Password	newProperty4		#newProperty4#	<encrypted value of the password>
Integer	newProperty3	0	#newProperty3#	0
Long	newProperty2	0	#newProperty2#	0
Boolean	newProperty1	false	#newProperty1#	false

The following table lists the values of five different types of properties if the default values are provided for all the properties and exported to consul server

Data Type	Property Name	Values Before Export	Values After export	Values After Export in Consul Server
		Default	**Default**	
String	newProperty5	TIBCO	#newProperty5#	TIBCO
Password	newProperty4	***	#newProperty4#	<encrypted value of the password>
Integer	newProperty3	1	#newProperty3#	1

Data Type	Property Name	Values Before Export	Values After export	Values After Export in Consul Server
Long	newProperty2	12345	#newProperty2#	12345
Boolean	newProperty1	true	#newProperty1#	true

The following table lists the values of five different types of properties if the default tokenized values are provided for all the properties and exported to consul server

Data Type	Property Name	Values Before Export	Values After export	Values After Export in Consul Server
		Default	**Default**	
String	newProperty5	#//test//newProperty5#	#newProperty5#	newProperty5
Password	newProperty4	#//test//newProperty4#	#newProperty4#	a String: PASSWORD
Integer	newProperty3	#//test//newProperty3#	#newProperty3#	0
Long	newProperty2	#//test//newProperty2#	#newProperty2#	0
Boolean	newProperty1		#newProperty1#	false

Data Type	Property Name	Values Before Export	Values After export	Values After Export in Consul Server
		#//test//newProperty 1#		

Note: After exporting String type property in group to consul server, the exported profile has key and value as `newGroup.newGroup1.newGroup1Property5`

Using Different Values of the Same Property Type in Different Activities within a Process

Each property can have a single value assigned to it. A property can be assigned to multiple activities within a process. All the activities using that particular property get the same value that is assigned to the property. If you want to have different values of the same type for different activities within a process, you must create different process properties for each value that you want to use in an activity. You then assign one of the created properties to each activity based on its value that you want to use for the activity. For example, a property called jdbcProperty of type JDBC Connection shared resource can be assigned to multiple JDBC activities in a process. The JDBC connection itself that is assigned to the property is the same across all the activities that use that property. If you change the connection to point to a different resource, then all activities using jdbcProperty get the new resource value. If you want each activity to use a different JDBC connection, you must create a separate process property of type JDBC Connection for each activity and assign each property the desired resource value. This can let you assign a different JDBC Connection shared resource to each activity.
Activities are bound to the property name and type, not the property value. The value gets injected into the property during runtime. Hence, if you change the value that is assigned to a property in one activity, the property value changes to the new value for all activities using that property.

To create a new process property for an activity that can automatically be associated with the activity, follow these steps:

1. In the **Process Editor**, click the activity icon for which you want to create a process property.
2. In the **Properties** view, click  . A new process property gets created.
3. Click the name to edit it if need be.
4. Set a value for the property by selecting the value type.

Working with Templates

A Template provides you a basic structure to create a TIBCO BusinessWorks™ Container Edition application project.

You can share the template among your team members to avoid creating a project again from scratch. It helps speedy project development with reduced errors that may occur in collaborative application development.

Create a template in such a way that it is not having too much functionality but serves as a base framework on which you can further develop additional capabilities.

Exporting a Project as a Template

You can export already created project, or first create a project with some basic functionalities and then export the project as template.

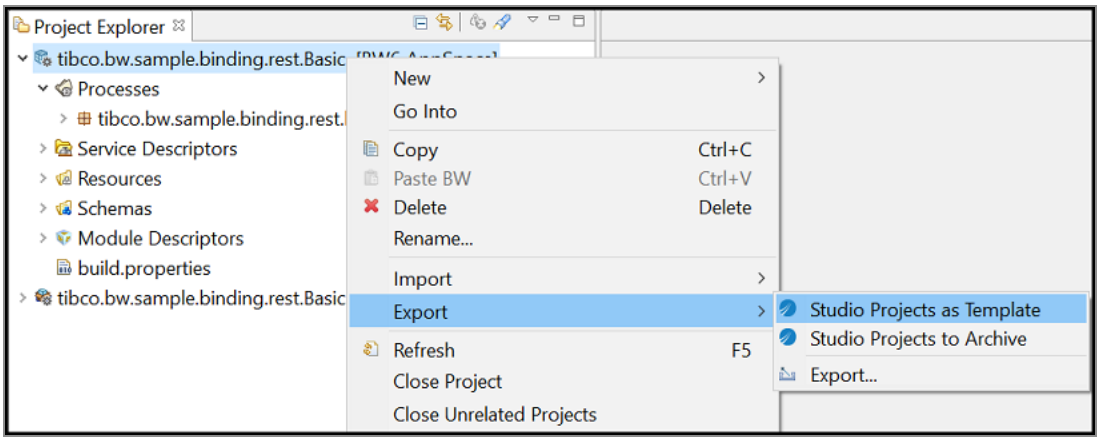
You can export an application module, a shared module, or a binary shared module as a template. Before creating a template, make sure that there are no errors in the project. The following steps show how to export a Binding REST sample as a template.

Before you begin

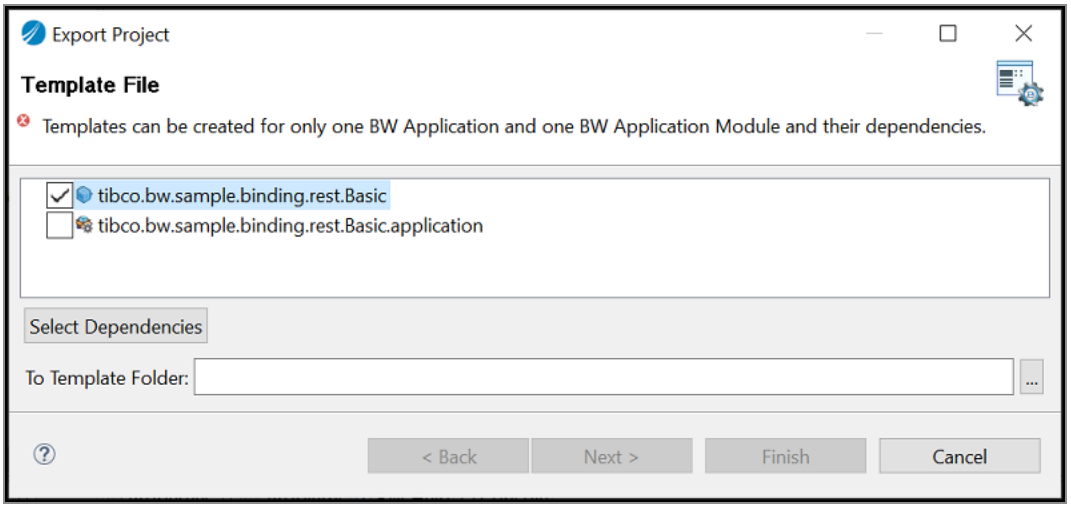
You have imported `tibco.bw.sample.binding.rest.Basic` sample in the Project Explorer. For more information, see "Accessing Samples" in *TIBCO BusinessWorks™ Container Edition Samples*.

Procedure

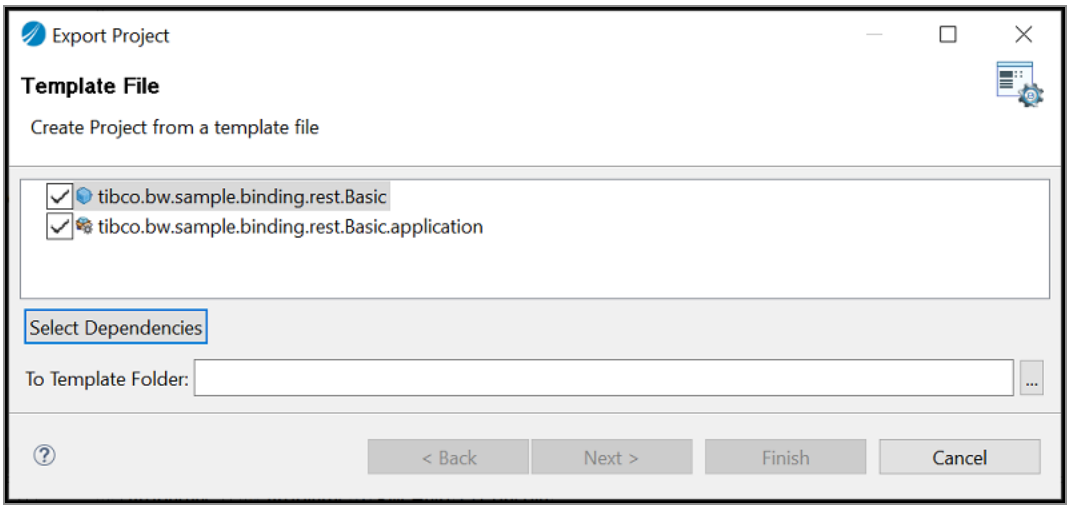
1. Select the project. Right-click and select **Export > Studio Projects as Template**



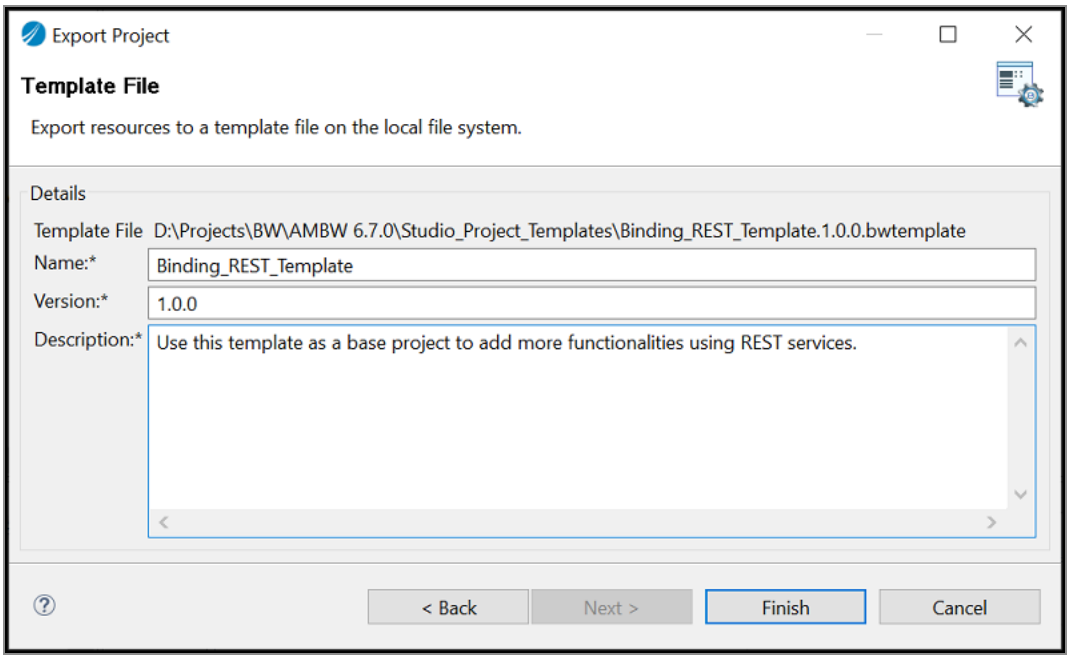
The Export Project dialog is displayed.



2. To select all dependencies associated with the project, click the **Select Dependencies** button .



3. Provide the template folder path and click **Next**.
4. Provide required details such as a suitable name, version and description to the template.



By default, the template name is the same as the project name.

5. Click **Finish**.

   The `Binding_REST_Template.1.0.0.bwtemplate` file is created at the specified location.

Creating a Project from a Template

You can create a project using an existing template.

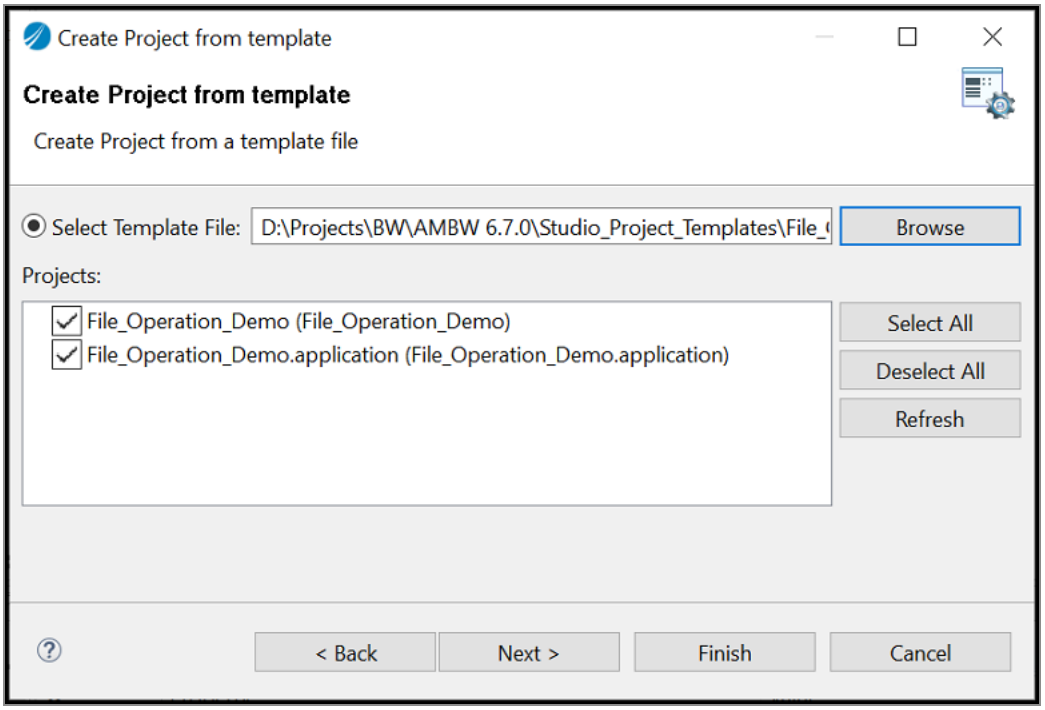
Procedure

1. Go to **File > New > Project > BusinessWorks** and select **BusinessWorks Project from Template**. Click **Next**.

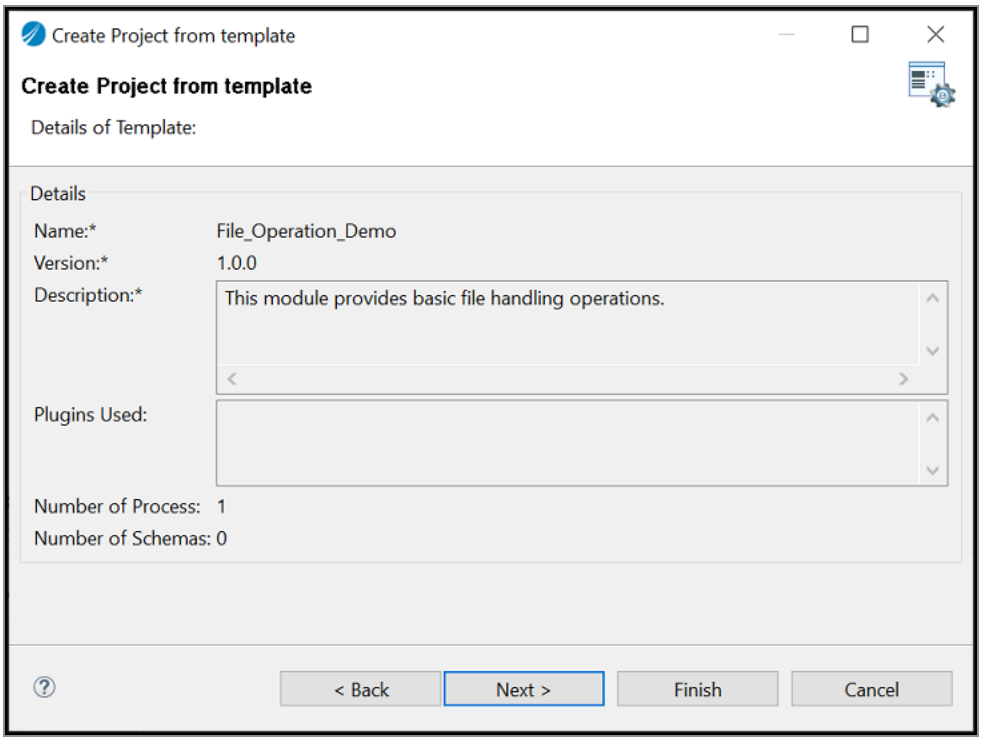
   The Create Project from the template dialog is displayed.

2. Provide the template file location.

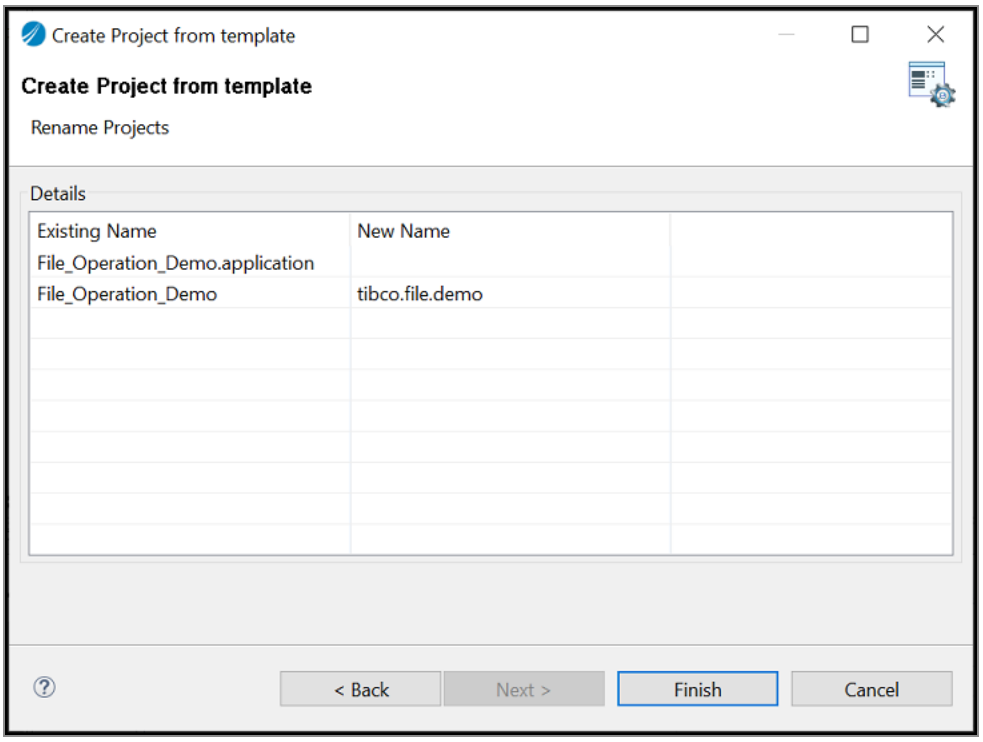
   A project in a template along with its dependencies is displayed.



3. To see information about the template, click **Next**.

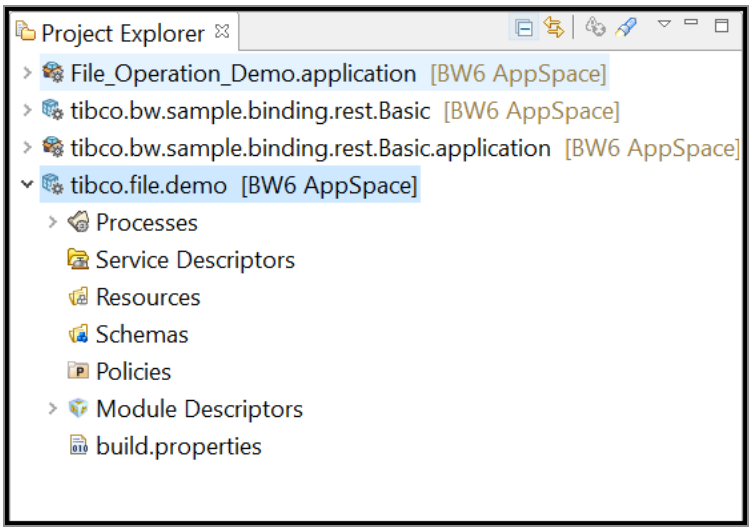


4. To rename a project, click **Next**. In the **Existing Name** column, select the project and then provide the name in the **New Name** column.



5. Click **Finish**.

   Project is successfully created in the Project Explorer.



   The metadata file is deleted after you import a template in the workspace.

   The project's `manifest.MF` file has two new fields - TIBCO-BW-TEMPLATE-VERSION and TIBCO-BW-TEMPLATE-NAME.

Modifying an Existing Template

You can add more functionalities to an existing template and save the project as a modified template.

Before you begin

Create a project from an existing template in your workspace.

Procedure

1. Open a process in the new project created.
2. Add some more activities. Reconfigure the activities if necessary and save the process.
3. Export the project again as specified in Exporting a Project as a Template.

   The template file is stored at the specified location.

Using Additional Features

Complex business processes make use of additional features such as process scopes, fault handlers, 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.

Before you begin

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 are encapsulated in a new scope.

2. Configure the new scope from the Properties view.

   General Tab

Field	Description
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.

Description Tab

Field	Description
Description	Enter a description for the new scope.

Variables tab: You can add local variables to the group from the **Variables** tab. For more information about adding scope variables, see Adding Scope 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.

Select Schema Element Declaration

Field/Action	Description
Workspace	When selected, the variable is valid only during the design-time.
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 checkbox to display all the XSD elements in the module. This checkbox is selected by default.
Include Process	Select the checkbox to display the process inline schemas in the

Field/Action	Description
Inline Schemas	module.
Include WSDL Inline Schemas	Select the checkbox 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	Add a primitive element to the new schema using the icon **Add Primitive Element**  . The new primitive element appears listed under the root element. Double-click the element to configure it. 
Primitive Types	Select the primitive type for the element from the drop-down list: • String • Integer • Decimal • Boolean • Date & Time • Binary

Field/Action	Description
	• URI • Any
Subtypes	Select the subtypes for the element from the drop-down list: • 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.

Field/Action	Description
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
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

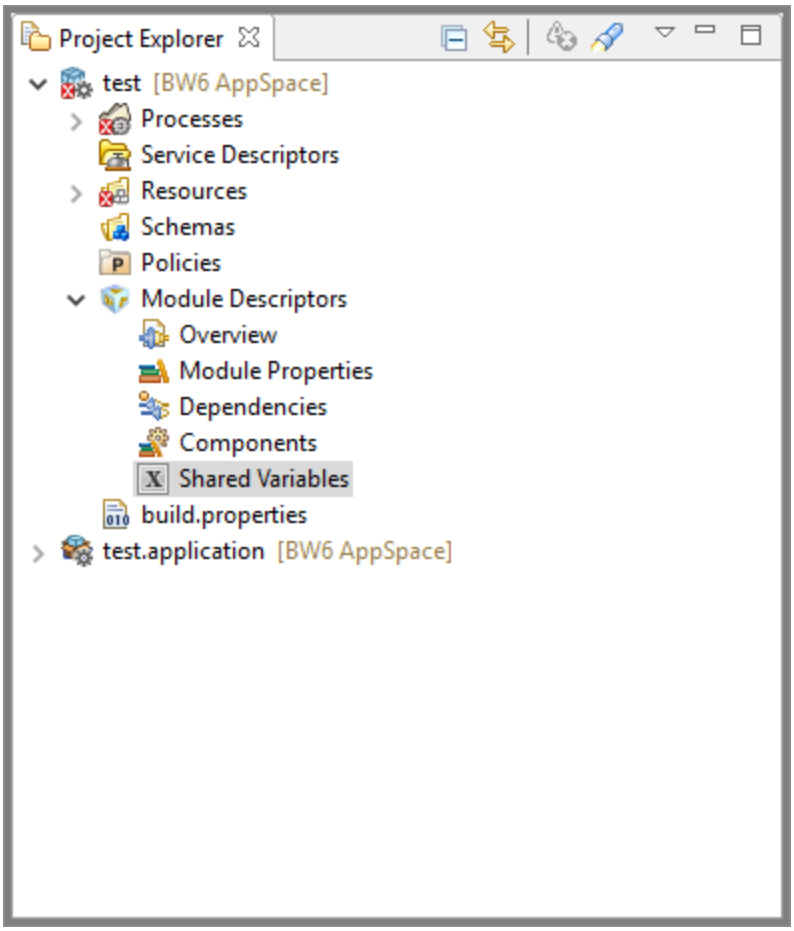
Defining and Using Shared Variables

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 BusinessWorks Container Edition 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	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: • String • Integer • Boolean • Date&Time • Complex Element...
	Persistent	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. Select the checkbox 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".

Tab	Field Name	Description
		Note: The engine persistence must be configured for the current value of the module shared variable to persist. **Note:** This checkbox only displays when configuring module shared variables. Job shared variables cannot be configured to be persistent.
Description	Description	Description for the shared variable.
Initial Value	Initial Value	Enter an initial value for the shared variable. Select one from the following options: • 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.

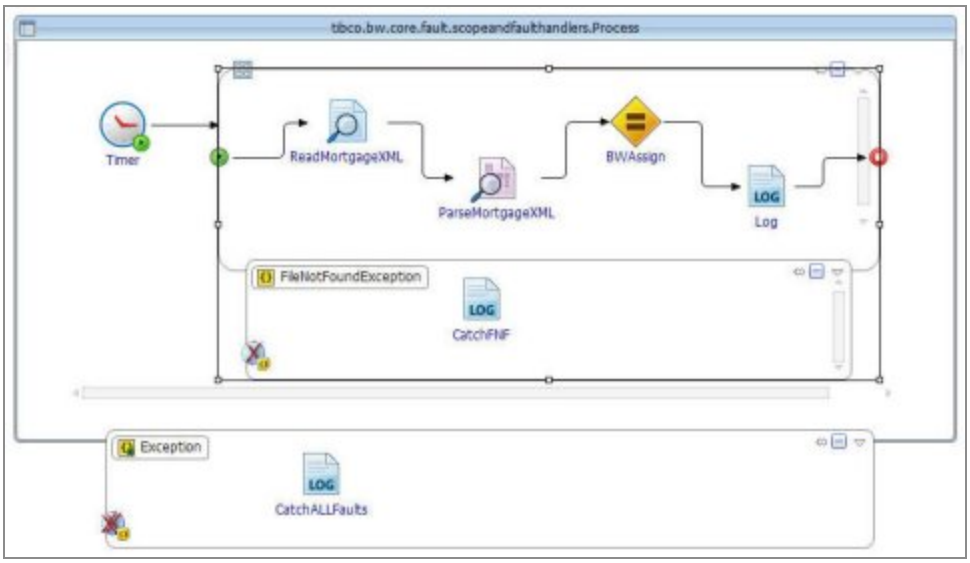
Using Fault Handlers

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**.

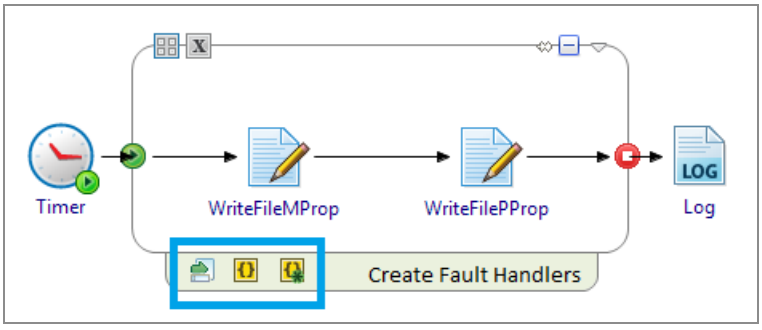
Fault handlers can be defined at the process level, or at a scope level within a process. The diagram below shows two fault handlers - one defined at the process level and the other defined at an inner scope level.

Fault Handler Attached to an Inner Scope



Procedure

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



3. 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 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.

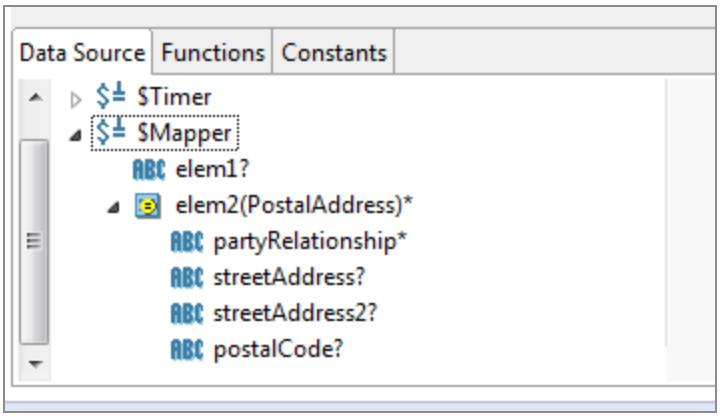
> **Note:** Ensure that the **Type** you select is an extension of the base type.

6. Optional. Select the **Cardinality** checkbox, 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...**.

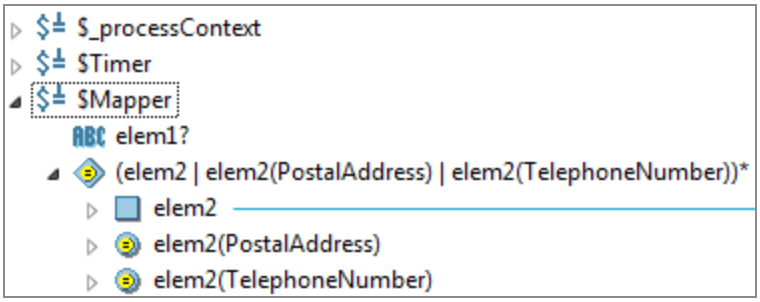
2. In the Coercion window, click the ✚ icon to add a coercion.
3. Accept the default option for the **Component Type** field.
4. Select a schema for the **Namespace** field by choosing an option from the drop-down menu, or click the browse icon to view a list of available schemas in the application module.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

   > **Note:** Ensure that the **Type** you select is an extension of the base type.

6. Optional. Select the **Cardinality** checkbox, 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.

> **Note:** 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.

Mapper Preferences

You can set the following mapper related preferences from TIBCO Business Studio for BusinessWorks. Navigate to **Window > Preferences > Mapper**.

Field	Description
Show mapping for selected	Select the checkbox to only display mappings for elements

Field	Description
element only	you select in the mapper.
Treat if-statement as required element	Select the checkbox to render an XML node correctly when in an if condition the value assigned for false is zero.
Check potential type mismatch post migration	Select the checkbox to display an error for invalid conversion.
Check unsynchronized runtime binding	Select this checkbox to detect the difference in the stored runtime XSLT and the computed runtime XSLT.
Avoid generating empty element for optional-nil element mapping to optional element	Select the checkbox to avoid generating empty elements.
Disable implicit-if existence checking	Select the checkbox and clean the project to remove the `<xsl:if test= "$Mapper/Element1">` from Optional to Optional mapping.
Generate if-statement for a constructor function on optional element	Select the checkbox to add the 'if-test' condition for the optional-to-optional mappings that are type-casted using constructors and to eliminate empty tags in the runtime output.
Assume all inputs untyped	After migration, boolean() function always evaluates to false with the XSLT version 1.0 and does not display any validation errors. Select the checkbox and clean the project. The errors related to untyped input are displayed. Use the **Quick Fix** option to resolve those errors and execute the project.
Show warning for expressions containing optional-nill elements	Select the checkbox and clean the project to detect expressions containing optional-nill elements.

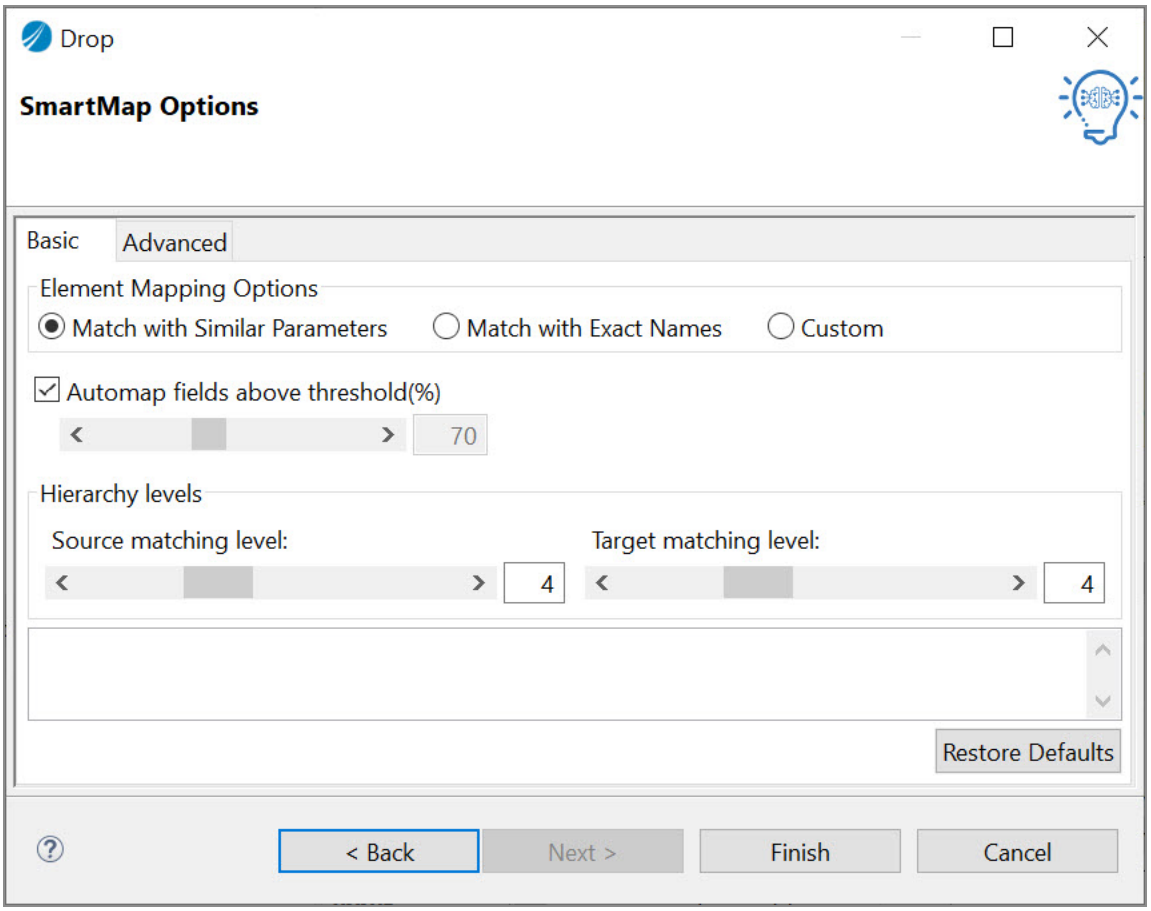
Field	Description
Enable quickfix to change XPath version from 2.0 to 1.0	Select the checkbox to remove the faulted transitions or group configuration errors in projects that were migrated to XPath 2.0.

Smart Mapper

Smart Mapper provides the ability to intelligently determine how data is to be mapped in the activity's input tab. Smart mapper provides functionality to match the source element and target element by comparing various parameters such as names, data types, depth, cardinality, and element ancestors.

The Smart Mapper feature and recording the user mapping selections is enabled by default. To disable the Smart Mapper, go to **Window > Preferences > Mapper** and unselect the **Enable Smart Mapper** checkbox. This also unselects the **Record user mapping selections** checkbox. Selecting the **Record user mapping selections** checkbox helps to record your manual mappings once you do a smart mapping. Click **Apply and Close**.

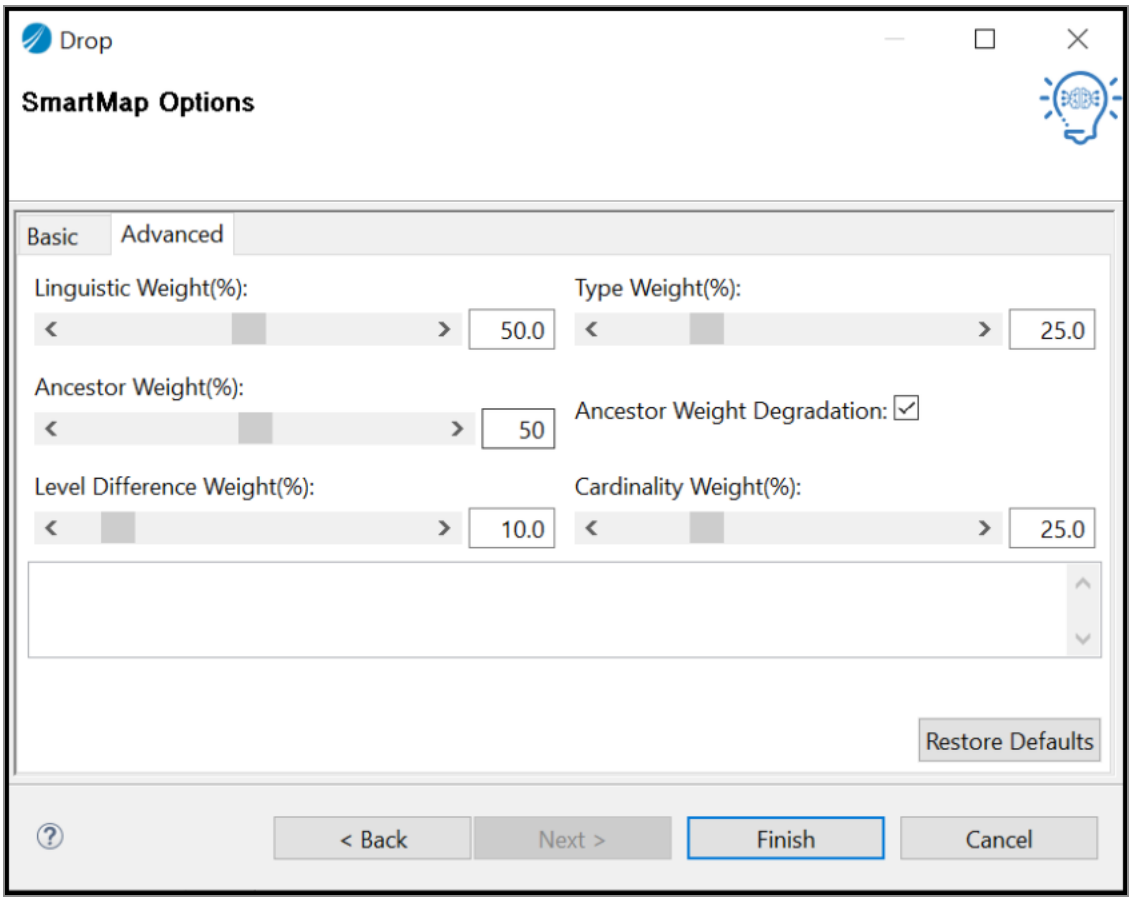
The smart mapper has the following options on the **Basic** tab:



Field	Description
Element Mapping Options	• **Match with Similar Parameters**: Default mapping policy, in which all parameters are considered while generating the mapping. Parameters are optimized to generate the best suitable matches. • **Match with Exact Names**: Mapping is done by considering the similarity in names rather than other parameters • **Custom**: Select this option to configure parameters manually which best suits your schemas and use cases. When you select this option, all fields in the **Advanced** tab are configurable.
Automap fields above	The threshold value is the minimum value to qualify for element matching. For more number of matches, lower this value, and for more accurate selection,

Field	Description
threshold(%)	increase the threshold value. The default value for the **Match with Similar Names** option is 70% and for the **Match with Exact Names** option is 90%. Clear the checkbox to customize this value.
Source matching level	Level up to which the mapper must look for a matching element from the selected source element for dragging. The supported range is 1 to 10. The default value is 4. **Warning:** Increasing this value might create memory related issues based on machine configuration and recursiveness of the schema.
Target matching level	Level up to which the mapper must look for a matching element from the to-be-dropped target element. The supported range is 1 to 10. The default value is 4. **Warning:** Increasing this value might create memory related issues based on machine configuration and recursiveness of the schema.
Restore Default	Restores all the values to default in the **Basic** tab.

The smart mapper has the following options on the **Advanced** tab:



Field	Description
Linguistic Weight(%)	It focuses more on the similarity in names, rather than its data type, ancestor, or cardinality. Increasing the value maps more similar named elements even when the other parameters such as datatype, cardinality, or ancestor names do not match. Similarly, reducing the value to give lesser priority to the similarity of the names of the element, and more to the other parameters based on their settings. The value ranges from 0.0 to 100.0. The default value for the **Match with Similar Names** option is 50.0 and for the **Match with Exact Names** option is 100.0.
Type Weight (%)	Type weight focuses on the data type similarity of the elements, which

Field	Description
	means increasing type weight considers the data type of the elements in priority to other parameters like similarity in name (linguistic weight, cardinality etc.). The value ranges from 0.0 to 100.0. The default value for the **Match with Similar Names** option is 25.0 and for the **Match with Exact Names** option is 0.0.
Ancestor Weight(%)	It focuses on ancestor names of the potential candidates from source and target elements. When set to zero, it does not consider the ancestor name of the element. When set to 100, it checks the value of the Linguistic Weight, and gives equal weight to the ancestor name matching. The value ranges from 0 to 100. The default value for the **Match with Similar Names** option is 50 and for the **Match with Exact Names** option is 0.
Ancestor Weight Degradation	If you clear this checkbox, ancestors at any level have the same priority while considering ancestor similarity. If you select this checkbox, the smart mapper gives priority to the parent element, then its ancestor and so on. By default, for the **Match with Similar Names** option, the checkbox is selected and for the **Match with Exact Names** option, the checkbox is clear.
Level Difference Weight(%)	It decides the priority to be given to the difference in levels of an element in a complex schema. Increasing the value considers the difference in level more than other parameters. It checks for levels of source and target elements in the schema. When you increase this value, it tries to search matching elements in the same level and vice versa. The value ranges from 0.0 to 100.0. The default value for the **Match with Similar Names** option is 10.0 and for the **Match with Exact Names** option is 0.0.
Cardinality Weight(%)	It considers the cardinality of the element. If its value is set to zero, it does not consider the cardinality at all. If you increase its value, it considers the

Field	Description
	cardinality more than the other parameters. The value ranges from 0.0 to 100.0. The default value for the **Match with Similar Names** option is 25.0 and for the **Match with Exact Names** option is 0.0.
Restore Default	Restores all the values to default in the **Advanced** tab.

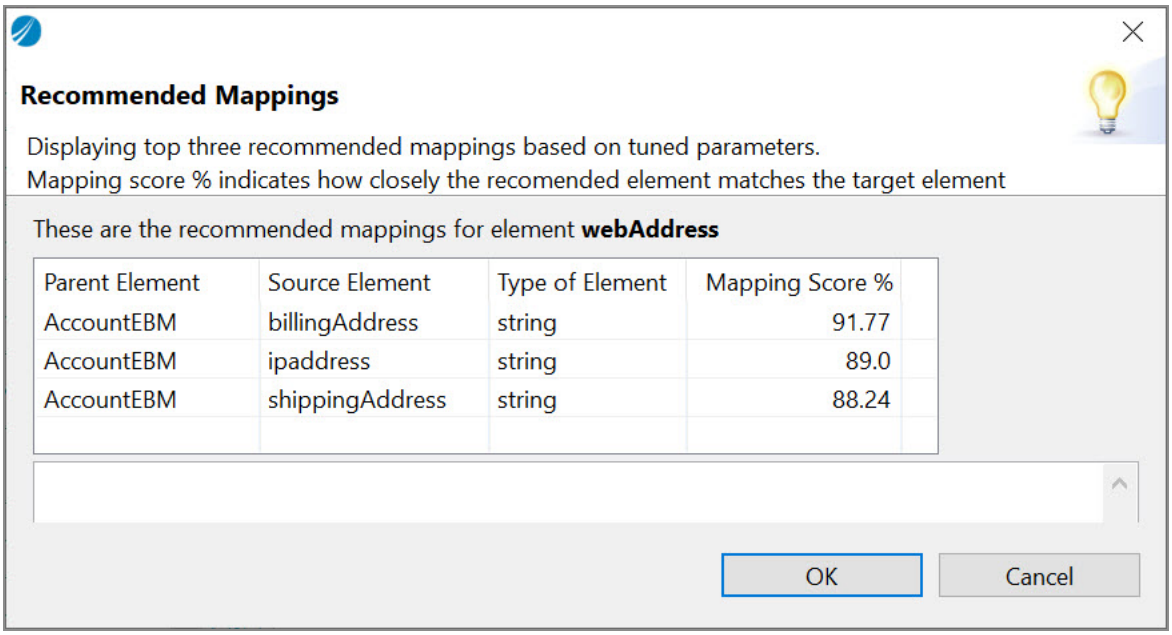
If there is already an existing mapping and then you apply smart mapping. The following user-created structures persist and other mappings are overwritten:

- Coercion
- Choice conditions
- Variables
- Constants
- statement with functions
- For Each statements

To obtain better mappings, increase or decrease the values and tune the parameters as per your requirements.

After performing a smart mapping operation, if some of the simple-type target elements are not matched, right-click the target element and select **Recommended Mappings**.

A list of elements is displayed that are possible close matches of the target element.



Note: The elements out of the matching levels do not have recommendations, and the recommendation information is lost if the current mapping session expires.

The recommended elements are based on the mapping option selected or the parameters tuned in the custom mapping.

The following attributes of the matching source elements are displayed:

- Parent Element - Name of the immediate parent of the recommended source element
- Source Element - Name of the recommended source element
- Type of Element - Data type of the recommended source element
- Mapping Score % - The rating of the recommended source element to the selected target element

If the recommended mappings are not as per your requirements, you can manually map the elements. Smart mapper remembers this manual mapping.

The next time you perform a smart mapping on the same schema, the smart mapper tries to map the same elements used in the manual mapping. This functionality is applicable only for that workspace.

If you have manually mapped an incorrect element during smart mapping and do not want the smart mapper to record this particular mapping choice, you can simply delete the mapping.

Alternatively, replace the current mapping with the desired element. This replaces the element recorded by the smart mapper.

If you no longer want the smart mapper to record your manual mapping choices, you can clear the **Record user mapping selections** checkbox from the Mapper preference dialog.

> ⚠ **Warning:** If you want to delete all the previously recorded manual mappings by the smart mapper, right-click on the root element in the target and select the **Reset to Default Recommended Mappings** option. This option deletes all the recorded mappings within that workspace.

Sharing SmartMapper Recommendation through External Database

With TIBCO BusinessWorks Container Edition 2.7.2, you can connect the smart mapper to an externalized database and share the mappings done on one machine across different processes, applications, and workspaces so that multiple users can access it. Earlier the mapping data used to be saved in the metadata of a workspace. Now, you can use a dedicated database to store the mapping data. Currently, following databases are supported to store the mappings:

- PostgreSQL
- Microsoft SQL Server
- Oracle Database

When you map the target data elements to the source data elements (expressions) for the first time, two tables are created in the database. One of these two tables contains the mapping and the other one contains the version of the mapping done.

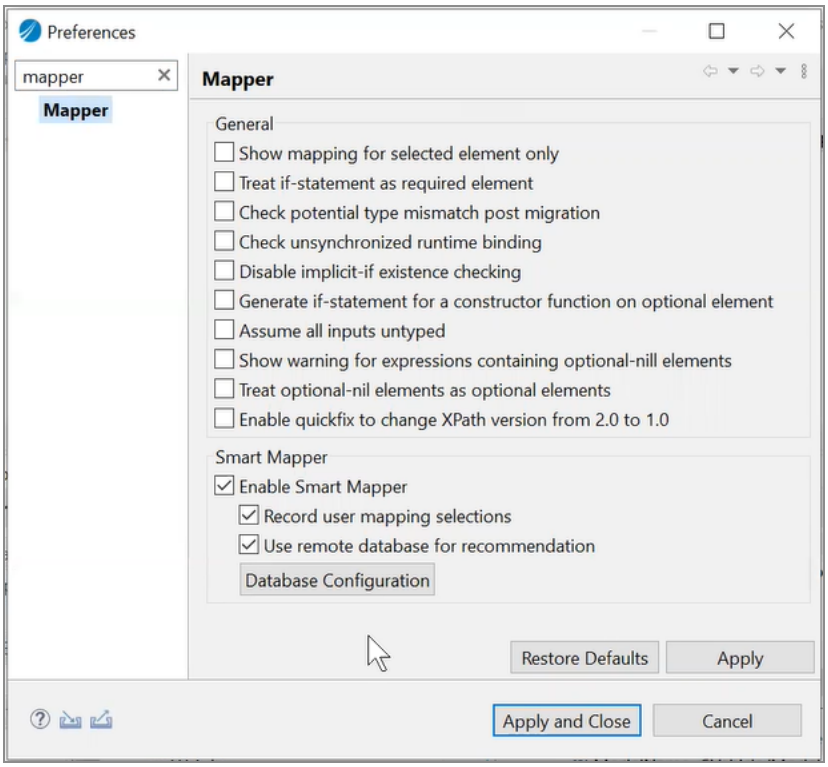
To enable the smart mapper feature:

Before you begin

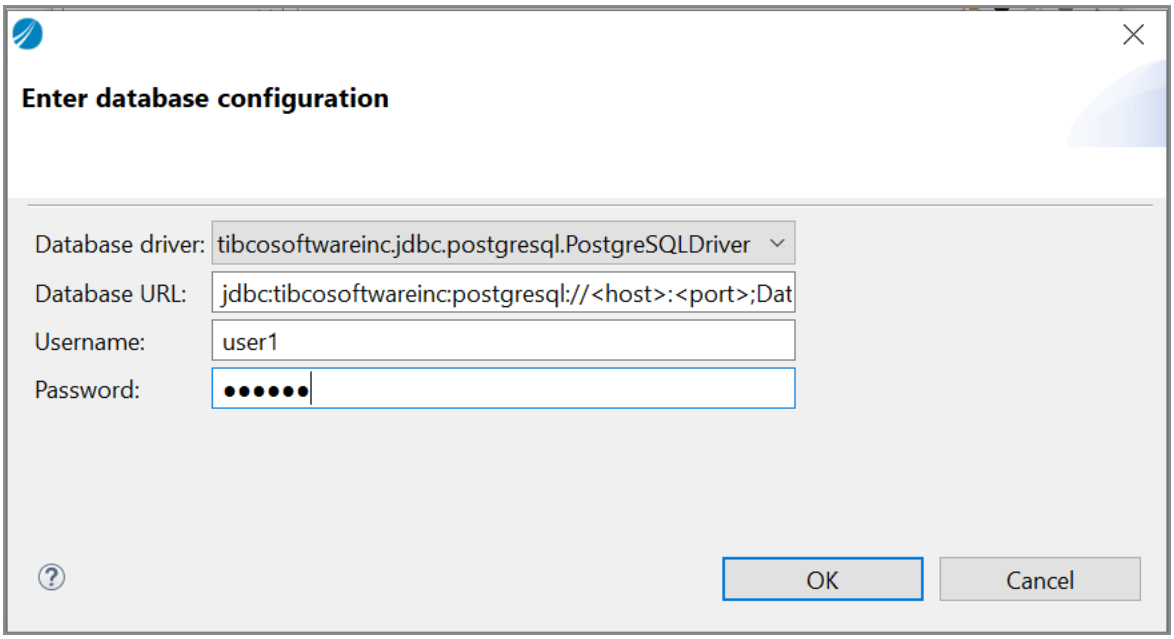
Create and set up a database to use for mapping the data elements.

Procedure

1. Navigate to **Window > Preferences > Mapper**. The mapper options are displayed.



2. Select the **Enable Smart Mapper** checkbox available under the Smart Mapper section.
3. Click **Database Configuration**. The **Enter database configuration** window is displayed.



4. Select a database driver from the list of available drivers. When you select the database driver, the database URL is automatically generated.
5. Enter the **host**, **port**, and **dbname** or **serviceName** values in the database URL.
6. Enter the **Username** and **Password** in the respective fields for the selected driver.
7. Click **OK**, **Apply**, and then **Apply and Close**.

Result

The Smart Mapper feature is now enabled with a configured database.

What to do next

Now you can proceed to map the data elements in your application.

To create a mapping between the data elements and expressions:

Procedure

1. Open the required process in your application.
2. Select the activity in which you want to map the elements.
3. Select the **Input** tab under Properties tab of the selected activity.
4. Drag an element from the **Data Source** tab and drop it on the required expression

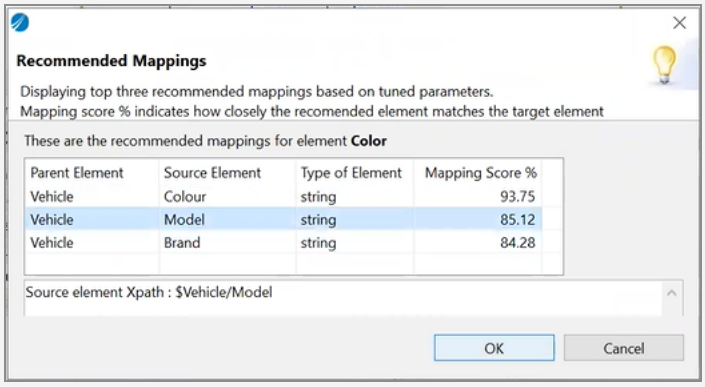
available in the right side pane. The **Drop** wizard is displayed.

> **Note:** The first time you map the elements, drag the complex element and drop it on the required expression and select the **SmartMap** option, and click **Next**, and then **Finish**. The two tables, with mapping and version details, are created and then proceed with mapping the individual elements.

5. Select the **SmartMap** option, click **Next**, and then **Finish**.

Note:

- You can check the recommended mappings by right-clicking the data element and selecting the **Recommended Mappings** option.



- If you are mapping a data element to an already mapped expression, the application displays a message that the expression already exists and whether you want to replace existing expression or incorporate into existing expression. By default, the **Replace existing expression** option is selected.

- To remove the mapping of an individual data element, right-click the data element and select the **Remove Mapping** option. The mapping done for the data element is deleted from the process as well as the database.
- To remove all the data element mappings at once, right-click the complex data element, and select the **Reset to Default Recommended Mappings** option. All the mapped elements are deleted from the process as well as the database. If you remove the mapping of a complex data element by right-clicking the complex data element and selecting the **Remove Mapping** option, the mapping is removed from the process only, but not

the database.

- To see the mappings done by another user, you must click anywhere outside the process editor pane and then click the activity for which the mapping was done by another user. Clicking anywhere outside the process editor refreshes the properties tab to display the updated mappings.

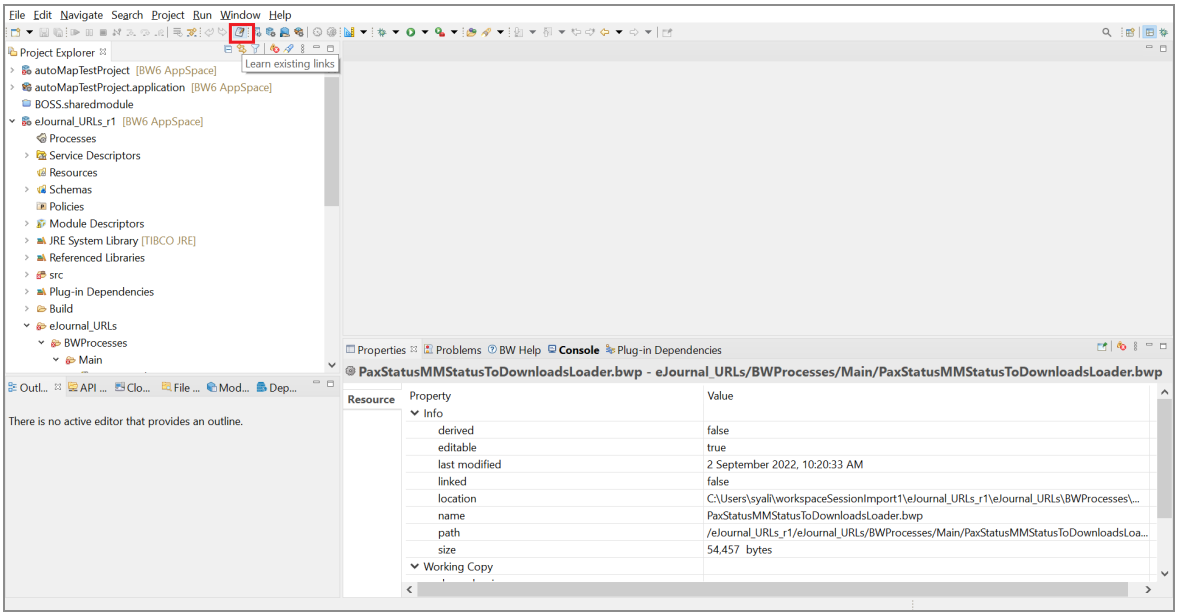
Learn Mapping Data

With the Learn Existing Links feature, you can store the mapping data of all projects in the current workspace. To use this feature, you first need to configure the database and then click the **Learn existing links** button. All the mapping data gets stored on the respective database.

Perform these steps to learn existing data:

Procedure

1. Configure the Smart Mapper database. For more information, see Sharing SmartMapper Recommendation through External Database.
2. After configuring the database, click the **Learn existing links** icon.



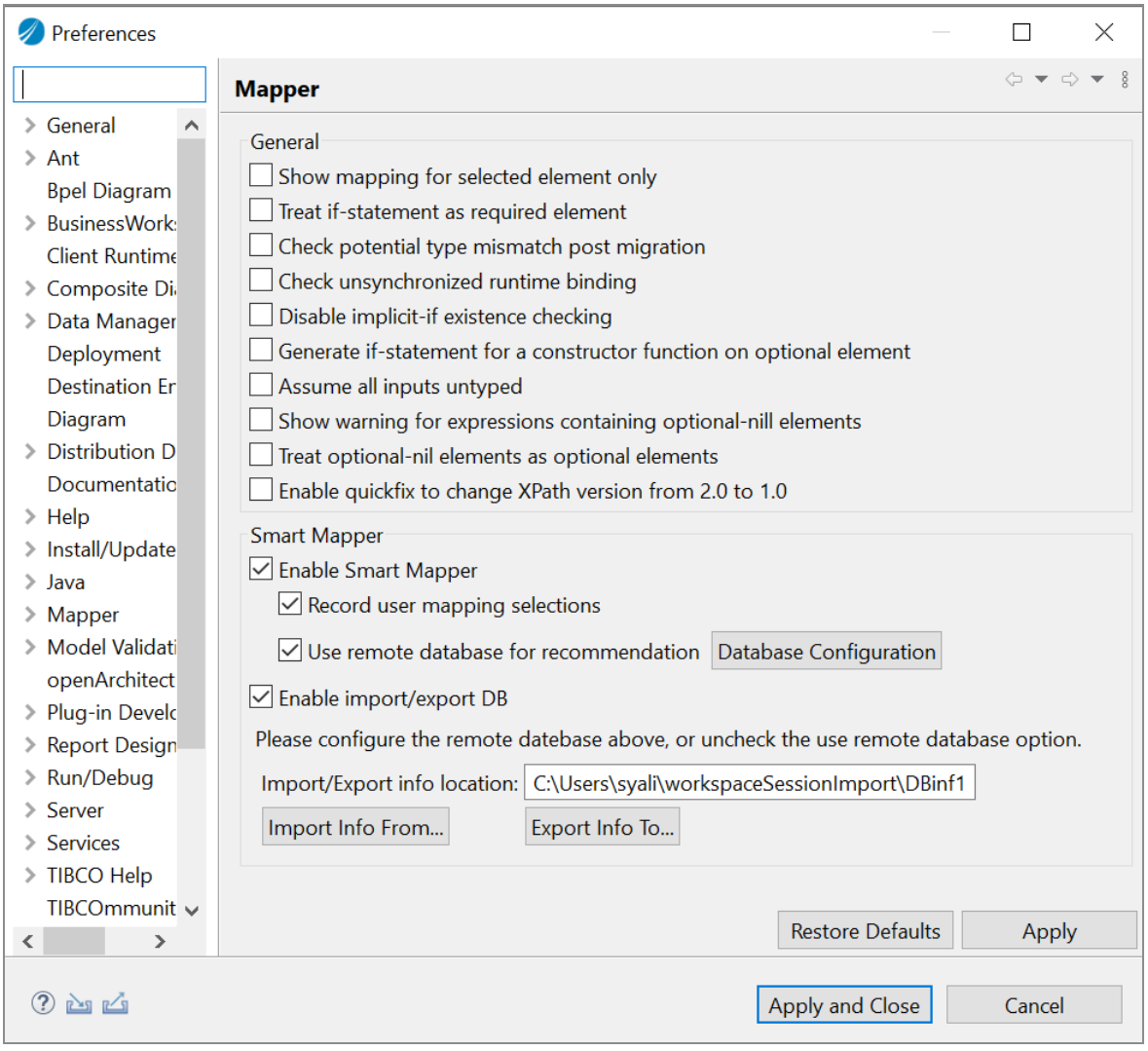
Importing and Exporting Recommendation Info

You can use this feature to import or export recommendation info. When you click the **Import** button, the mapping data present in the DBnfo file imports to the configured database. Similarly, when you click the **Export** button, the mapping data present in the database exports to the DBinfo file.

Perform these steps to import/export recommendation info:

Procedure

1. Configure the Smart Mapper database. For more information, see Sharing SmartMapper Recommendation through External Database.
2. Navigate to **Window > Preferences > Mapper**.



3. Enter the correct location of the DBinfo file.
4. Click the **Import Info From...** button to import mapping data from the DBinfo file to the configured database.
5. To export mapping data from the database to the DBinfo file, click the **Export Info To...** button.

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** checkbox.
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 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_Wrkspace\BW6.x\V.x`
3. 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`

> **Note:** 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.

4. Deploy the application after the process diagram is generated.

Password Obfuscator Utility

The password obfuscator utility is used to encrypt sensitive data such as passwords when configuring user defined services or environment variables. This utility can also be used to encrypt data using the custom encryption key.

This utility can be found at `<TIBCO_HOME>/bwce/version/bin/bwobfuscator`.

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.

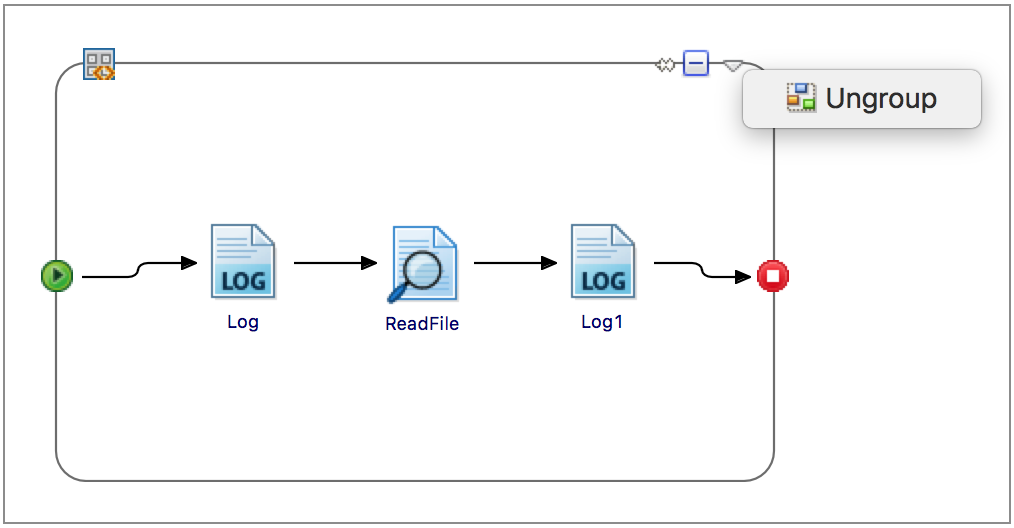
Procedure

1. In TIBCO Business Studio for BusinessWorks, click **Window > Preferences**. On macOS X, click **TIBCO Business Studio > Preferences**.
2. In the Preferences dialog, 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**. For instructions, see Configuring the Ungroup Preferences. When this preference is selected, the following actions 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.

> **Note:** 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 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 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.

- 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 TIBCO Business Studio for BusinessWorks. 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 Container 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 is 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 TIBCO Business Studio for 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

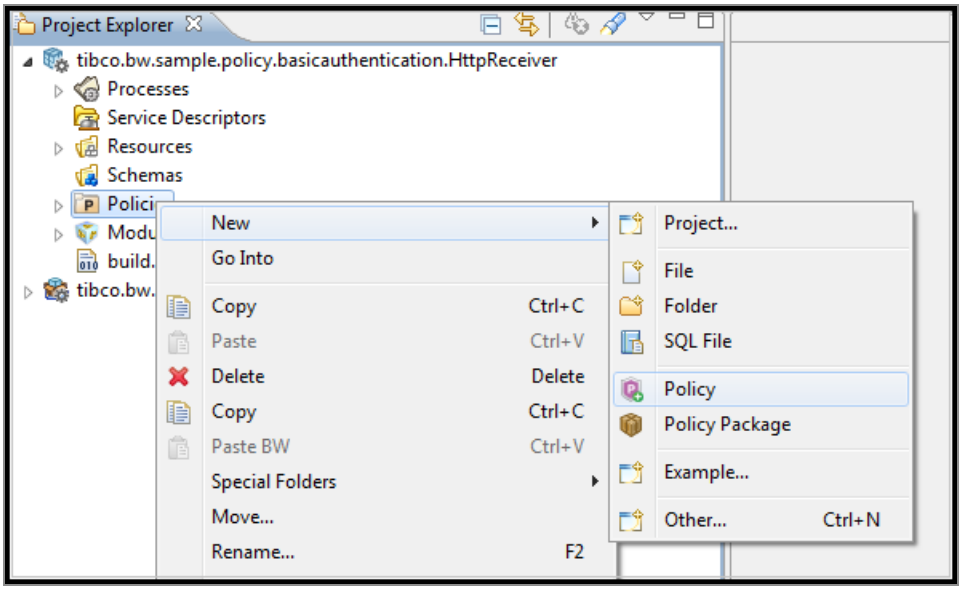
Implement the Basic Authentication policy to ensure user credentials in request messages are authenticated.

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.

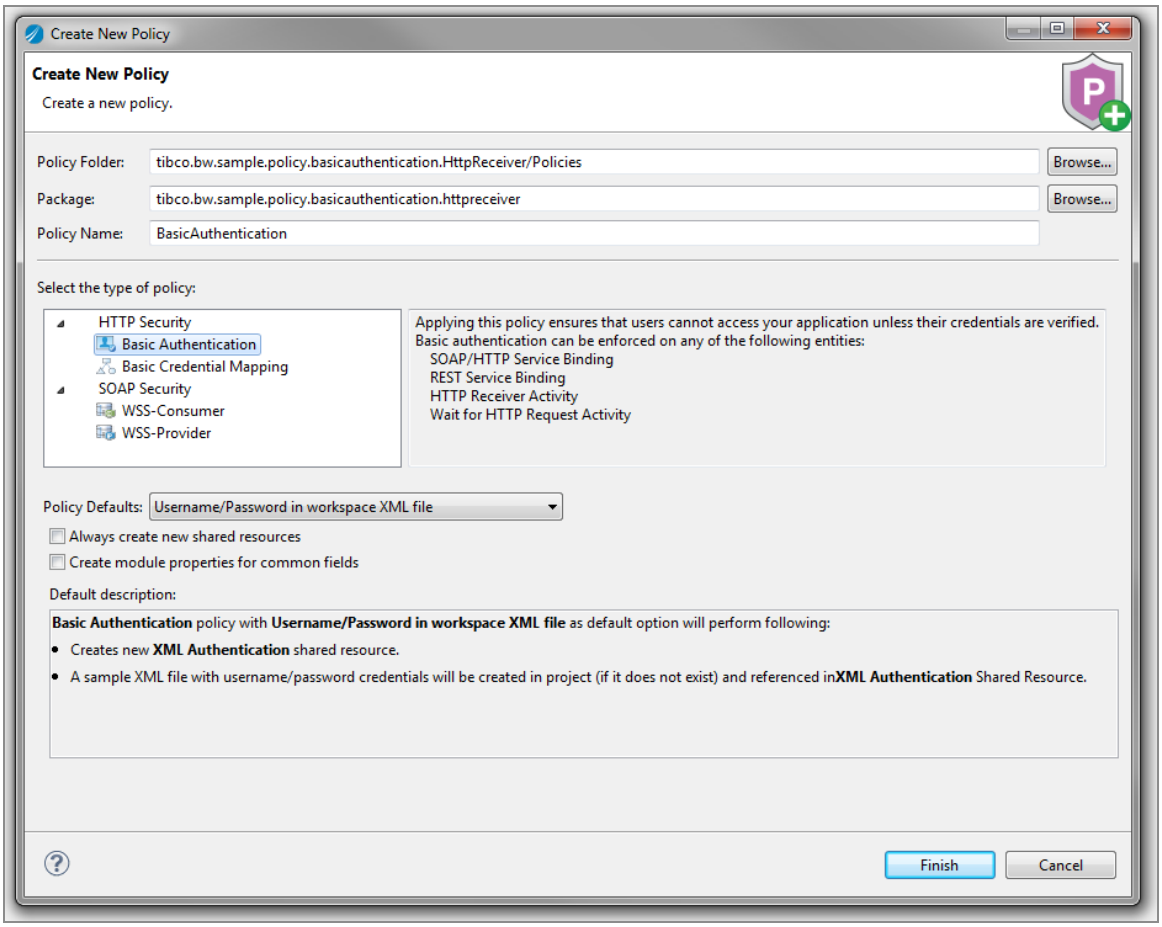
Setting Up a Policy with Resources

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 are located by 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:

> **Note:** 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 BusinessWorks Container Edition Bindings and Palettes Reference* guide.

- XML Authentication
- LDAP Authentication

For policy configuration details, see the topic "Basic Authentication", under "Policy Resources" in the *TIBCO BusinessWorks Container Edition 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

> **Note:** Credentials authenticated on this activity are not used for propagation during credential mapping.

- SOAP Service Binding
- REST Service Binding

For more information about how to enforce a policy on an activity, or a binding in your application, see Associating Policies.

Enforcing Basic Credential Mapping

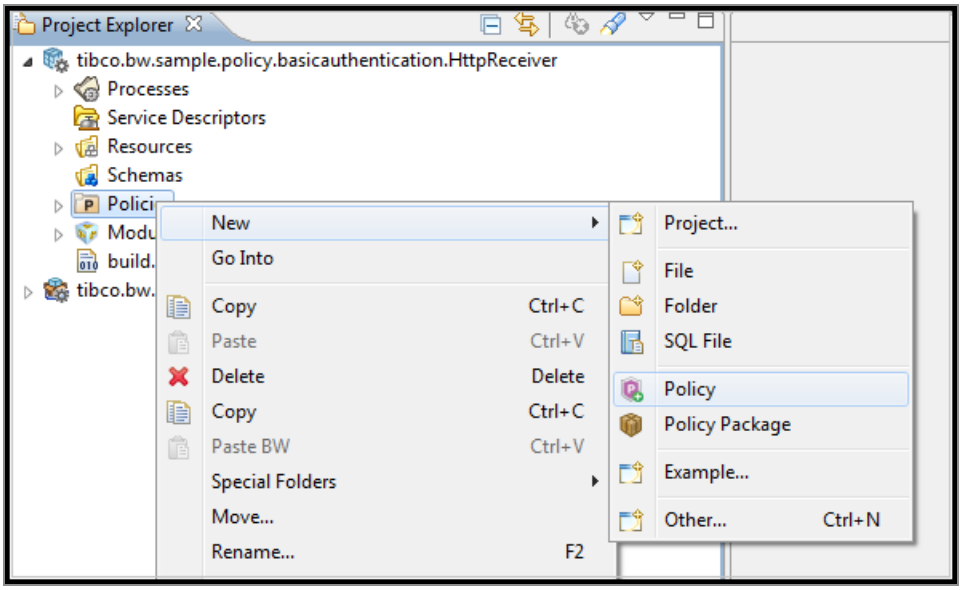
Map credentials for different types of users by enforcing the Basic Credential Mapping Policy.

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

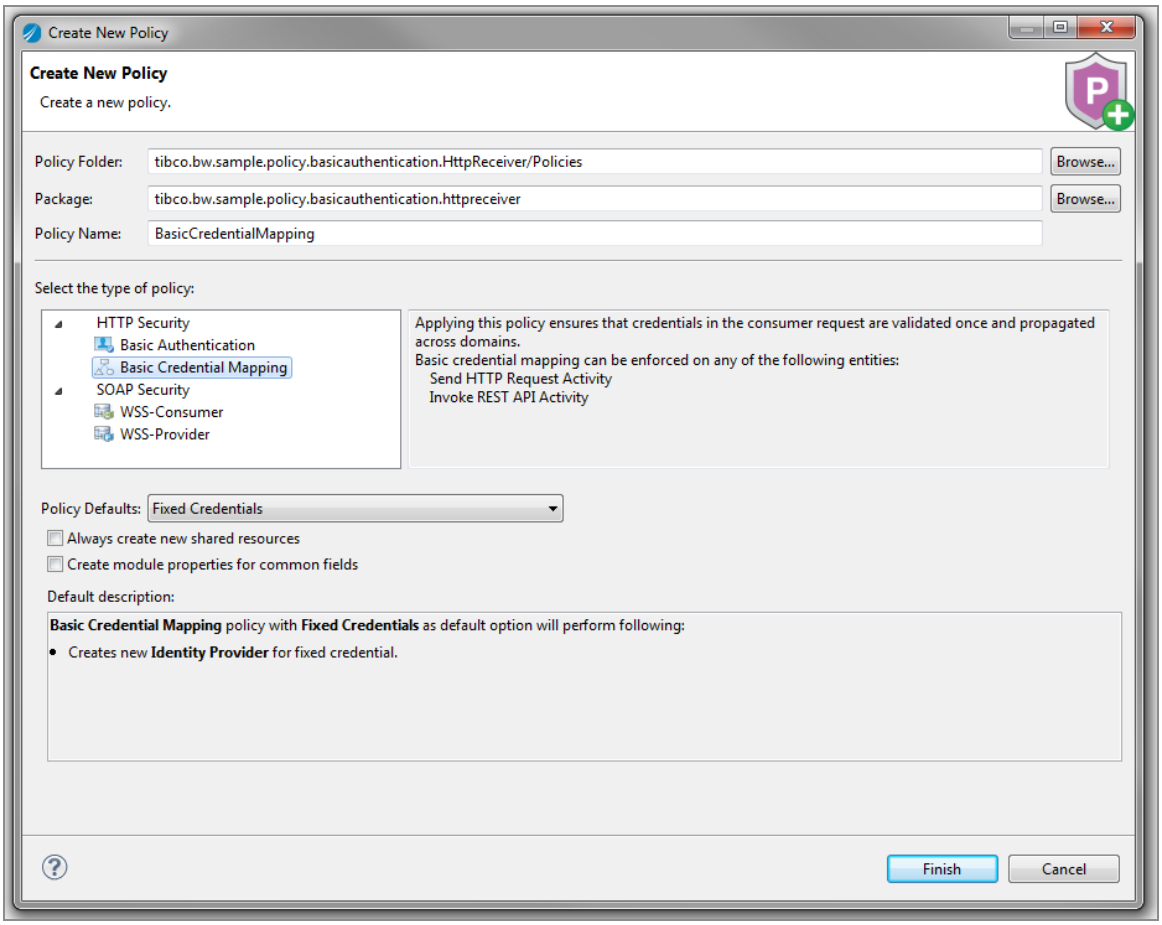
Setting Up a Policy with Resources

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 is displayed.



2. Specify the following values in the Create New Policy Window:
    - **Policy Folder**: Name of the folder where policies is 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:

Note: 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 BusinessWorks™ Container Edition Bindings and Palettes Reference* guide.

For more information about policy configuration details, see Basic Credential Mapping, under Policy Resources in the *TIBCO BusinessWorks Container Edition 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

Note: 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 more information 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

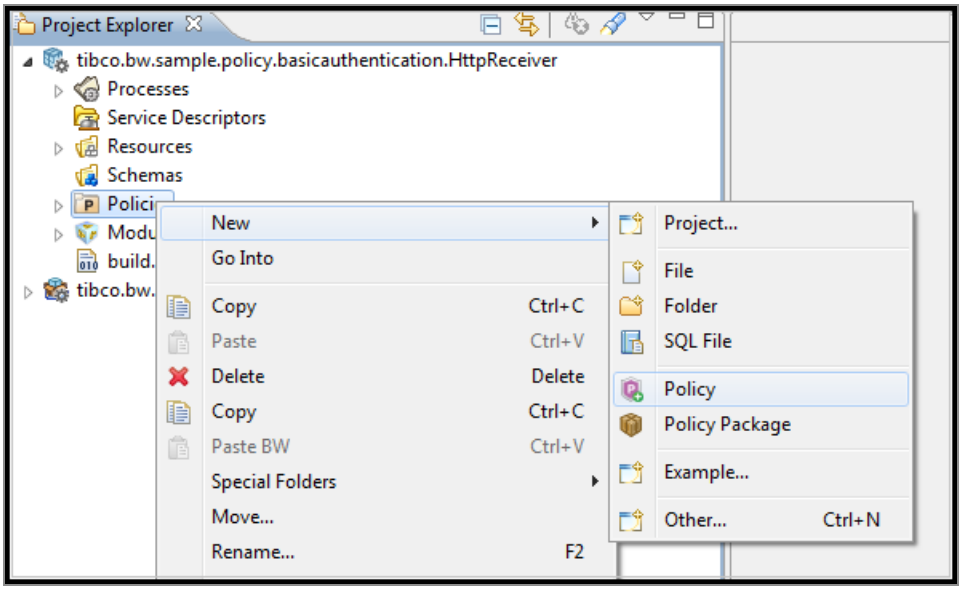
Enforce the WSS Consumer policy to ensure that the confidentiality, integrity, and the time stamp of a request remains secure.

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

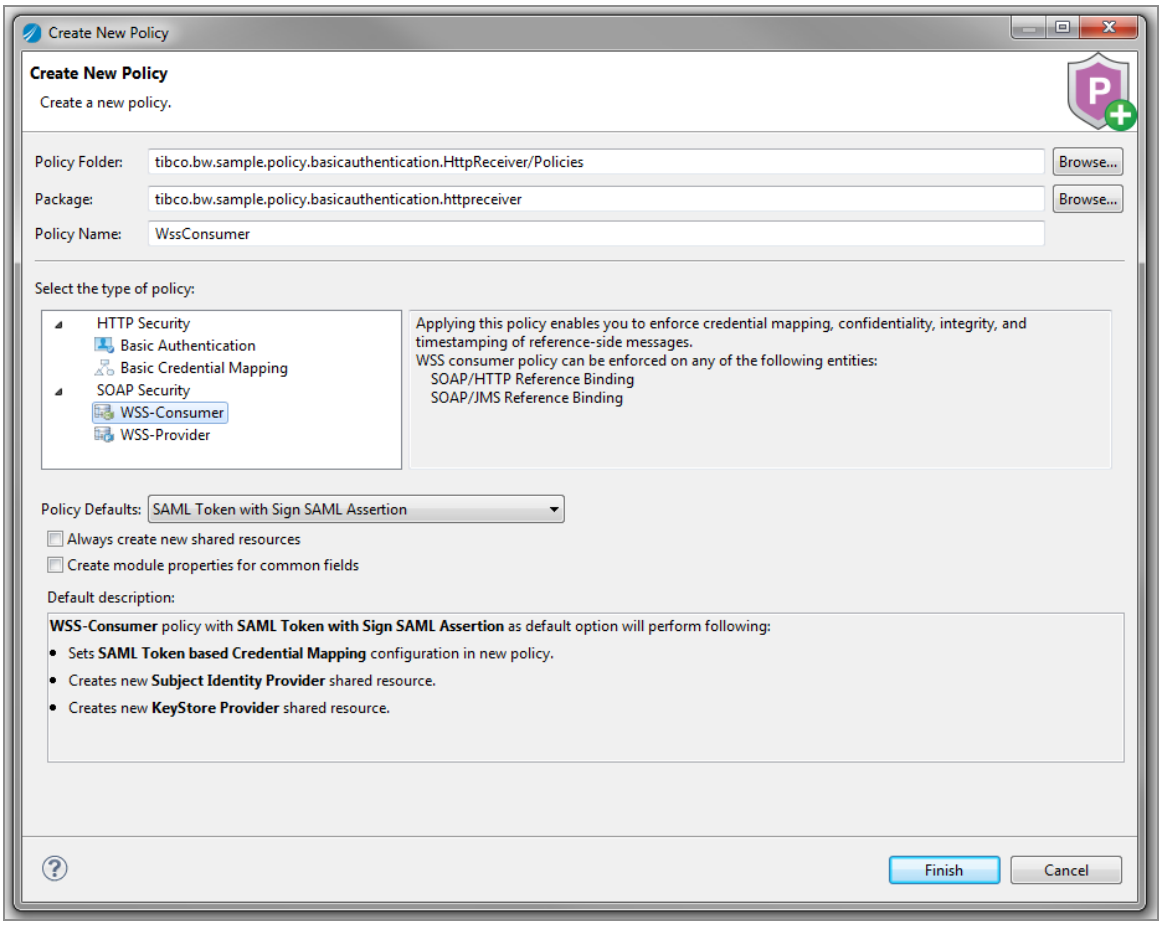
Setting Up a Policy with Resources

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**.



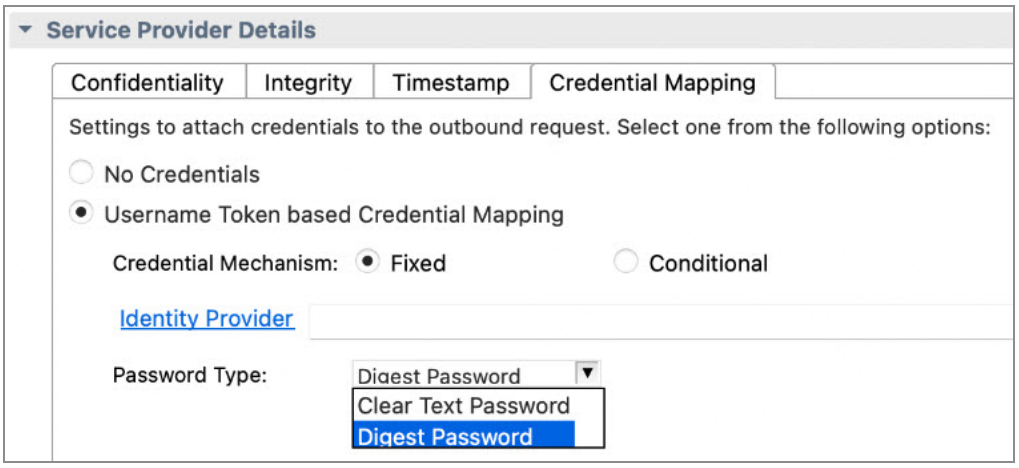
   The Policy Wizard is displayed.



2. Specify the following values in the Create New Policy Window:
    - **Policy Folder**: Name of the folder where policies are 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:

Note: 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:
    - An **Identity Provider** resource, with the default file name WSSConsumer_FixedIdentityProvider.userIdResource
    - A **Password Type**: The users have option to select the password types, **Clear Text Password** and **Digest Password**. By default, the **Clear Text Password** option is selected.



Note:

- Currently we are supporting the Password Digest feature at the reference side and not the service side.
- By default, the Password Type field is disabled until you add Identity Provider.

- **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
  - A **Password Type**: The users have option to select the password types, Clear Text Password and Digest Password. By default, the **Clear Text Password** option is selected.
- **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 BusinessWorks™ Container Edition 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 BusinessWorks Container Edition 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 more information about how to enforce a policy on a binding in your application, see Associating Policies.

Enforcing WSS Provider

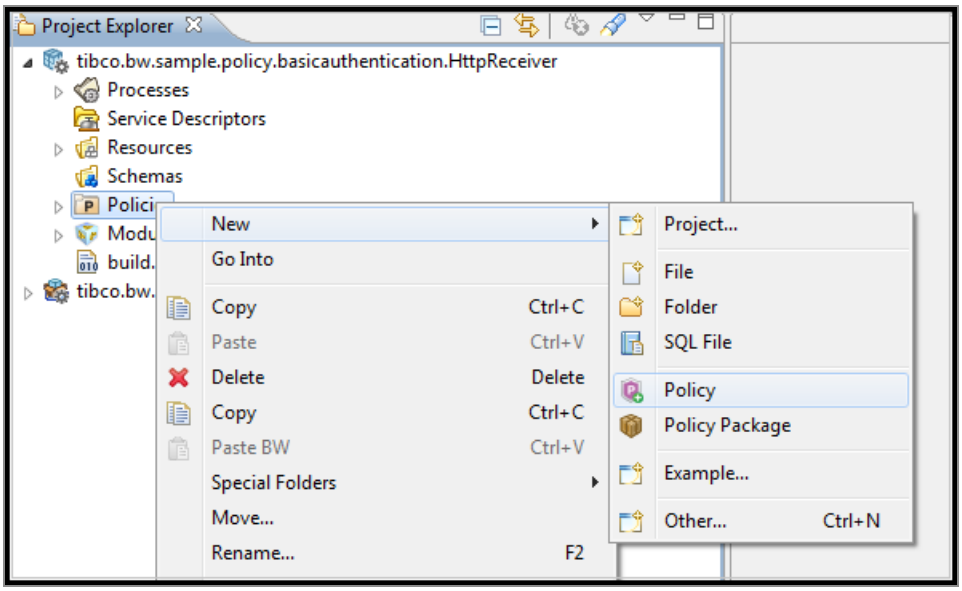
Use the WSS Provider policy to enforce authentication, confidentiality, integrity, and the time stamping of service-side messages.

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

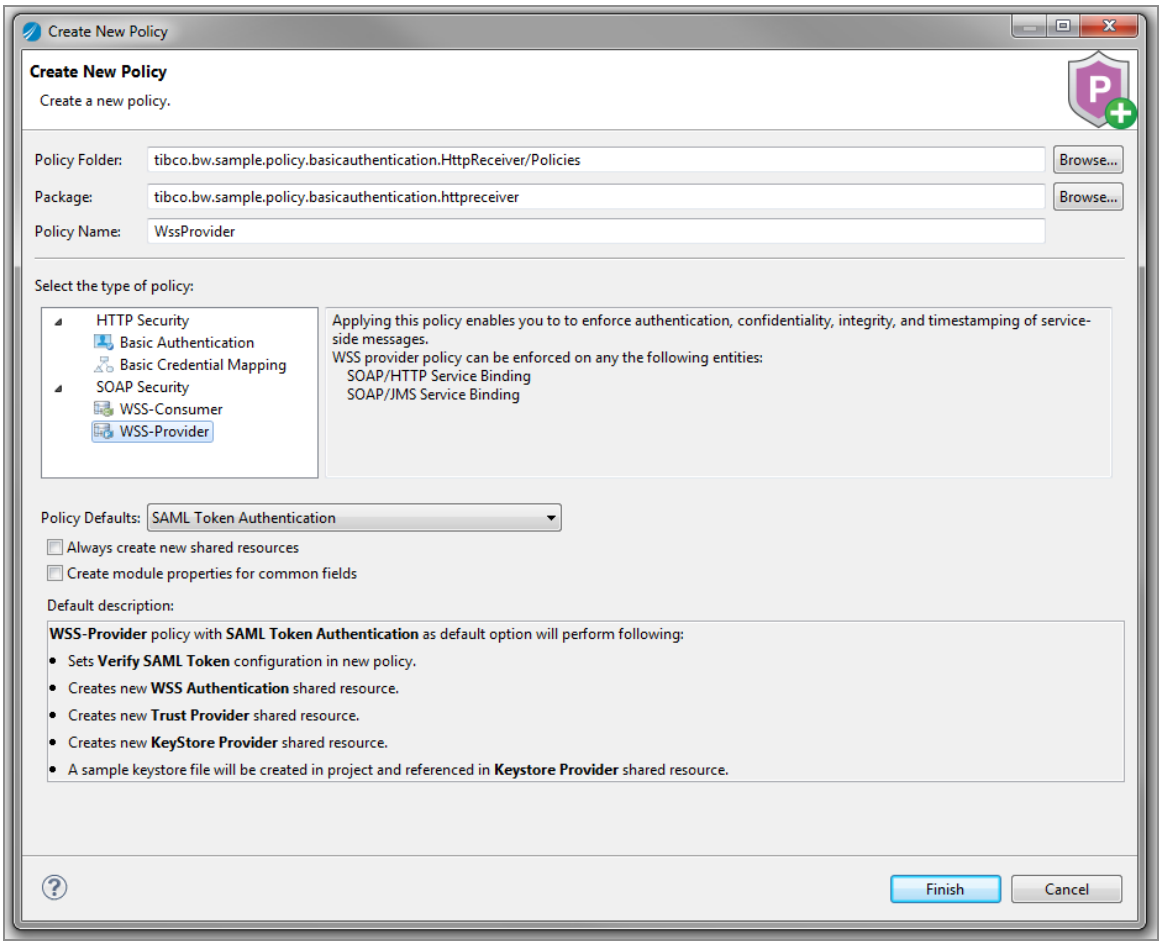
Setting Up a Policy with Resources

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 is displayed.



2. Specify the following values in the Create New Policy window:
    - **Policy Folder**: Name of the folder where policies are 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:

Note: 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 BusinessWorks Container Edition 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 BusinessWorks Container Edition 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 more information on how to enforce a policy on a binding in your application, see 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. This 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. The value of the property is false, when auto building is turned off.

Note: 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.

TIBCO BusinessWorks Container Edition 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.

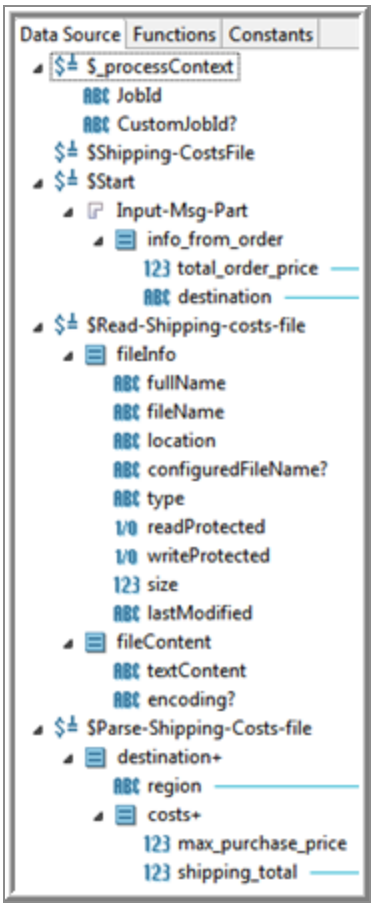
To use XPath in the product, you need to be familiar with the basic XPath concepts. However, to learn more about XPath when building complex expressions refer to the XPath specification from http://www.w3.org/.

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.

Note: A new preference, **Duplicate Target Namespace for XSD** has been added for validation. For more information, see Best Practices > Avoid XML Collisions in *TIBCO BusinessWorks™ Container Edition Application Development*.

Search Predicates

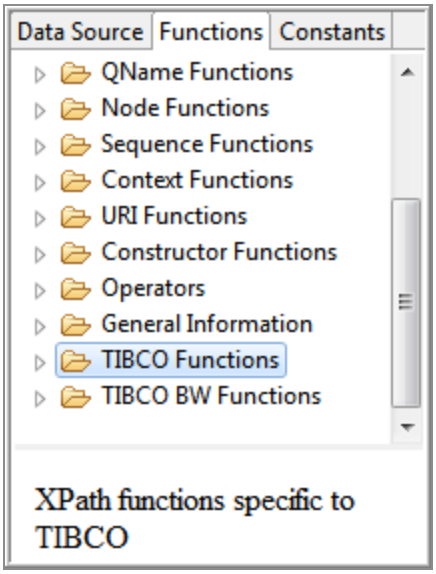
An XPath expression can have a search predicate. The search predicate is used to locate a specific element in a repeating schema element. For example, the $Parse-Shipping-Costs-file/destination/region item is a repeating element. To select only the first item in the repeating element, specify the following:

```
$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.

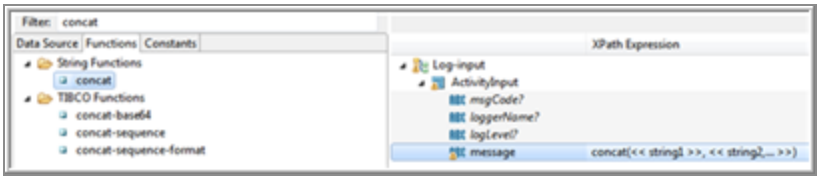


You can also use the Custom XPath Function Wizard to create your custom XPath function group. For more information, see **Creating Custom XPath Functions** in the *TIBCO BusinessWorks™ Container Edition Bindings and Palettes Reference.*

XPath Expression

The XPath expression is used to create 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.

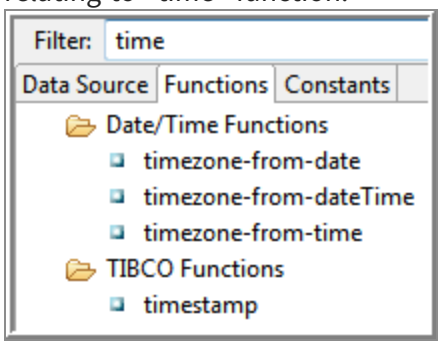


You can drag and drop schema elements from the **Data Source** tab into the function's placeholders.

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 **XPath Expression** field.
Functions	Displays the available XPath functions. These are categorized into groups and each function can be dragged from the function list into the **XPath Expression** field. 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. For more information about XPath functions, select XPath functions in XPath builder. The description of the function is displayed.

Elements	Description
Filter	Use this field for a refined function search in the mapper. Clicking the **Functions** tab displays the **Filter** field. For example, type "time" in the **Filter** field to obtain consolidated results relating to "time" function. 
Constants	Displays the constants available for use in XPath expressions. These are categorized into groups and each constant can be dragged from the constants list into the **XPath Expression** field. 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. Constants can also be used for inserting the following **TIBCO BW Predefined Module Properties**. • **Activity Name** - returns the name of the activity on which the module property is set. • **Application Name** - returns the application name. • **Application Version** - returns the version of the application specified in the **Version** field under the **Overview** tab of the application. • **Application Full Version**- returns the three digit version of the application in the form of <major>.<minor>.<micro>. • **Deployment Unit Name** - returns the ID of the application specified in the **ID** field under the **Overview** tab of the application. • **Deployment Unit Type** - returns the deployment unit type as

Elements	Description
	`application`. • **Deployment Unit Version** - returns the deployment unit version specified in the **Version** field under the **Overview** tab of the application. • **Domain Name** - returns the name of the domain in which the application is deployed. • **Module Name** - returns the name of the application module. • **Module Version** - returns the version of the module specified in the **Version** field under the **Overview** tab of the application module. • **Process Name** - returns the name of the process in which the module property is used. • **Process Stack** - returns the entire process path including the nested subprocesses, and the parent process. For example `main.Process/SubProcess1->sm.SubProcess1/SubProcess2->sm1.SubProcess2` • **Engine Name** - returns the name of the engine. By default, the name of the engine is `Main`.
Documentation Panel	Describes each selected function. On clicking a function on the **Function** 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. You can drag and drop items from the **Data Source** tab or the **Functions** tab to create the formula.

XPath Builder

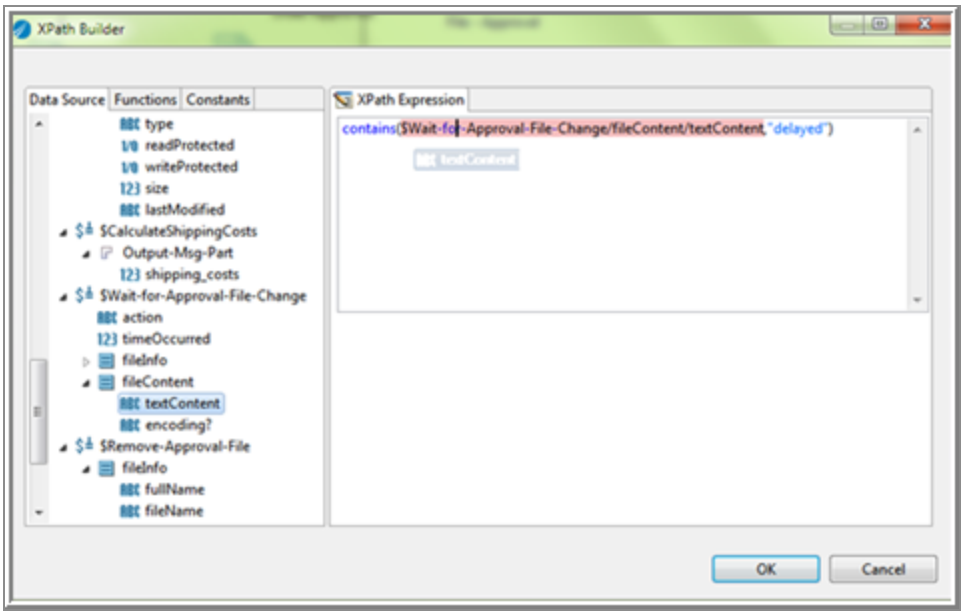
Using XPath Builder, you can drag and drop schema elements and XPath functions to create XPath expression.

Note: 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.

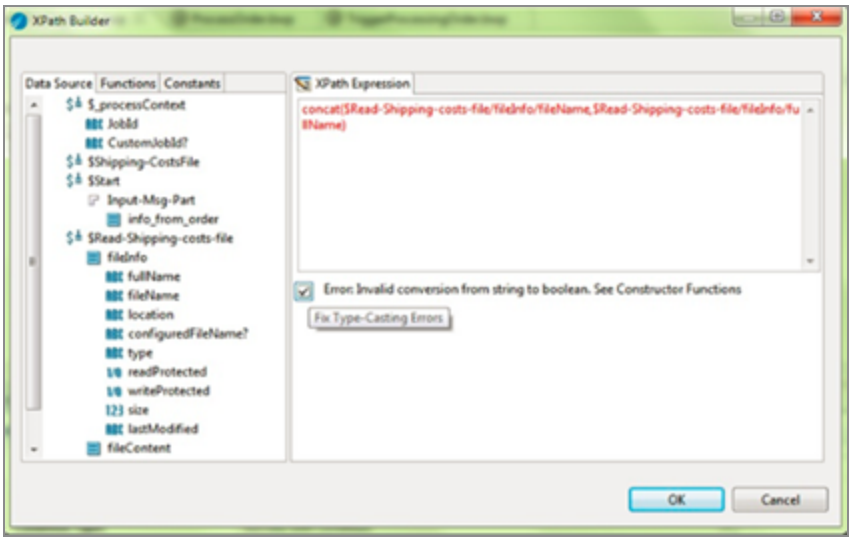
Note: XPath Builder is also available from **Sequence Key** field and **Custom Job Id** field of all process starter activities (such as **Timer**, **File Poller**, and so on).

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

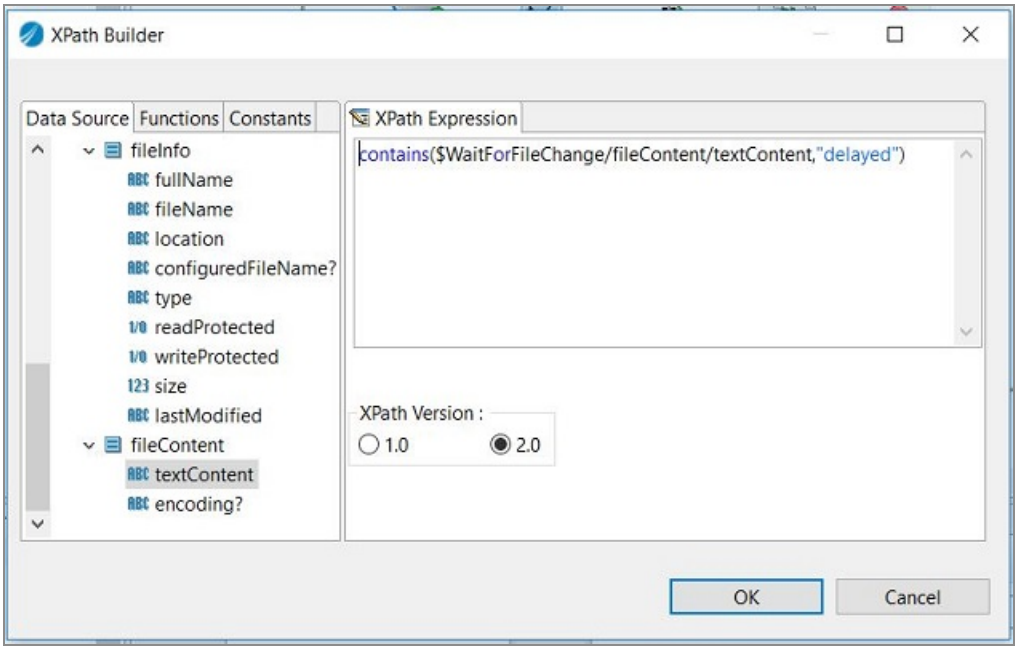
XPath Builder



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 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.

Note: 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.

Date and Time Functions

There are some functions in the XPath formula builder that allow you to parse or format strings that represent dates and times.

These functions are:

- format-dateTime(<<format>>, <<dateTime>>)
- format-date(<<format>>, <<date>>)
- format-time(<<format>>, <<time>>)
- parse-dateTime(<<format>>, <<string>>)
- parse-date(<<format>>, <<string>>)
- parse-time(<<format>>, <<string>>)

The format parameter of these functions is based on the format patterns available for the java.text.SimpleDateFormat Java class. In the format parameter, unquoted alphabetic characters from A to Z and a to z represent the components of the date or time string. You can include non-pattern alphabetic characters in the string by quoting the text with single quotes. To include a single quote, use ’’.

The following table describes the alphabetic characters and their associated presentation in a date or time string.

Formatting characters in date or time strings

character	Description	Example
G	Era. Four or more Gs return the full name of the era.	AD
y	year. yy returns two-digit year.	2003; 03
M	Month in a year. Three or more Ms return text name.	August; Aug; 08
w	Week in a year	48
W	Week in a month	3
D	Day in a year	253
d	Day in a month	25
F	Day of a week in a month	2
FNn	Day in a week	Friday
a	AM/PM marker. Four or more as return the full name.	AM
H	Hour in a day (0-23)	22
k	Hour in a day (1-24)	2
K	Hour in AM/PM (0-11)	10

Formatting characters in date or time strings(Continued)

character	Description	Example
h	Hour in AM/PM (1-12)	4
m	Minute in an hour	54
s	Second in a minute	48
S	Milliseconds	456
z	Time zone represented as a GMT offset.	GMT-08:00
Z	RFC 822 four-digit time zone format.	-0800
all other letters	Reserved	-

For any format pattern letter that returns a numeric value (for example, w, h, and m), the number of letters in the format pattern represents the minimum number of digits. For formatting functions, if the date or time has fewer digits than the number of pattern letters, the output is padded with zeros. For parsing functions, when the date or time has fewer digits than the number of characters in the format pattern, the extra characters are ignored, unless they are needed to determine the boundaries of adjacent fields.

The following table illustrates some example date and time format patterns and the resulting string.

Date-Time Pattern	Result
"yyy.MM.dd G ’at’ HH:mm:ss"	2003.3.11 AD at 09:43:56
"[FNn] [MNn] [D], [Y]"	Tuesday Mar 11, 2023
"hh ’o’’clock’ a, zzzz"	9 o’clock AM, GMT-8:00

"K:mm a"	0:08 PM
"yyMMddHHmmssZ"	010704120856-700

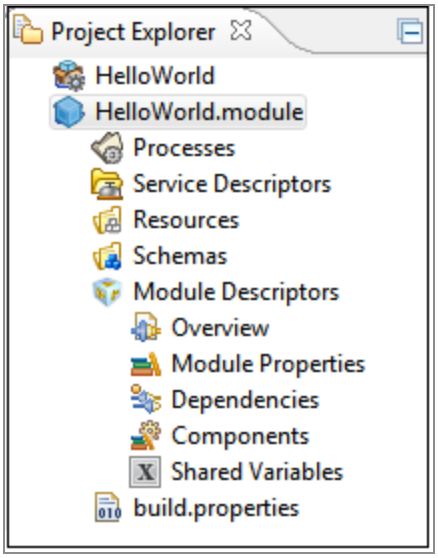
Developing a SOAP Service

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

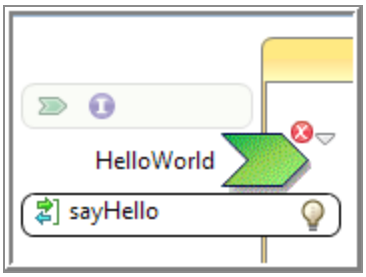
1. Click the process package, for example, "HelloWorld", and then click the **Create a new Business Works Process** icon.



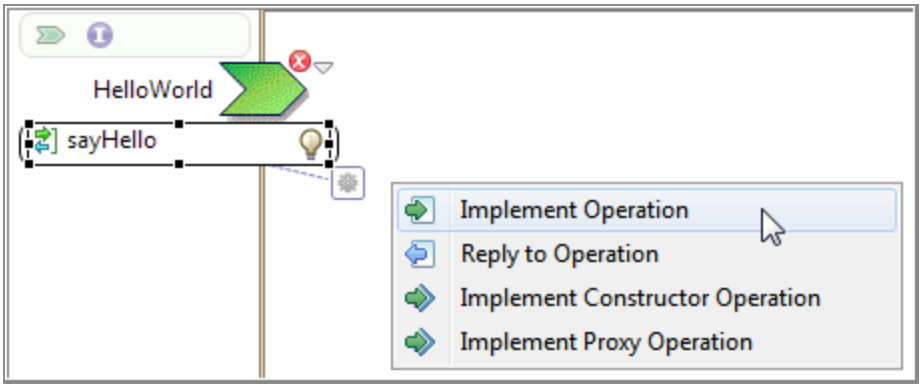
2. Select a process on which you want to add a service, and click the **Create Service** icon.

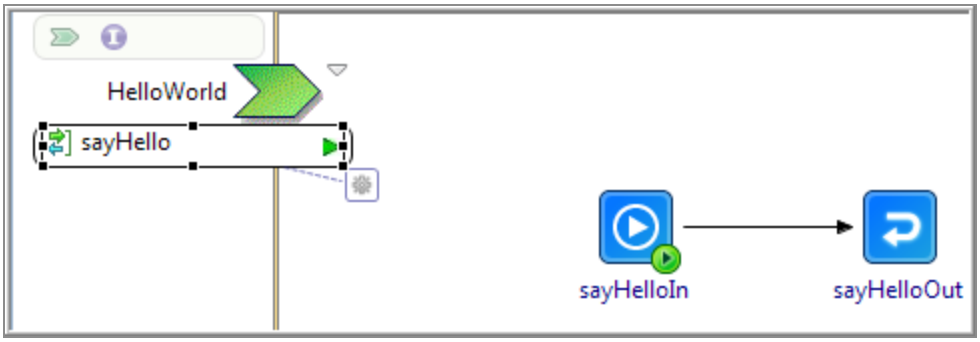
The New Service dialog is displayed.

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 the `sayhello` operation, and select **Implement Operation**.

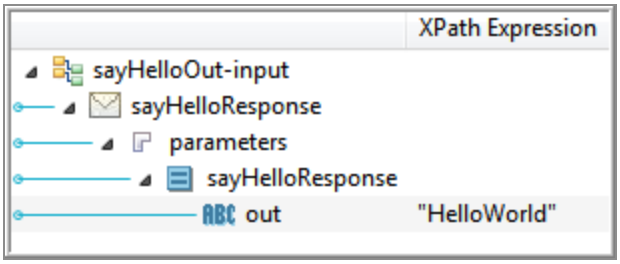




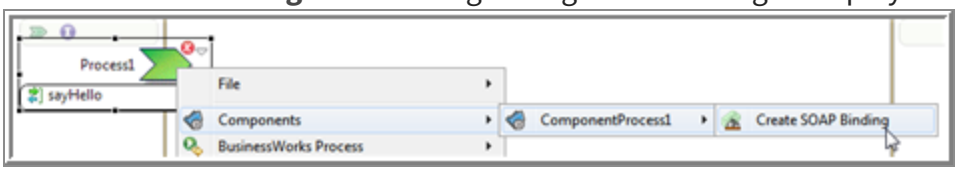
Choose **Implement Constructor Operation** option, if there are multiple operations in a port type.

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

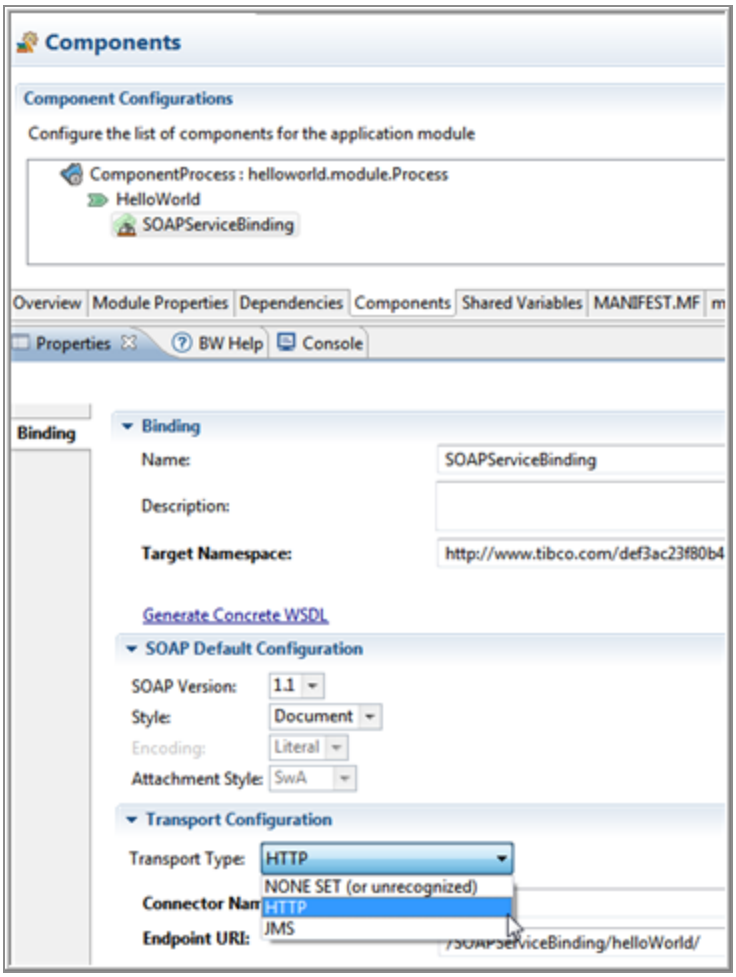
5. Click the **Reply** activity (`sayHelloOut`) and under the Properties view, click the **Input** tab. Configure Reply message.



6. Right-click the green chevron and select **Components > ComponentsProcess > Create SOAP Binding**. The Binding Configuration dialog is displayed.



7. To configure transport on the SOAPServiceBinding, select **HTTP** from the **Transport Type** dropdown list in **Transport Configuration**.



8. Click the **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.

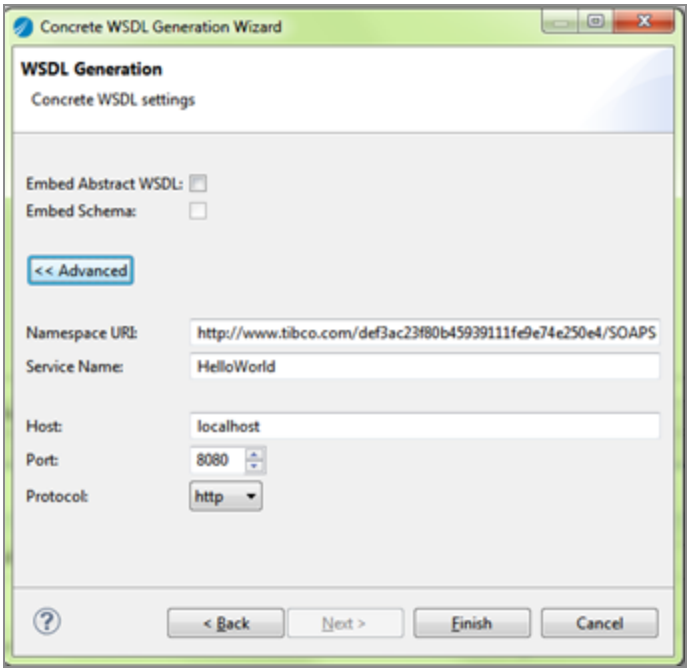
9. To generate the concrete WSDL of the SOAP service created in the above steps, click the **Generate Concrete WSDL** link.

10. Click **Workspace**. In the Folder Selection window, select the **Service Descriptor** folder of the current module and click **OK**.

    The Generate Concrete WSDL screen shows the specified location and the name of the WSDL.

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

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



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

The concrete WSDL is generated at the specified location.

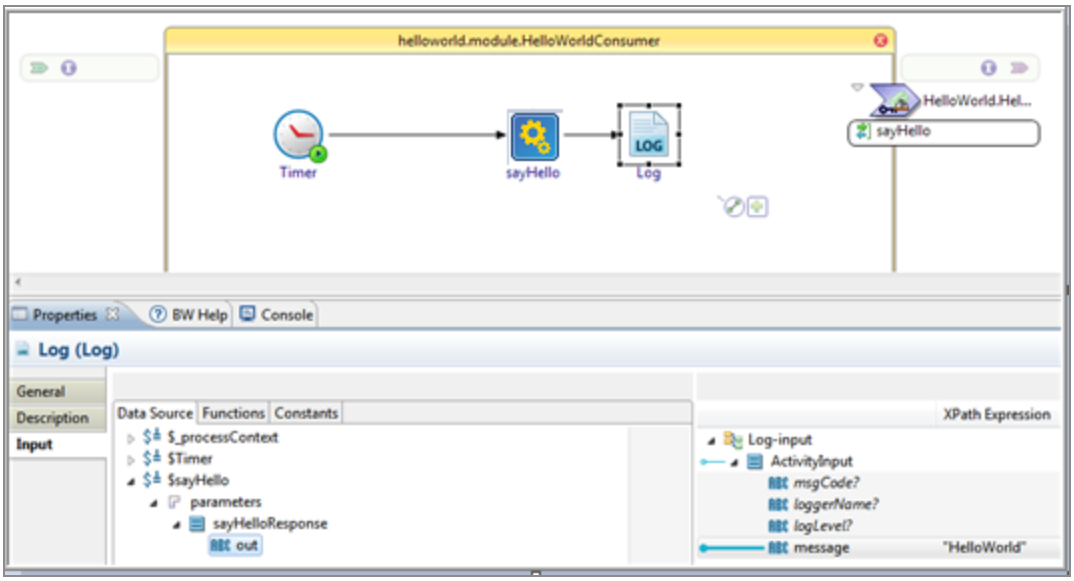
Consuming SOAP Services

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

Creating a Consumer for SOAP Service

Procedure

1. Click the process package, for example, "HelloWorld", and then click the **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 dropdown 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.
6. Configure the **Log** activity with a message.



   The SOAP Reference Binding is created.

7. Run the project.

Designing and Testing 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 BusinessWorks Container Edition currently allows the following HTTP methods to be performed on resources: GET, PUT, PATCH, 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.

Restrictions on XML Schema

This topic lists the restrictions on XML Schema.

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 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

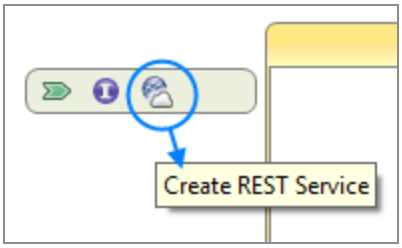
A REST service provider exposes the resources in a process definition that can be invoked by clients using one of the following operations- POST, GET, PUT, PATCH, and DELETE.

Before you begin

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 BusinessWorks Container Edition 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.

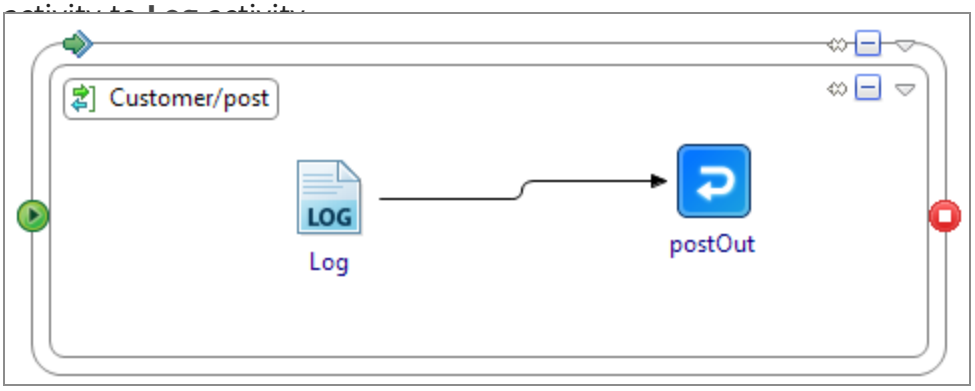
- `Summary` about the new REST service.
- `Resource Service Path`: Specifies the URI that is used to access the REST service.
- `Type of Resource`: Select if the service works on a single resource or a collection.
- `Operations`: By default, the GET operation is selected. Select or deselect the operations 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.

   Note: 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 BusinessWorks Container Edition Getting Started* guide.

Discovering API Models from TIBCO Business Studio for BusinessWorks

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

Before you begin

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 **API Explorer** tab, click the **View Menu** downward-facing triangle icon (▽) and select **Settings**.

   The Settings dialog is displayed.

   The registries for the TIBCO BusinessWorks Container Edition - 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 appear in the API Explorer:

- **API Presentation** - specifies how the APIs 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 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 are 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 checkboxes 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 are 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 show you its version, the resource path, and the operations to perform on that resource.

> **Note:** 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.

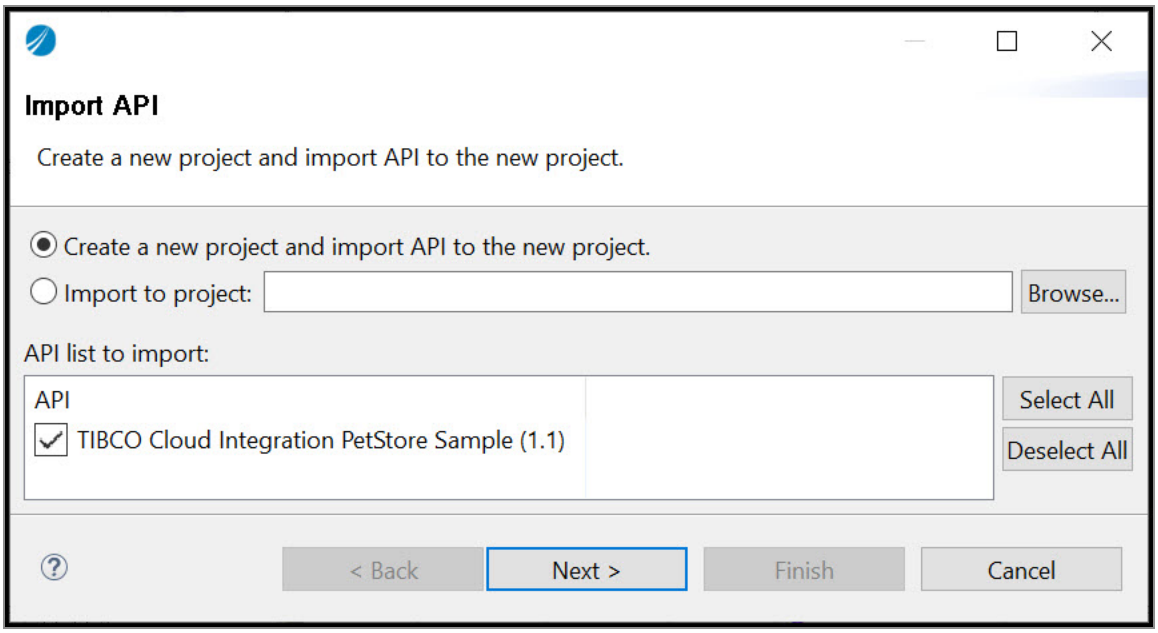
Importing an API Model into your Workspace

The APIs that are discovered from local and remote servers are displayed in the **API Explorer** tab of the TIBCO BusinessWorks Container Edition. 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.

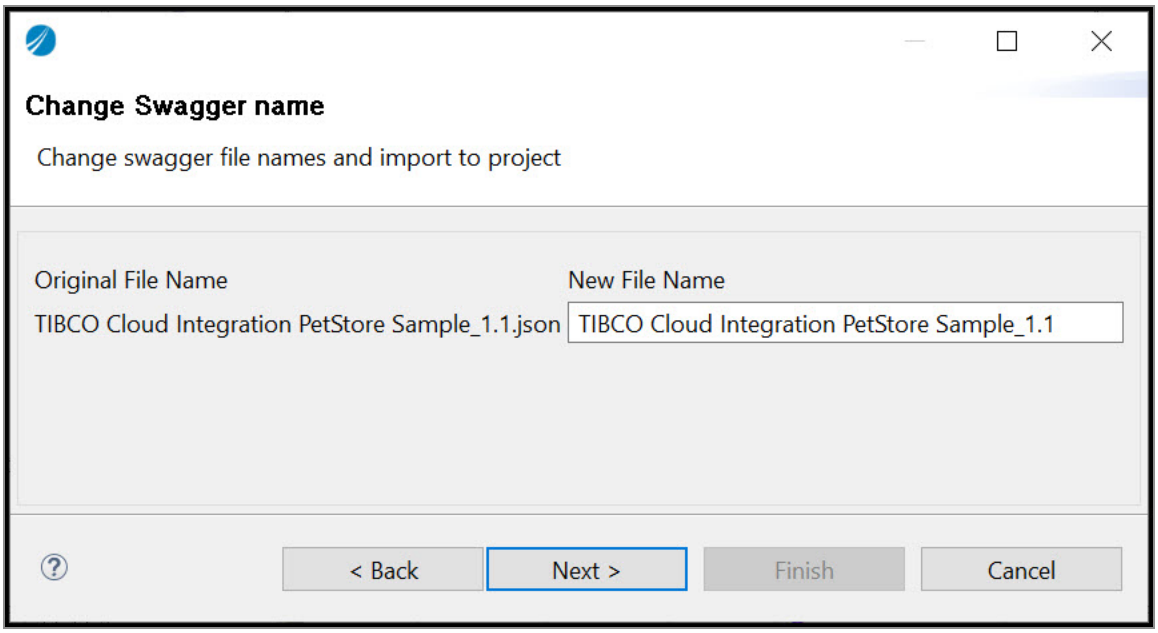


Every API you selected in the **API Explorer** is listed in this dialog. If an API has multiple versions, all versions are listed. By default, all APIs listed here are selected. You can deselect APIs that you do not want to import by clearing its checkbox.

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

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

The Change Swagger name dialog opens.



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

The New BusinessWorks Application Module dialog 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.

> **Note:** 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 File

TIBCO Business Studio for BusinessWorks supports the creation of an XML schema for an imported Swagger 2.0 or a Swagger 3.0 file.

You can create an XML schema for a Swagger 2.0 or a Swagger 3.0 files in one of two ways described below.

Before you begin

A Swagger 2.0 or a Swagger 3.0 file must exist in the **Service Descriptors** folder of the project. Make sure to import the Swagger file into the **Service Descriptors** folder before you follow these steps:

Procedure

1. 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 contains only those elements that are common to both files. It does 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 appears 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.

   > **Note:** 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 apply on the enhanced Java development tooling available in TIBCO Business Studio for BusinessWorks to develop a Java application. A project with Java nature contains a default source folder for Java classes, `src`, in addition to other folders.

Note: 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 an application module, select the **Use Java configuration** checkbox.
- 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 TIBCO BusinessWorks Container Edition Application

An TIBCO BusinessWorks Container Edition 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 TIBCO BusinessWorks Container Edition 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 TIBCO BusinessWorks Container Edition 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 the 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.

Before you begin

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

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){
   }
   ```

   Note: You can invoke static or non-static methods using **Java Invoke** activity. For more information about Java Invoke activity, see the *TIBCO BusinessWorks Container Edition Bindings and Palettes Reference*.
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 TIBCO BusinessWorks Container Edition Module Properties in a user-defined Java code referenced in Java Global Instance, follow these steps:

Procedure

1. In the TIBCO BusinessWorks Container Edition module, specify a dependency on the package "**com.tibco.bw.palette.shared.java**" using **Import-Package**.
    a. Double-click **Dependencies** located under **TIBCO BusinessWorks Container Edition 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 TIBCO BusinessWorks Container Edition module properties.

Accessing Module Properties from Java Invoke Activity

You can access the TIBCO BusinessWorks Container Editionmodule properties and Java system properties from the user-defined code invoked from the Java Invoke activity and Java Event Source.

Procedure

1. Under the TIBCO BusinessWorks Container Edition 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". For more information on how to use these methods, see *API Reference* .

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

Procedure

1. 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

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 checkbox and browse to select the location to be used.
3. **Version**: Version of the application.
4. **Create Application Module**: Selected by default to create an application module with the specified name. Clear the checkbox if you do not want to create an application module.
5. 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

Application properties have the largest scope of all properties. Application properties are useful when you want to share the same value for an existing module property across multiple processes in an application or between an application module and a shared module. You can either promote an existing module property to the application level or create a new application property.

Creating an Application with Multiple Profiles

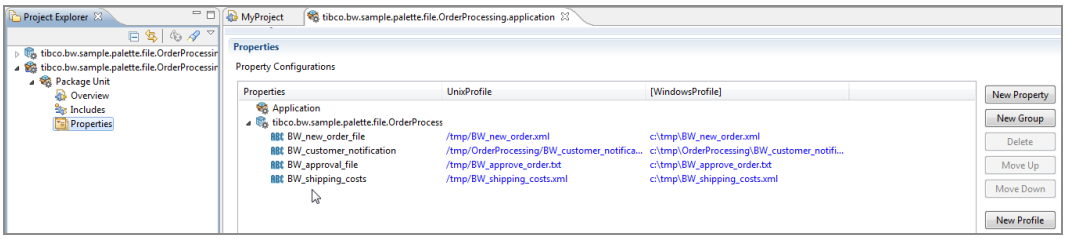
You can define multiple profiles when creating an application in TIBCO Business Studio for BusinessWorks.

A *profile* is a collection of module and application properties that an application uses. When an application is deployed with different properties, different profiles are available for each deployment. For example, you can create a Windows profile for an application that runs on a Windows machine and another for the same application running on a UNIX machine.

Before you begin

An application is created with profiles using TIBCO Business Studio for BusinessWorks. For more information about creating applications see the *TIBCO BusinessWorks Container Edition Application Development* guide. The following screenshot shows an application with a profile for Windows and another for UNIX. Each profile has a set of defined properties and values. The values use the appropriate operating system syntax to point to the files in the file system. The files are created and maintained outside of TIBCO Business Studio for BusinessWorks.

Application Profiles



Follow these steps to create an application profile:

Procedure

1. Start TIBCO Business Studio for BusinessWorks and open an application.
2. Expand the application and double-click **Properties** under **Package Unit**.

   This displays the **Properties** pane in the **Process Editor**.
3. Click the **New Profile** button to add a new profile.
4. In the **Create New Profile** window, enter a name for the new profile. For example, enter WindowsProfile and click **OK**.

   The WindowsProfile gets created and available to the right of the **[default]** column in the Properties pane.
5. Double-click the field under the profile that corresponds to a property, and enter a value for the property.
6. Save the project.

   You can create multiple profiles as needed.

Setting the Default Application Profile

When you have multiple application profiles for an application, you must select one of the profiles as the default profile. The default profile is indicated by the square brackets [ ] around the profile name.
Follow these steps to select a default application profile:

Procedure

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 set as the default application profile.

   The profile gets selected.
3. Click **Set as Default**.
4. Click **Yes** when prompted to confirm whether you want to set the selected profile as the default profile.

   The profile name is surrounded by square brackets.

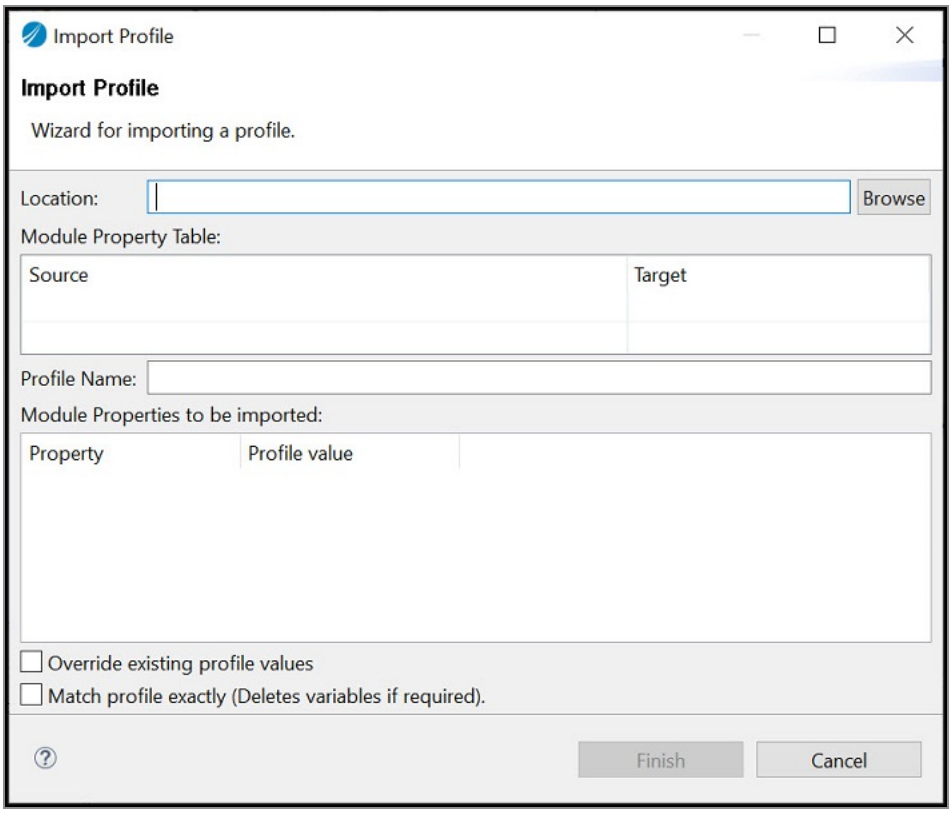
Importing an Application Profile

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.
2. Click **Import Profile**.

   The Import Profile dialog opens.



3. Use the **Browse** button to browse to a location of the `<profile-name>.substvar` profile file you want to import.
4. To override existing profile values with the values in the imported profile, select the **Override existing profile values** checkbox. To keep values from the imported profile only, select the **Match profile exactly (Deletes variables if required)** checkbox.
5. Click **Finish**.

   The imported profile is visible under **Property Configuration**.

Important:

- TIBCO Business Studio for BusinessWorks deletes the module properties and profile variables if they are removed from the profile file, which is being imported, by selecting the **Match profile exactly (Deletes variables if required)** checkbox.

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.

Do the following to export an application profile as a substvar and properties file:

1. Click the **project.application**, expand **Package Unit** folder and double-click **Properties** to open the Properties page.
2. In the Properties view, select the profile, and click the **Export Profile** button. The Export Profile wizard opens.
3. Select the properties to be exported and use the **Browse** button to browse to a location where you want to download the profile file as a substvar or properties file.
    a. To export as substvar file, select the checkbox **Export as substvar file** and browse the location to export the substvar file.

    A `<ProfileName>.substvar` file is created in the location specified. You can now import this profile file into another application.

    b. To export as a properties file, select the checkbox **Export as properties file** and browse the location to export the properties file. The properties that are selected while exporting the profile are generated as `key-value` pairs in the <ApplicationName.application>-<ProfileName>.properties file. The keys in the Properties file is generated using the names of the properties that are exported. For example, `//<ApplicationName>//<PropertyName>=<value>`.

    > **Note:** In the properties view, the values of the properties in the profile can be anything, when the profile is exported as a properties file it changes to `#//<ApplicationName>//<PropertyName>#`

    c. To export the application properties file with property names separated by a dot (.), select the **Use dot ('.') as a separator** checkbox. This checkbox is enabled only when the **Export as properties file** checkbox is selected. The properties that are selected while exporting the profile are generated as `key-value` pairs in the <ApplicationName.application>-<ProfileName>.properties file. The keys in the Properties file is generated using the names of the properties that are exported. For example, `<ApplicationName>.<PropertyName>=<value>`

Note:
- In the Properties view, the values of the properties in the profile can be anything, when the profile is exported as a properties file it changes to `#<ApplicationName>.<PropertyName>#`
- TIBCO recommends to use the dot (.) separator for exporting properties file while configuring the Kubernetes Configmap with TIBCO BusinessWorks™ Container Edition.

d. To export the application properties file with property names separated by a underscore (_), select the **Use underscore ('_') as a separator** checkbox. This checkbox is enabled only when the **Export as properties file** checkbox is selected. The properties that are selected while exporting the profile are generated as `key-value` pairs in the <ApplicationName.application>-<ProfileName>.properties file. The keys in the Properties file is generated using the names of the properties that are exported. For example, `<ApplicationName>_<PropertyName>=<value>`

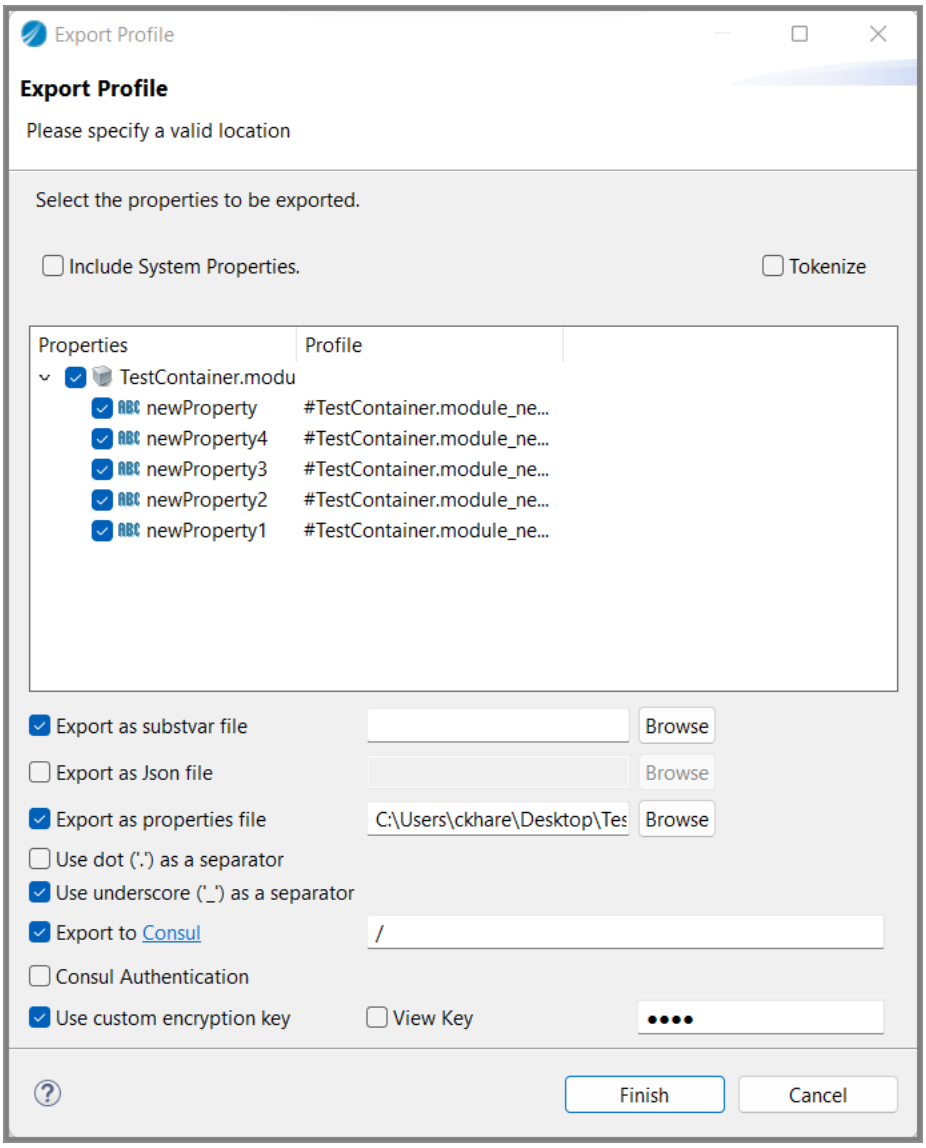
Note:
- The **Use underscore ('_') as a separator** and **Use dot ('.') as a separator** checkboxes cannot be used together. It is a best practice to use only one checkbox at a time.
- In the Properties view, the values of the properties in the profile can be anything, when the profile is exported as a properties file it changes to `#<ApplicationName>_<PropertyName>#`

e. To use custom encryption key to export to Consul, select the **Use custom encryption key** checkbox. To view the custom encryption key in a plain text format, select the **View Key** checbox. The custom encryption key is only applicable to password type module properties.

Note:

- While running the application, use the CUSTOM_ENCRYPTION_KEY environment variable. The value of this environment variable must be the custom key used while exporting the profile to Consul.
- It is recommended to update or rotate custom key on a regular basis following the ideal security standards.

4. Click **Finish**.



The following table lists the values of five different types of properties if the default values are not provided for all the properties and exported as properties file.

Data Type	Property Name	Values Before Export	Values After Export	Values After Export in properties file
		Default	Default	
String	newProperty5		#//test//newProperty5#	newProperty5
Password	newProperty4		#//test//newProperty4#	<encrypted value of the password>
Integer	newProperty3	0	#//test//newProperty3#	0
Long	newProperty2	0	#//test//newProperty2#	0
Boolean	newProperty1	false	#//test//newProperty1#	false

The following table lists the values of five different types of properties if the default values are provided for all the properties and exported as properties file.

Data Type	Property Name	Values Before Export	Values After export	Values After Export in properties file
		Default	Default	
String	newProperty5	TIBCO	#//test//newProperty5#	TIBCO
Password	newProperty4	***	#//test//newProperty4#	<encrypted value of the password>
Integer	newProperty3	1	#//test//newProperty3#	1

Data Type	Property Name	Values Before Export	Values After export	Values After Export in properties file
Long	newProperty2	12345	#//test//newProperty2#	12345
Boolean	newProperty1	true	#//test//newProperty1#	true

Note: For String type property in group, the values after export in properties file is `//test///newGroup/newGroup1/newProperty5=TIBCO`

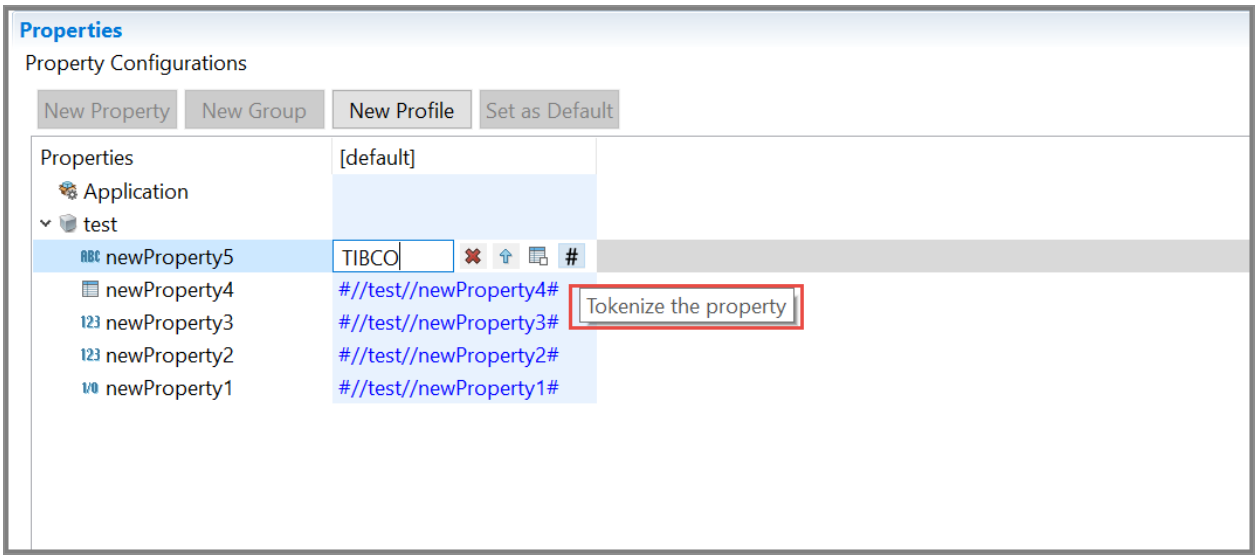
The following table lists the values of five different types of properties if the default values are provided for all the properties and exported as properties file using (.) dot as a separator.

Data Type	Property Name	Values Before Export	Values After Export	Values After Export in properties file
		Default	**Default**	
String	newProperty5	TIBCO	#test.newProperty5#	TIBCO
Password	newProperty4	***	#test.newProperty4#	<encrypted value of the password>
Integer	newProperty3	1	#test.newProperty3#	1
Long	newProperty2	12345	#test.newProperty2#	12345
Boolean	newProperty1	true	#test.newProperty1#	true

Note: For String type property in group, the values after export in properties file is `test.newGroup.newGroup1.Property5=TIBCO`

Tokenizing Application Properties

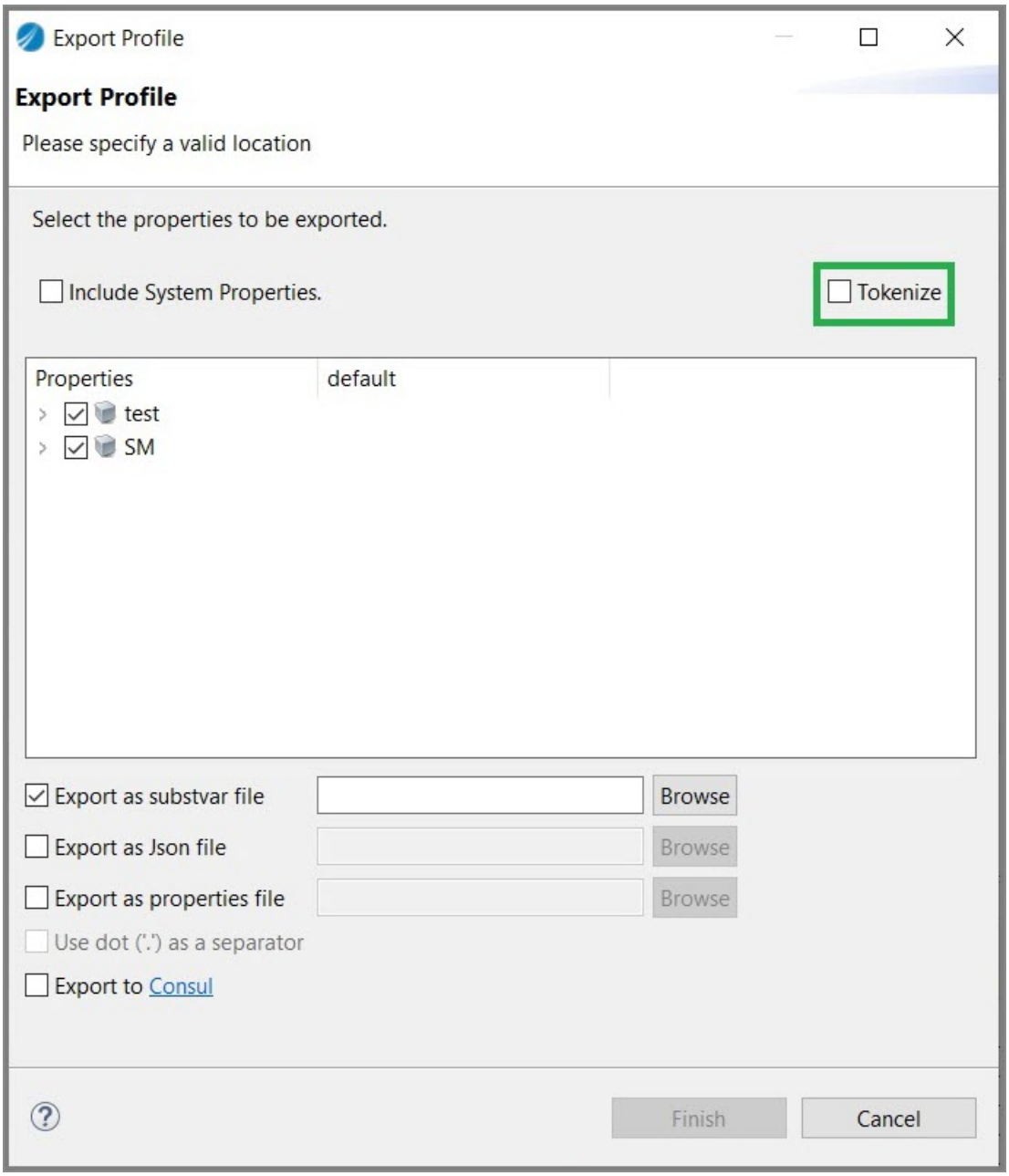
To tokenize application properties, a new button **Tokenize the property** is added in the Properties view for application properties. Tokenizing supports properties of type boolean, string, integer, long, and password.



After tokenization, the property value is set in the format //<ApplicationName>//<PropertyName>. Once the user tokenizes a property, the original default value for the property is lost.

Note: Tokenize the property button is available only for applications with Deployment Target set as Container.

By default, the **Tokenize** checkbox is unchecked. Select the **Tokenize** checkbox to auto-tokenize property value for the **Export as properties file** profile.



The following table lists the values of five different types of properties if the default tokenized values are not provided for all the properties and exported as properties file

Data Type	Property Name	Values Before Tokenization	Values After Tokenization/ Values Before Export	Values after Export	Values after Export in the properties file
		Default	Default	Default	Default
String	newProperty5		#//test//newProperty5#	#//test//newProperty5#	newProperty5
Password	newProperty4		#//test//newProperty4#	#//test//newProperty4#	a String: PASSWORD
Integer	newProperty3	0	#//test//newProperty3#	#//test//newProperty3#	0
Long	newProperty2	0	#//test//newProperty2#	#//test//newProperty2#	0
Boolean	newProperty1	false	#//test//newProperty1#	#//test//newProperty1#	false

The following table lists the values of five different types of properties if the default tokenized values are provided for all the properties and exported as properties file

Data Type	Property Name	Values Before Tokenization	Values After Tokenization/ Values Before Export	Values after Export	Values after Export in the properties file
		Default	Default	Default	Default
String	newProperty5	TIBCO	#//test//newProperty5#	#//test//newProperty5#	newProperty5
Password	newProperty4	***	#//test//newProperty4#	#//test//newProperty4#	a String: PASSWORD
Integer	newProperty3	1	#//test//newProperty3#	#//test//newProperty3#	0
Long	newProperty2	12345	#//test//newProperty2#	#//test//newProperty2#	0
Boolean	newProperty1	true	#//test//newProperty1#	#//test//newProperty1#	false

Note: For String type property in group, the values after export in properties file is `//test///newGroup/newGroup1/newProperty5=newGroup.newGroup.newProperty5`

The following table lists the values of five different types of properties if the default tokenized values are provided for all the properties and exported as properties file using dot (.)as a separator.

Data Type	Property Name	Values Before Tokenization	Values After Tokenization/ Values Before Export	Values after Export	Values after Export in the properties file
		Default	**Default**	**Default**	**Default**
String	newProperty5	TIBCO	#//test//newProperty5#	#test.newProperty5#	newProperty5
Password	newProperty4	***	#//test//newProperty4#	#test.newProperty4#	a String: PASSWORD
Integer	newProperty3	1	#//test//newProperty3#	#test.newProperty3#	0
Long	newProperty2	12345	#//test//newProperty2#	#test.newProperty2#	0
Boolean	newProperty1	true	#//test//newProperty1#	#test.newProperty1#	false

Note: For String type property in group, the values after export in properties file is `test.newGroup.newGroup1.Property5=newGroup.newGroup.newProperty5`

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 run time as it contains all the bundles and metadata that is required to deploy and run the application.

> **Note:** 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 TIBCO BusinessWorks Container Edition processes in SVG format.

There are multiple ways to create a deployment artifact:

- From the Project Explorer view in TIBCO Business Studio for BusinessWorks, open **Project.application > Overview** and click **Export Application for Deployment** link. 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** checkbox. 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`.

Note: When importing projects created in a version of the software that is lower than TIBCO BusinessWorks Container Edition 2.3.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 on the **Problems** tab.
- Warning: A warning is displayed on the **Problems** tab.
- Ignore: The validation error is not displayed on the **Problems** tab.

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 Cloud Foundry.

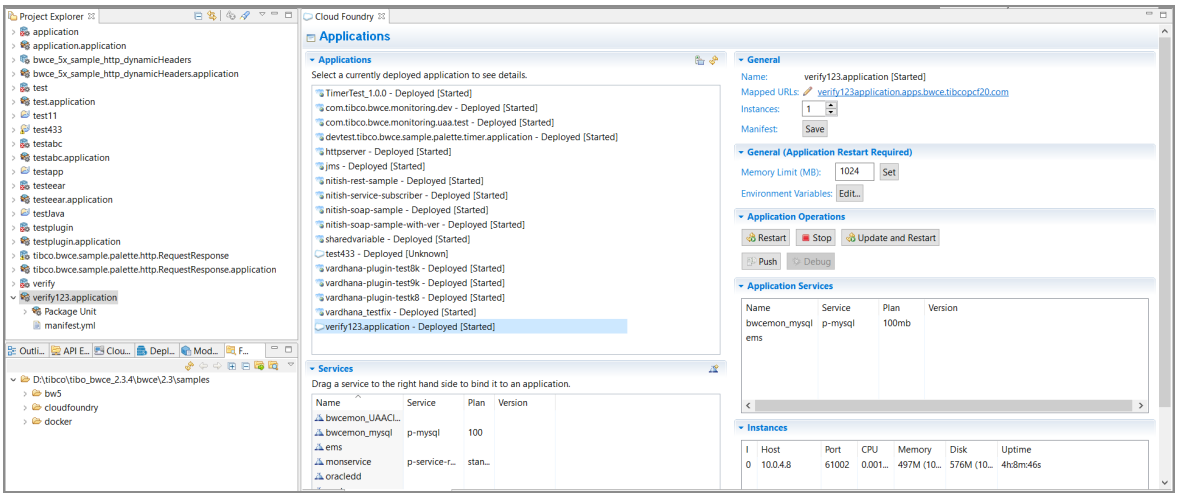
Deploying an application on Cloud Foundry from TIBCO Business Studio for BusinessWorks

You can deploy an TIBCO BusinessWorks Container Edition application on Cloud foundry directly from the TIBCO Business Studio for BusinessWorks.
It provides the following features to deploy an application:

- Drag and drop application for server application view.
- Select the services that are available in the PCF server.
- Define the environment variable and deploy the application on the configured PCF server.

Procedure

1. To deploy the application on Cloud Foundry, you can choose either of the following two options:
    - Drag the application from the **Package Explorer** view on the Pivotal Cloud Foundry server in the **Servers** view, or
    - Right-click the Pivotal Cloud Foundry server in the **Servers** view, select **Add and Remove** from the server context menu, and move the application from the **Available** to the **Configured** column.

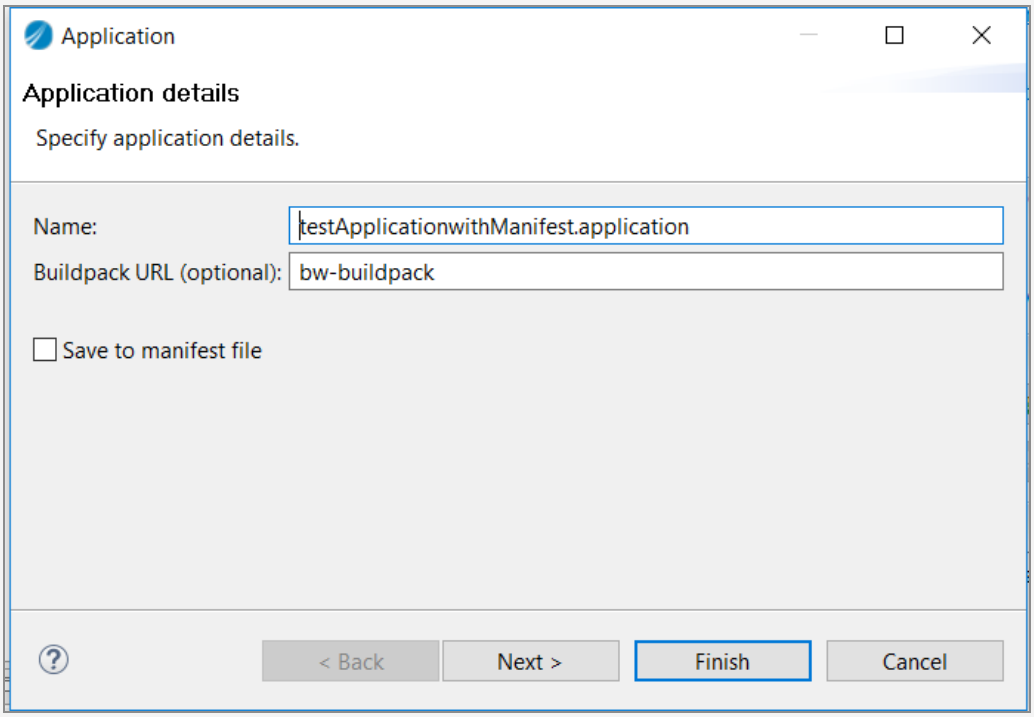


2. In the **Application Details** window:
    - By default, the **Name** field is populated with the application project name. You can enter a different name. The name is assigned to the deployed application, but does not rename the project.
    - If you want to use an external build pack to stage the application, enter the URL of the build pack.

    Click **Next** to continue.

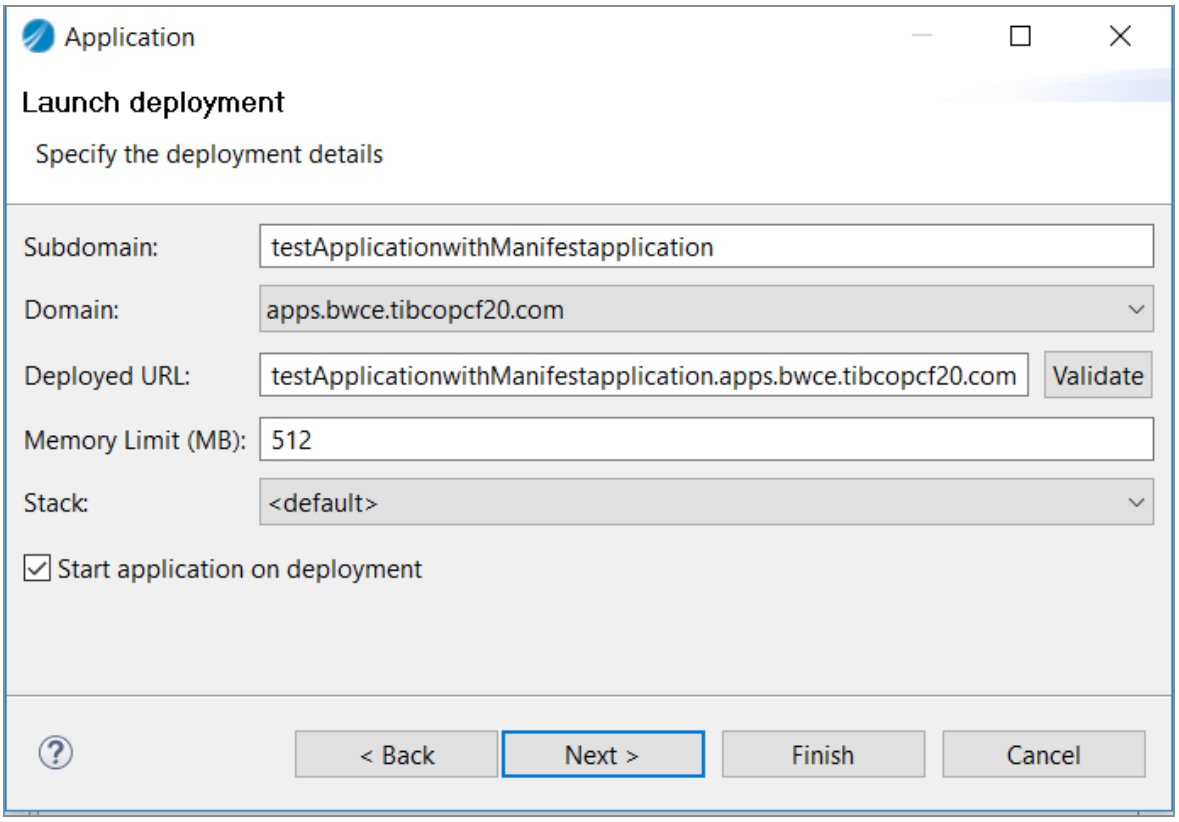
    You can deploy the application without further configuration by clicking **Finish**.

Note: The application default values may take a second or two to load, the **Finish** button might not be enabled immediately. A progress indicator indicates when the application default values are loaded, and the **Finish** button is enabled.



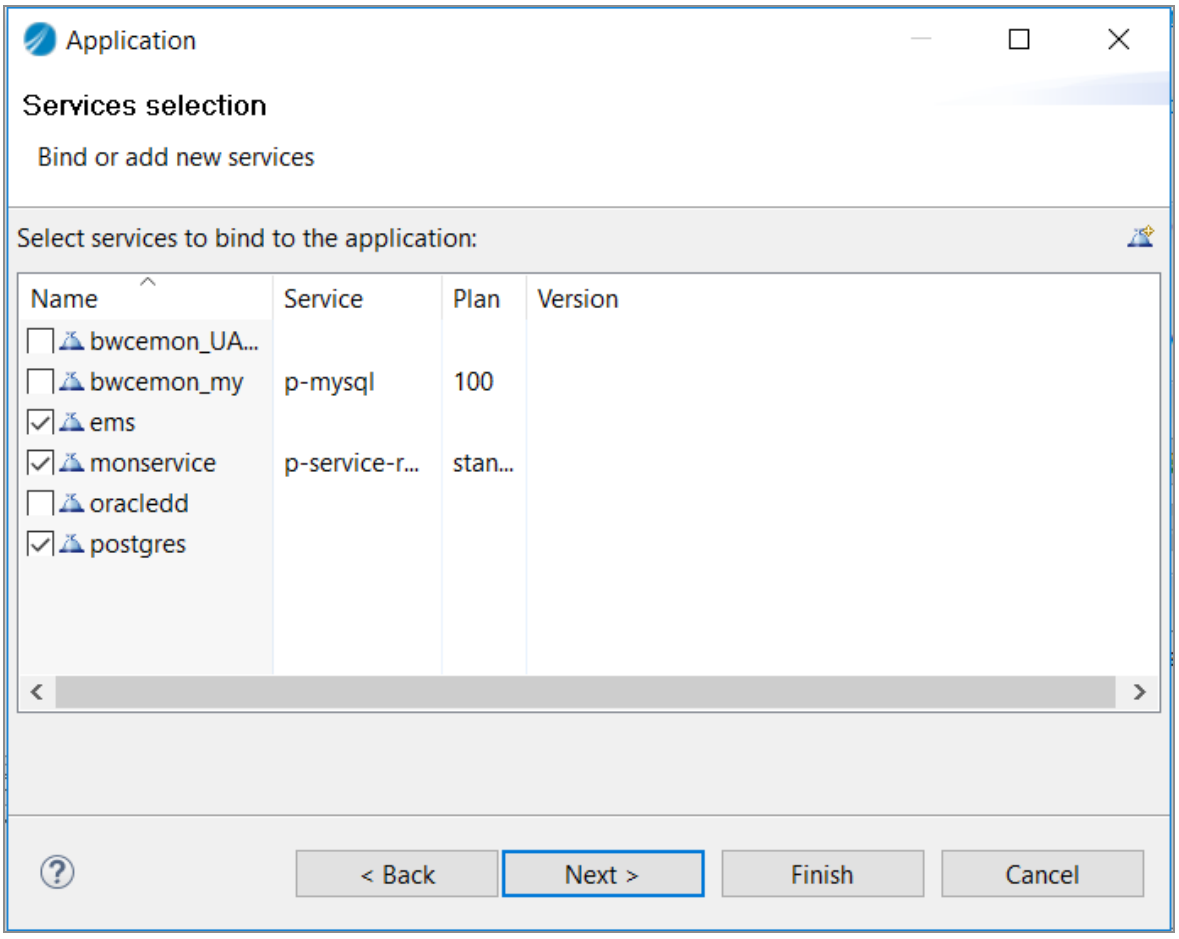
3. In the **Launch Deployment** window:
    - **Host**: By default, contains the name of the application. You can enter a different value if desired. If you push the same application to multiple spaces in the same organization, you must assign a unique host to each.
    - **Domain**: Contains the default domain. If you have mapped custom domains to the target space, they appear in the pull-down list.
    - **Deployed URL**: By default, contains the value of the **Host** and **Domain** fields, separated by a period (.) character.
    - **Memory Reservation**: Select the amount of memory to allocate to the application from the pull-down list.
    - **Start application on deployment**: If you do not want the application to be started on deployment, uncheck the box.

Note: This version of the Cloud Foundry eclipses plug-in does not provide a mechanism for mapping a custom domain to a space. You must use the cf map domain command to do so.



4. The **Services Selection** window lists services provisioned in the target space. Checkmark the services, if any, that you want to bind to the application, and click **Finish**. You can bind services to the application after deployment, as described in Bind and Unbind Services.

   As the deployment proceeds, progress messages appear in the Console view. When deployment is complete, the application is listed in the Applications pane.


Application
Services selection
Bind or add new services
Select services to bind to the application:
Name
Service
Plan
Version
bwcemon_UA...
bwcemon_my
p-mysql
100
ems
monservice
p-service-r...
stan...
oracledd
postgres
< Back
Next >
Finish
Cancel

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, and the resource references that are 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, and the resource references that are 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**.

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.

Note: 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**.

The processes you specified are removed, and no longer display in the Component Process editor.

Hot Update of Application Module Properties and Module Property for JDBC Shared Resource.

This feature is to update the application module properties and the JDBC shared resource module property without restarting the application.

The hot update feature can only be used by configuring an application with Configuration Management Services and Credential Management Services. For more information, see Using Configurations from Configuration Management Services and Credential Management Services for Properties of Type Password.

Note: Hot update on Kubernetes ConfigMap is not supported.

Hot Update is applied on Module properties and can be extended to utilize it for the shared resources (JDBC shared resource). A few fields in the shared resource when configured, can be updated via the hot update feature.

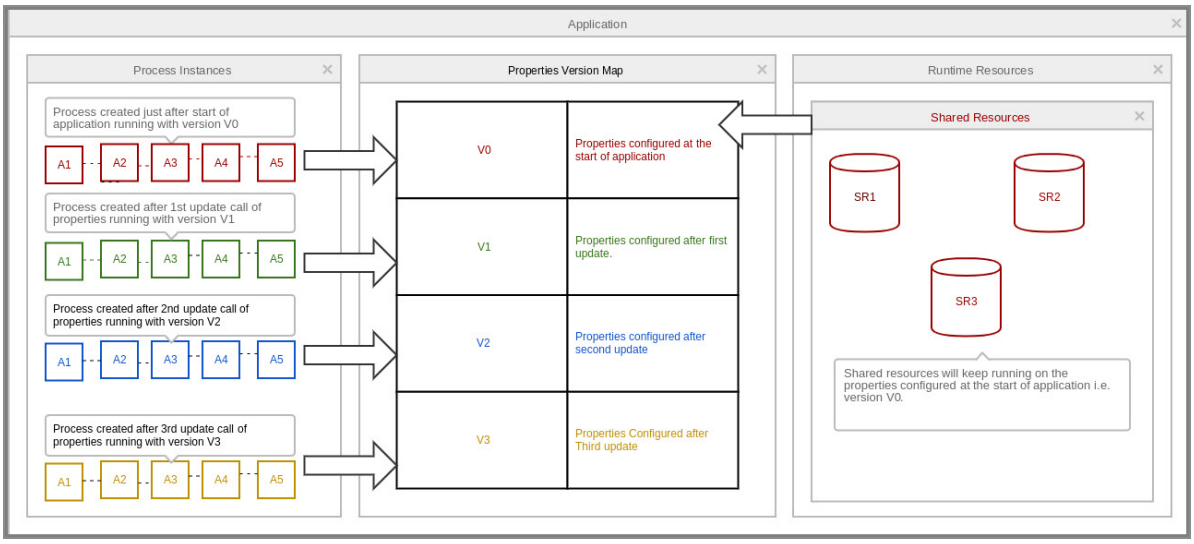
Affected areas where the updated properties come in effect at runtime:

The updated properties are affected only in the activity input mapping through:

- The updated properties are affected only in the activity input mappings through the `getModuleProperty()` of the x-path function.
- The process properties mapped to the module properties used as input mappings.

Behavioral changes during runtime after update call of the application properties and the JDBC shared resource module property:

In an application, a number of jobs (process instances) are running in an AppNode. After the update call of application module properties, the already running process instances continue to run with the old set of the module properties, and the new process instances created after the update run with the new set of module properties.



Behavioral changes for subprocesses after update call of application properties and JDBC shared resource module property:

There are two types of subprocesses:

- Direct subprocesses:
    - Spawned: For spawned subprocesses, the latest set of application properties and JDBC shared resource module property is used.
    - Non-spawned: The same set of application properties and JDBC shared resource module property is used as of the parent process.
- Service-based subprocess: In the service-based subprocess, the parent process waits for a reply from the subprocess, so the same set of properties as the parent is used for the subprocess.

Using hot update for application module properties

- To configure hot update, change the properties of the application at the configured configuration management service.
- Hit a public rest API **`/bwm/monitor.json/refresh`** of the container at port **`8090`**.
- The updated properties from the configured management services are used to update the application module properties inside the application.

Using hot update for JDBC shared resource module property

- To configure hot update, tokenize the JDBC shared resource module property. For more information, see Tokenizing Application Properties.
- Hit a public rest API **`/bwm/monitor.json/refresh`** of the container at port **`8090`**. To perform the hot update, use docker and consul for changing the supported shared resource module properties.
- The updated properties from the configured management services are used to update the module properties inside the application.

Limitations for hot update for JDBC shared resource module property

- You cannot change the driver.
- You can only change the IP address/host, port number, database name, and properties of the URL.
- If the entire URL or the protocol in the URL is changed, the changes are ignored.

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 .

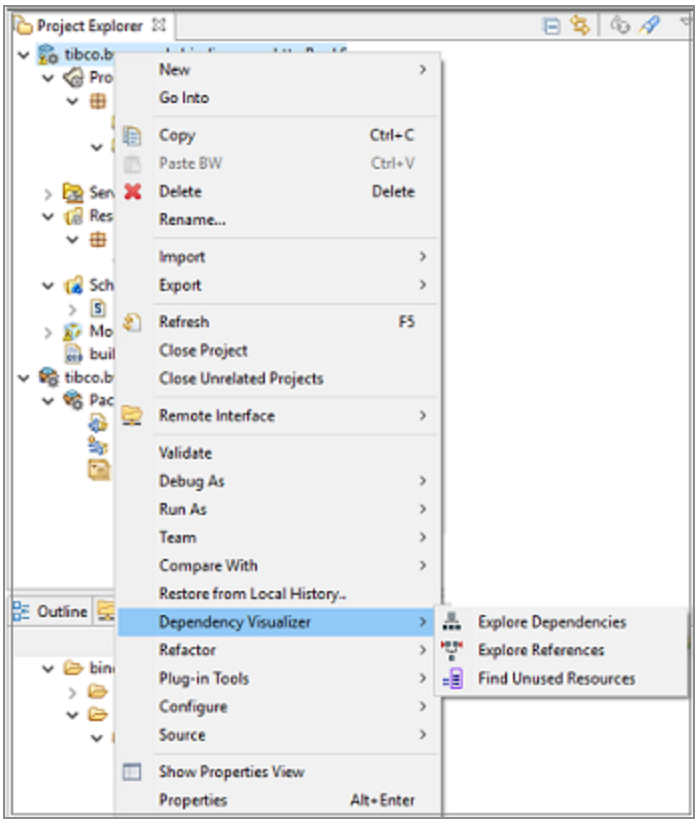
The information can be configured using the **Change Graph Layout** option . The layout options available are:

- 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

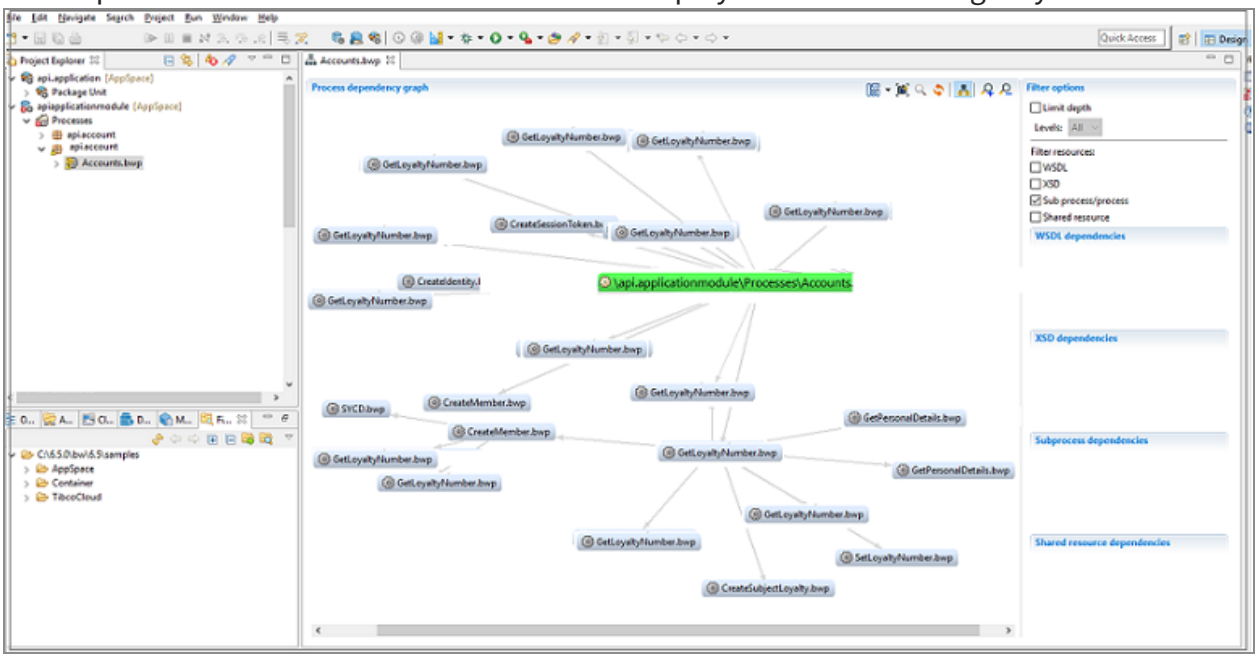
To access the Dependency Visualizer option from TIBCO Business Studio for BusinessWorks, navigate from the right-click menu of the application or the shared resource to **Dependency Visualizer > Explore Dependencies**. The dependency graph is displayed.



The dependencies between application modules and shared modules can be viewed and analyzed. The Project Dependency Graph shows the interdependence 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 it's 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.

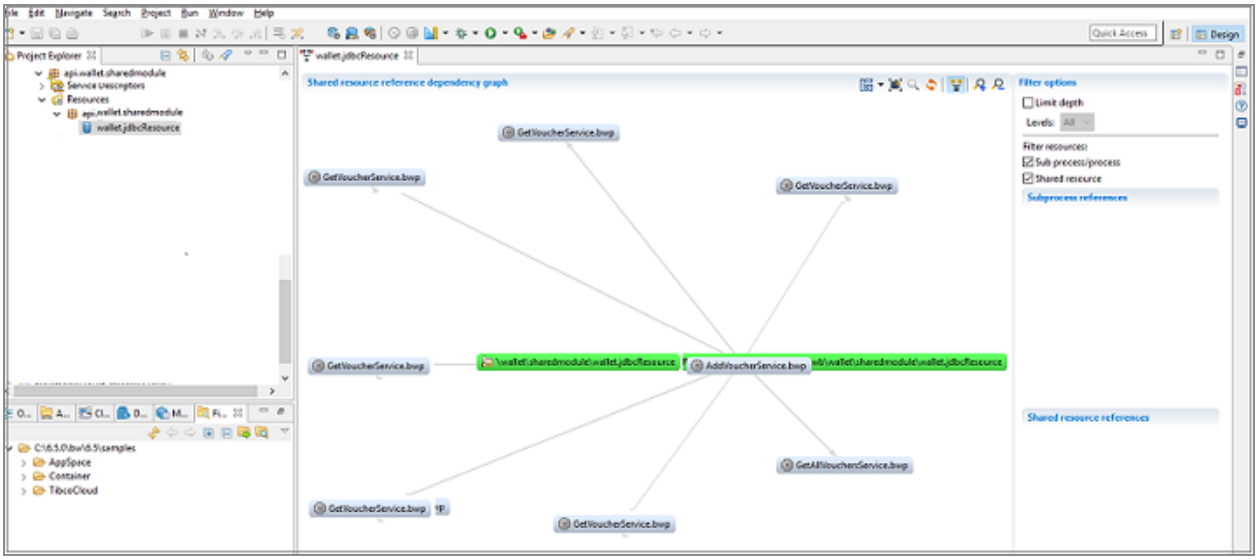
The required dependent nodes can also be highlighted using the option **Highlight Dependencies**.

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**.

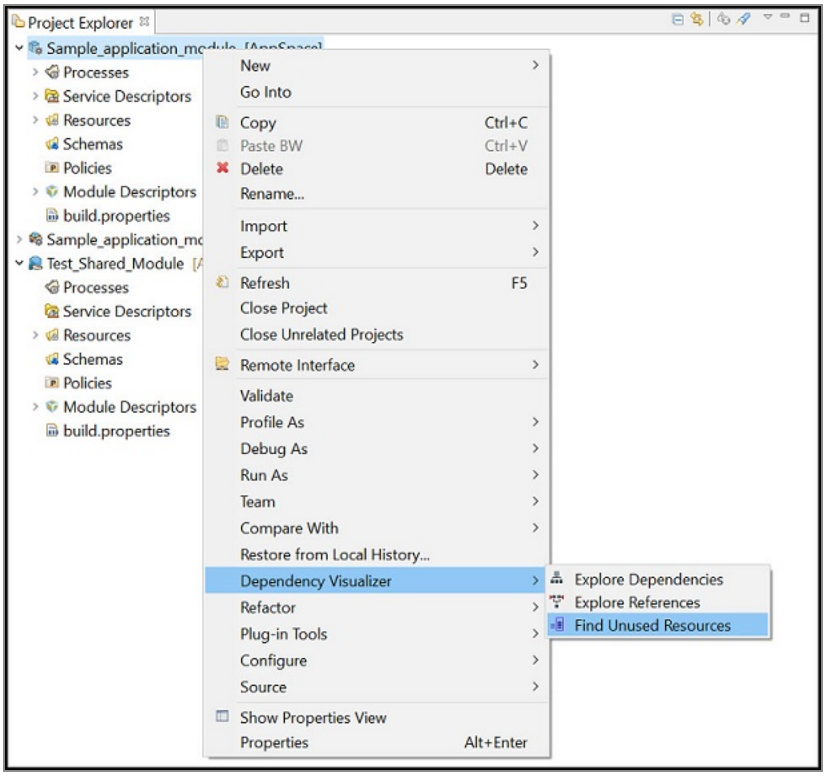
When you click the root node or any of the level references, the references are displayed in the following categories:

- WSDL References
- XSD References
- Sub process/processes References
- Shared Resource References

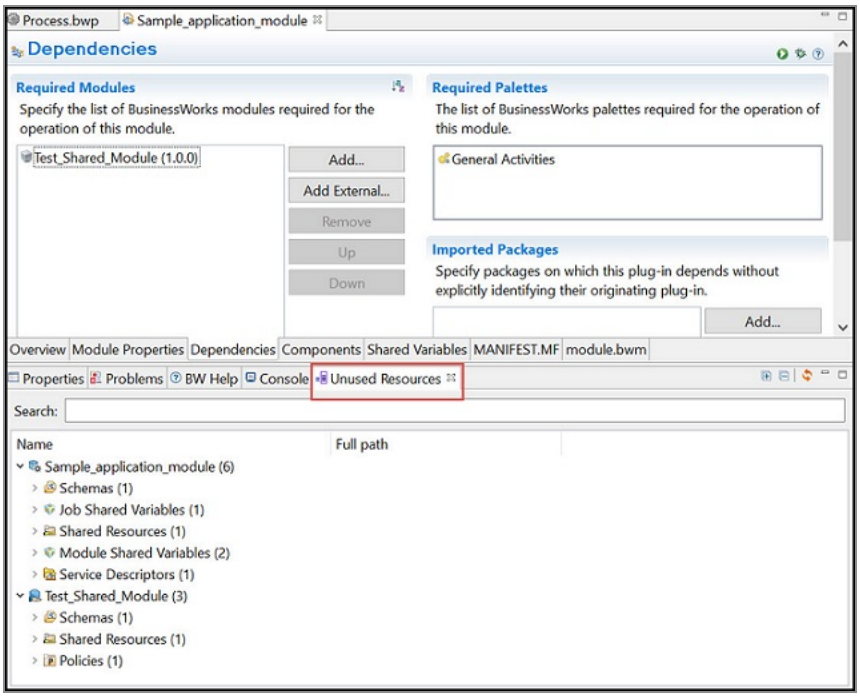
Unused Resources

The view displays unused resources from the 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.



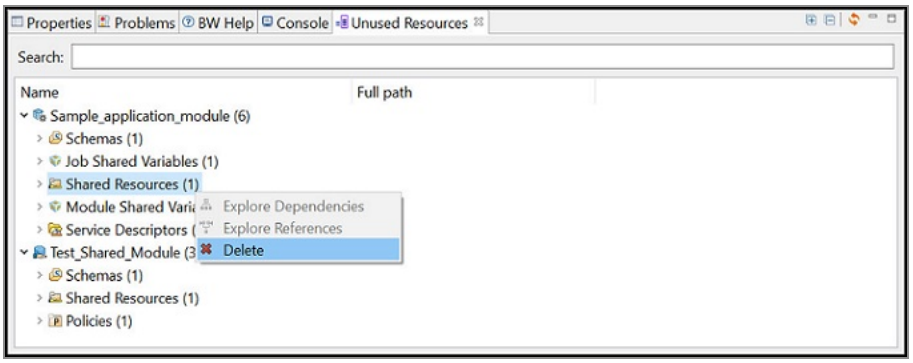
On the **Unused Resources** view, the total number of unused resources is displayed. This includes:

- Process
- Schemas
- WSDL
- Shared Resource
- Policy
- Job Shared Variables
- Module Shared Variables

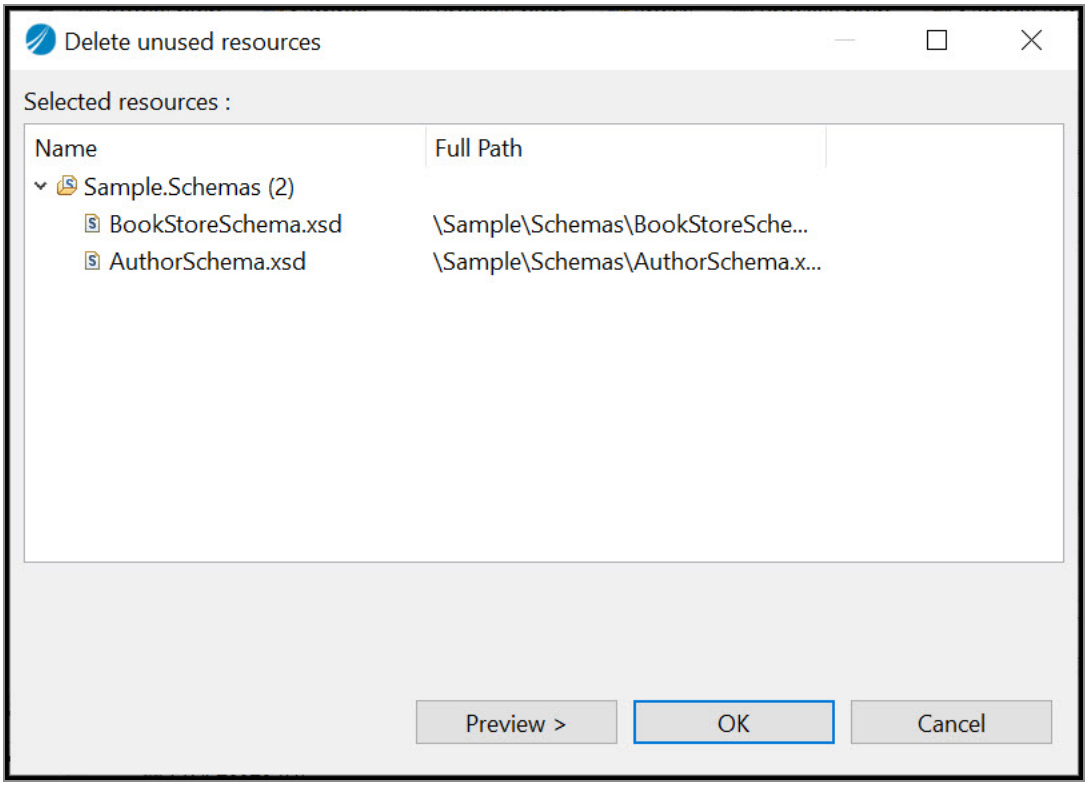
If you add, remove, or re-factor unused resources in the application and you need to find unused resources again, use **Refresh** button in the **Unused Resources** view to get the updated unused resources. Refresh operation takes some time when using with large projects.

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**.

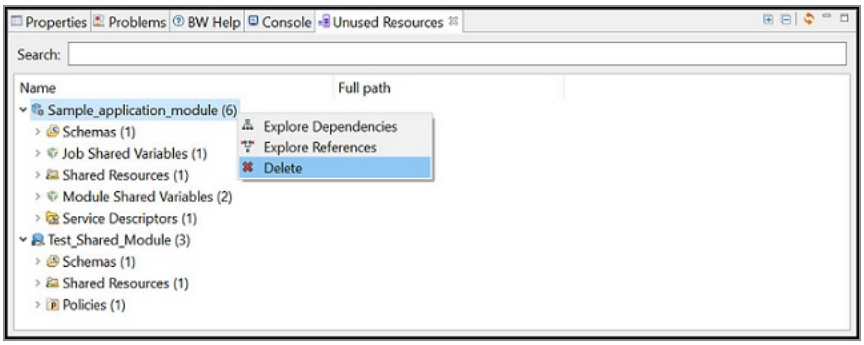


When you try to delete an unused artifact which has dependencies as well as references on the other unused artifacts, a new Delete unused resources dialog opens.



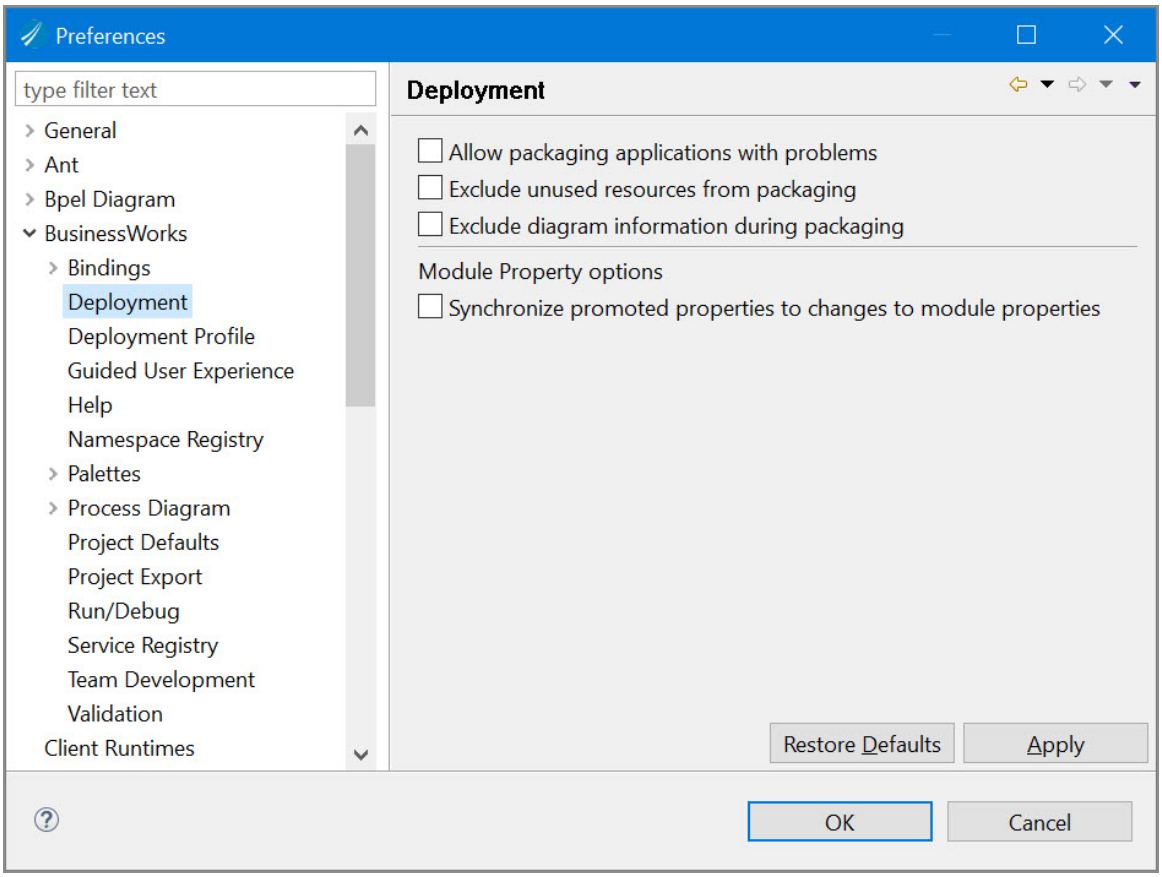
It shows the list of selected resources. To see the list of corresponding dependencies, click **Preview** button. To delete dependent resources, click **Ok**.

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.

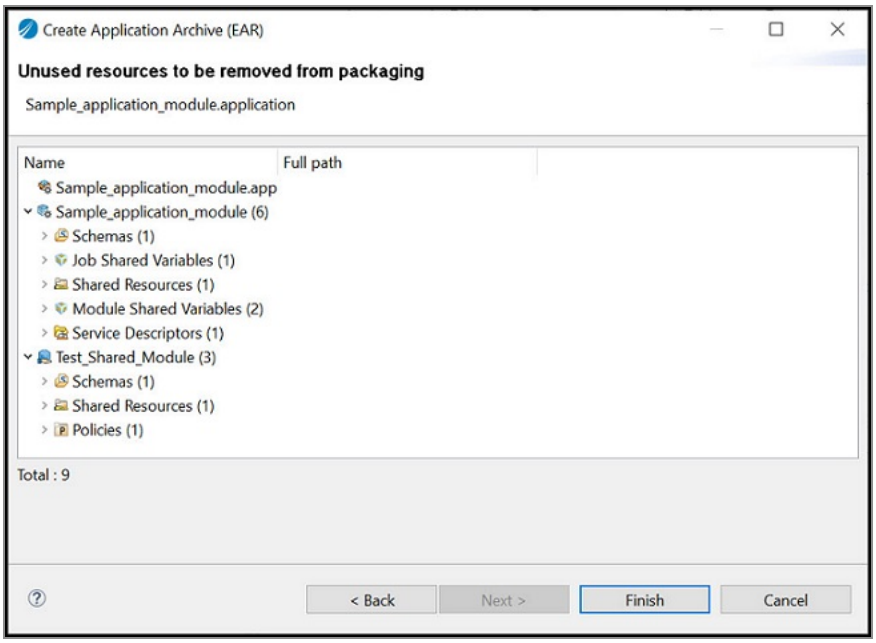


Generate EAR without unused resources

To generate an application archive file (.EAR) without unused resources, select the **BusinessWorks > Deployment > Exclude unused resources from packaging** checkbox in the Preferences dialog.



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 BusinessWorks Container Edition Projects

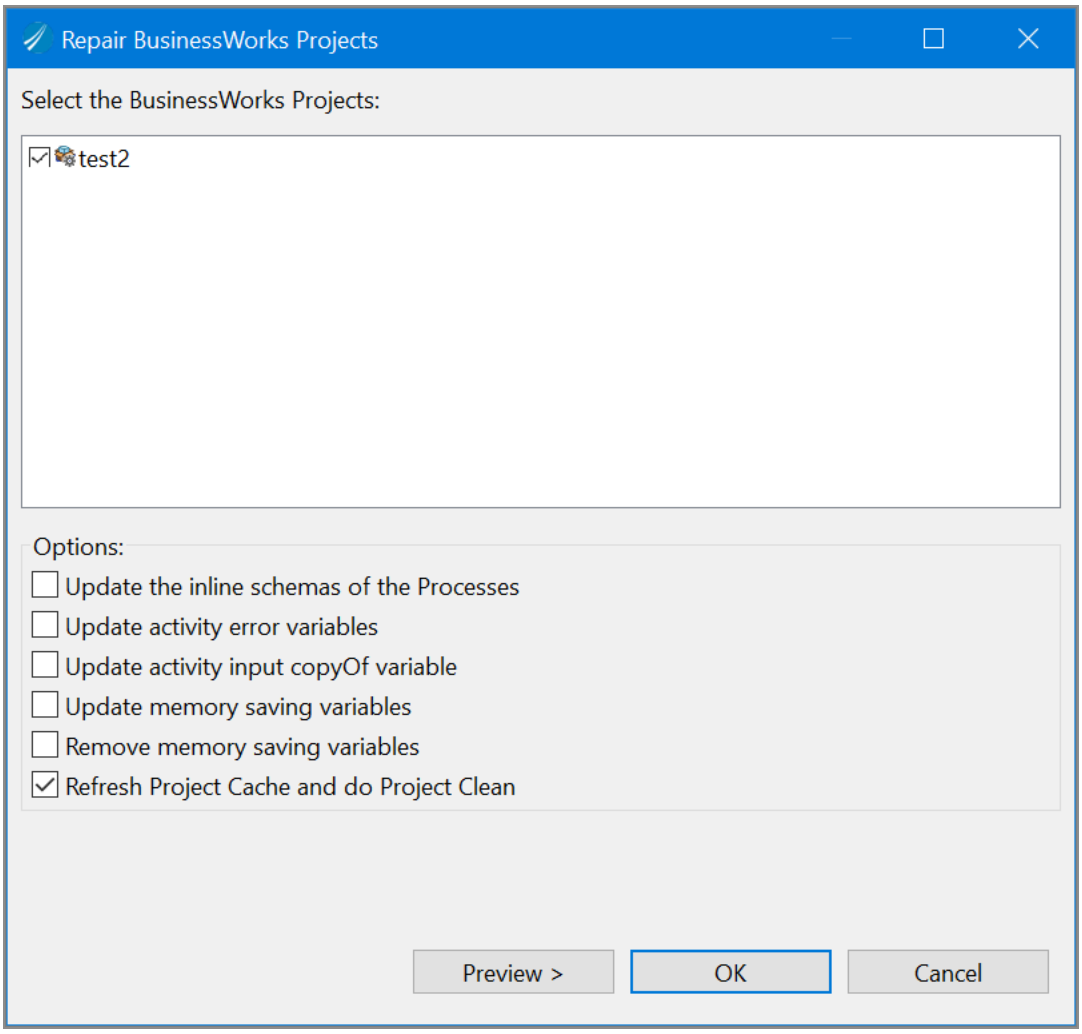
Repairing the BusinessWorks projects is one of the refactoring activities available on right-clicking the project in the Project Explorer pane, and clicking **Refactor > Repair BusinessWorks Projects...**.

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.

Note: 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.



The following table describes the options available in the Repair BusinessWorks Projects wizard.

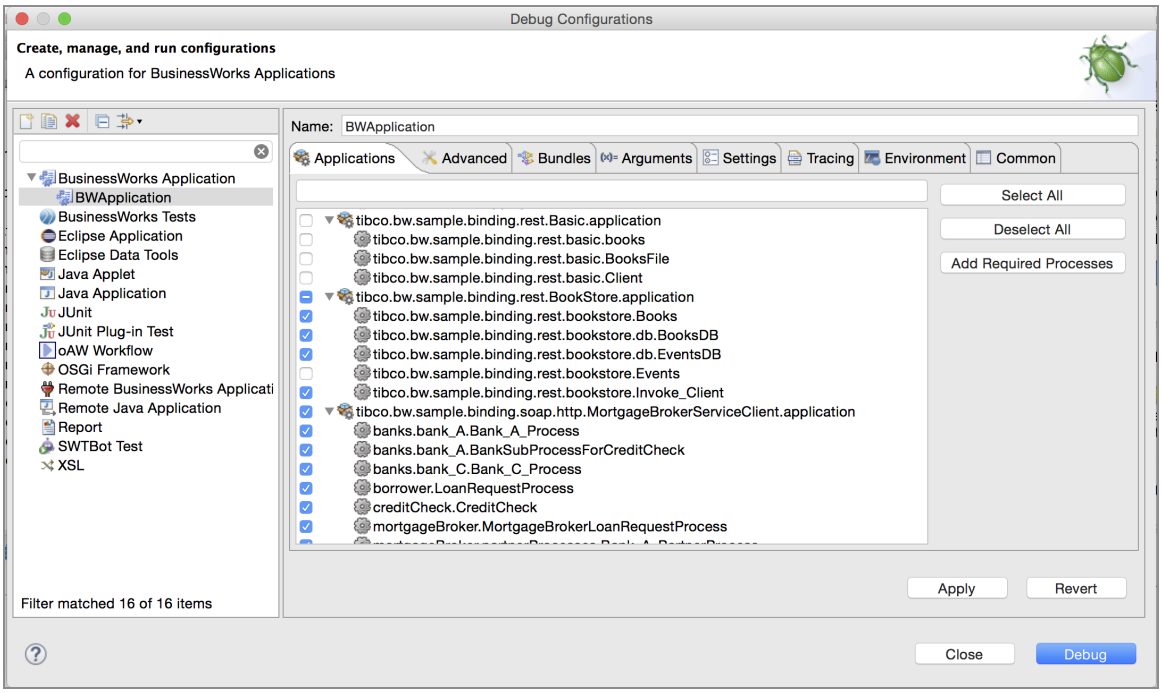
Field	Description
Update the inline schemas of the Processes	To update the Engine type inline schemas in the existing projects with new format
Update activity error variables	To create an extra error variable for the activities in the existing projects

Field	Description
Update activity input copyOf variable	To update the *tibex:copyOf* 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.

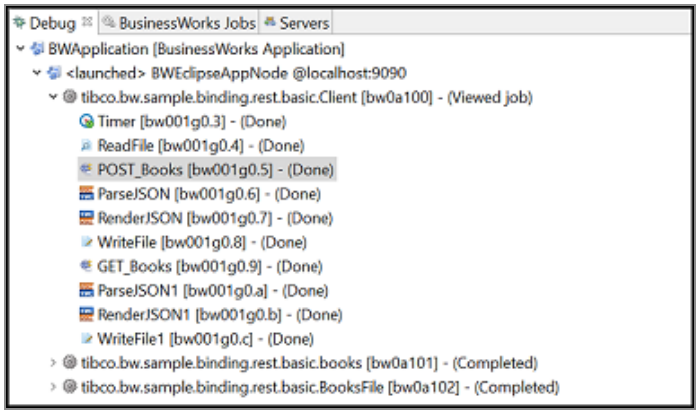
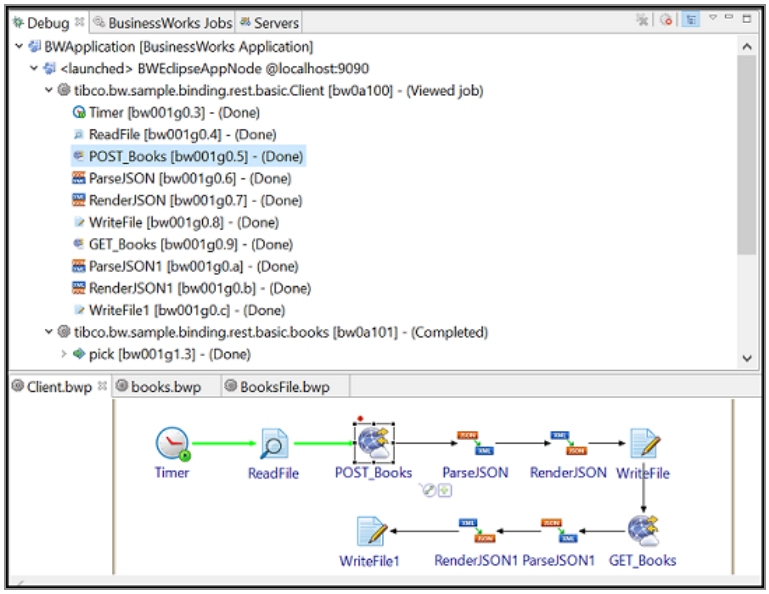
By default, the debugger lists all the process and sub processes of an application module, shared module and nested shared module in the debug configuration window. You can select applications, and processes in an application, to launch in the debugger.

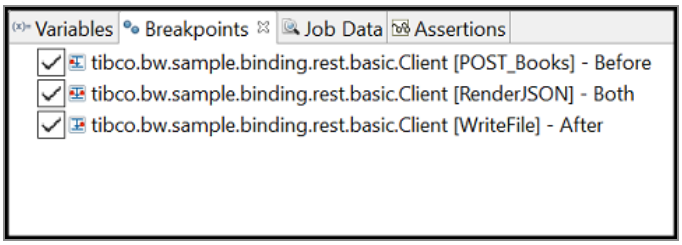
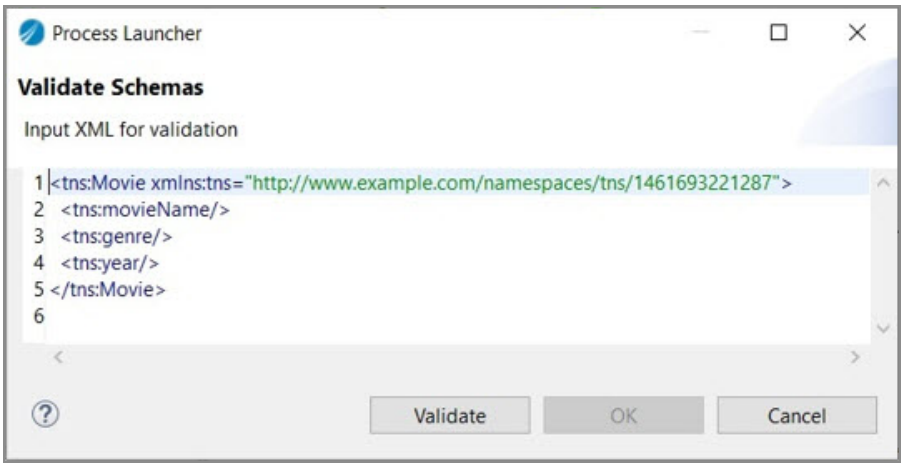


The Debug perspective consists of a set of views which are related to the debugging task. Some views, for example the Project Explorer view, are not available in the Debug perspective, while others, such as Debug and Breakpoints, are available in the Debug perspective. There are multiple ways to open 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:

View	Description
Debug	Shows the list of debug launches and allows you to manage them using the icon bar. When you are done with debugging any activity in a process, that activity is indicated with the **(Done)** label.  If you select any activity, that activity is highlighted with a square in a process. 
BusinessWorks Jobs	Shows all running jobs and allows you some basic management such as, to Clear All Jobs .

View	Description
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.
Breakpoints	Shows the breakpoints used for debugging.  For more information about adding breakpoints for debugging an application, see Using Breakpoints.
Job Data	Shows available information about the running process instances. You can copy the job data to a text file to see variables and values used in a process. For more information, see Copying Job Data.
Process Launcher	Shows available sub-processes that can be launched. Inputs to the process instance can be provided in the process launcher. 

View	Description
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.

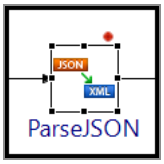
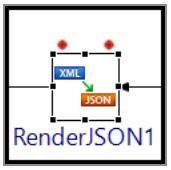
Using Breakpoints

Use breakpoints when debugging an application to halt and check for values passed among activities or processes.

Procedure

1. Select an activity in a process.
2. Right-click the activity and select any of the following options:

Option	Result
Breakpoint > Set Before	Sets a breakpoint before the activity. You can see the red dot on a top-left side of the activity icon. For example:  Additionally, you can set another breakpoint after the activity. Right-click the activity and select **Breakpoint > Set After**.
Breakpoint > Set After	Sets a breakpoint before the activity. You can see the red dot on a top-right side of the activity icon.

Option	Result
	For example:  Additionally, you can set another breakpoint before the activity. Right-click the activity and select **Breakpoint > Set Before**.
Breakpoint > Set Before/After	Sets breakpoints before as well as after the activity. You can see the red dot on a top-left side and top-right side of the activity icon. For example: 

3. Similarly, you can remove breakpoints. Right-click the activity with a breakpoint and select:

Option	Description
For the breakpoint set after the activity	Right-click the activity and select **Breakpoint > Remove After**.
For the breakpoint set before the activity	Right-click the activity and select **Breakpoint > Remove Before**.
For the breakpoint set before and after the activity	You can remove any of the before, after, or both breakpoints.

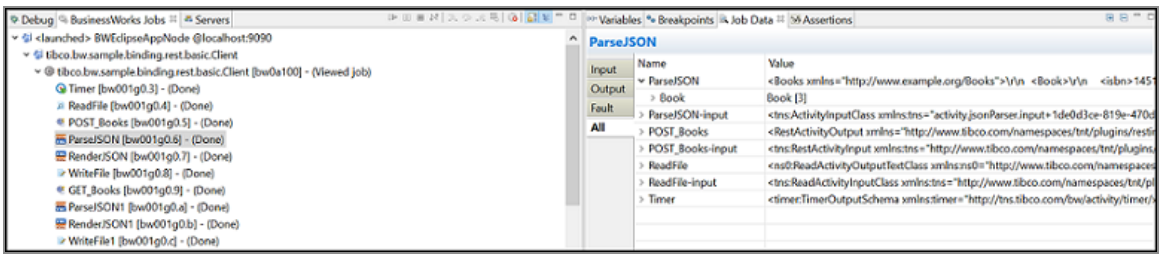
Copying Job Data

You can copy the job data in a clipboard to have a clearer view in other text editors.

Procedure

1. In the Debug perspective, go to the **BusinessWorks Jobs** tab.
2. Select an activity of which you want to copy the job data.

   The input, output, and fault job data is available in the Job Data tab. The All tab contains data of all prior activities in the alphabetical order of the selected activity in a process.



3. Select any row in the Job Data view. Right-click and select:

Option	Description
Select All	Selects all job data in the Job Data tab.
Copy Values	Copies only data in the *Value* column for the selected row. For example: If you select a row statuscode 200, then this option copies only 200 to the clipboard.
Copy Variables	Copies data in a tag and value format. For example: If you select a row statuscode 200, then this option copies <statuscode> 200 </statuscode> to the clipboard.

Note: TIBCO Business Studio for BusinessWorks displays only a limited amount of user input as a job data tool tip in the job data view of the debugger. If you want to view the complete content of the job data use the Copy Values or Copy Variables option in the debugger view.

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, you can select the following:

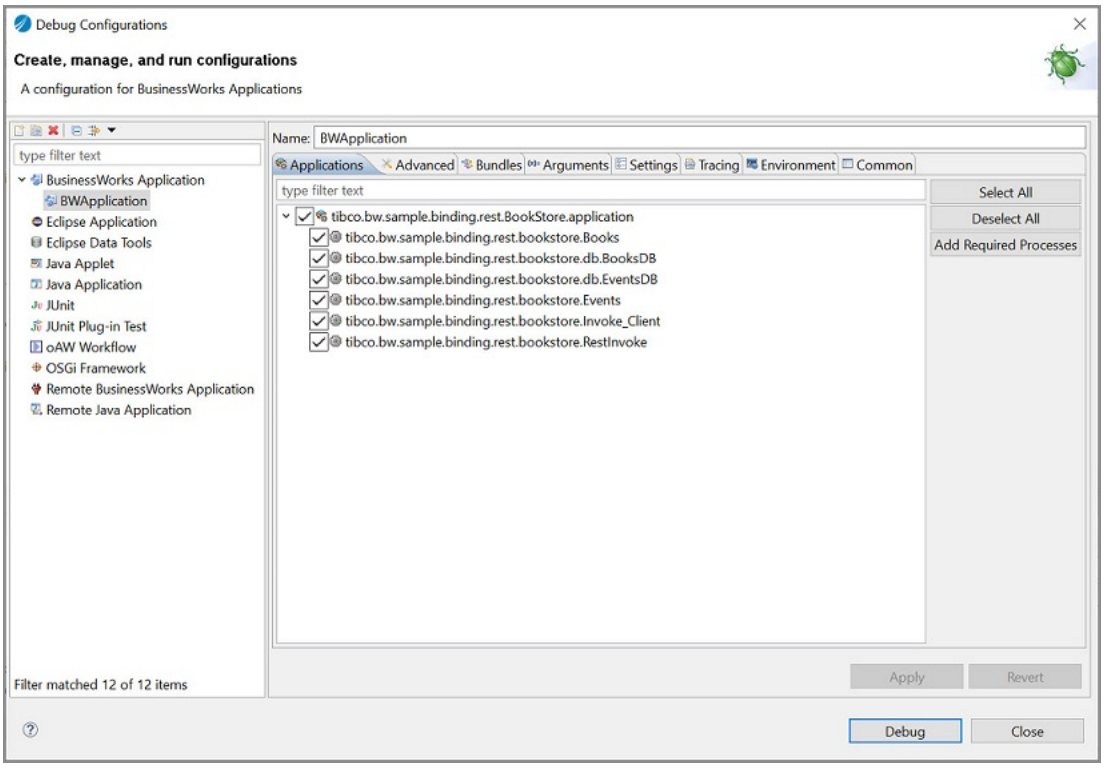
- Applications to debug.
- Advanced configurations such as logging configuration and engine debug port.
- Arguments: program arguments such as the target operating system, target architecture, target web services, working directory, and so on, and VM arguments such as *TIBCO_HOME*, *port number*, or 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**.

Note: 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 checkbox 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.

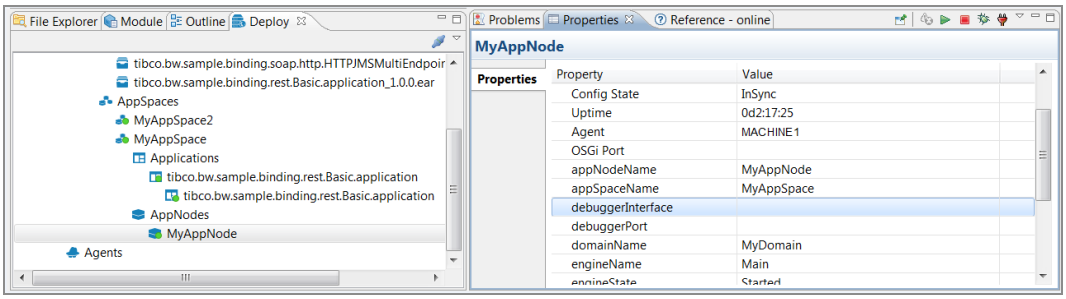
6. After completing the execution, click the **Terminate** icon to stop the process.

Remote Debugging

You can debug an application running on a remote AppNode through TIBCO Business Studio for BusinessWorks.

Procedure

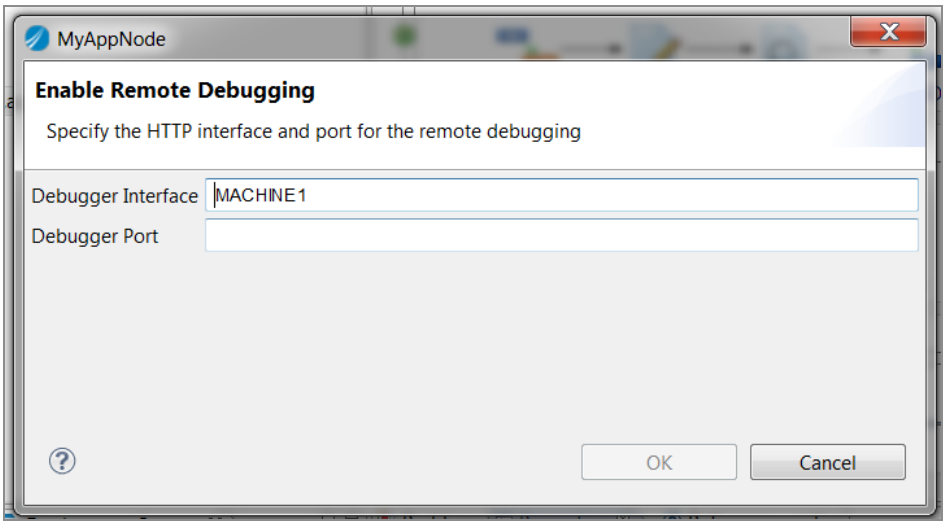
1. Enable the AppNode for debugging. (The AppNode must be running.)
   a. Open the network in the **Deploy** pane and choose the AppNode. The AppNode properties are displayed in the **Properties** pane.



b. Click the **Enable Debug** icon in the Properties pane to enable remote debugging.

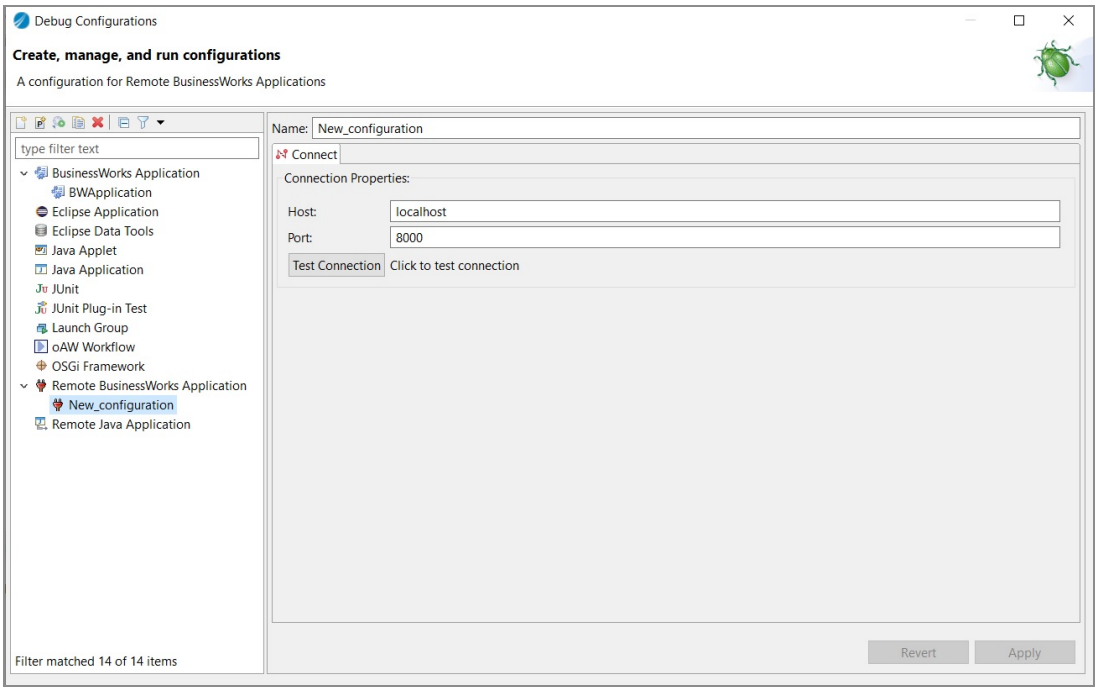
c. Enter the interface and port for remote debugging on the selected AppNode in the Enable Remote Debugging dialog.

- **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: `Internal server error`



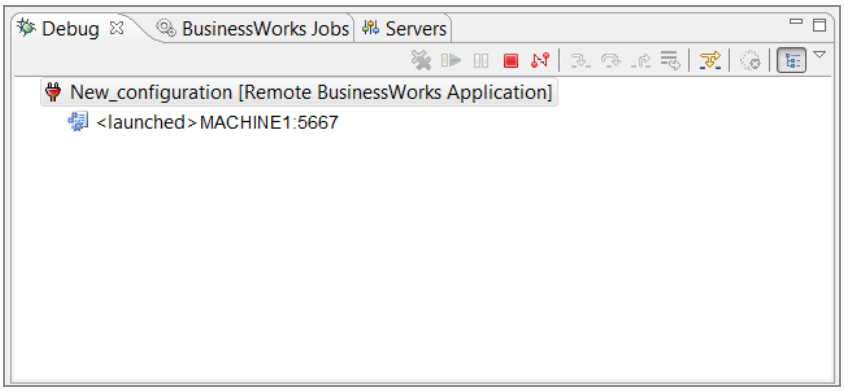
> **Note:** The remote debugger can also be launched with the **Debug** icon in the Properties view. The connection parameters on the Enable Remote Debugging dialog are automatically be entered based on the AppNode configuration.

2. In TIBCO Business Studio for BusinessWorks, create a Remote Debug launch configuration.
    a. Choose **Run > Debug Configurations**.
    b. In the Debug Configuration dialog, go to **Remote BusinessWorks Application > New_configuration** and 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.
        - **Port**: The remote debug port. The port cannot be in use.
        - **Test Connection**: This button is used to check connection with the deployment server.



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.



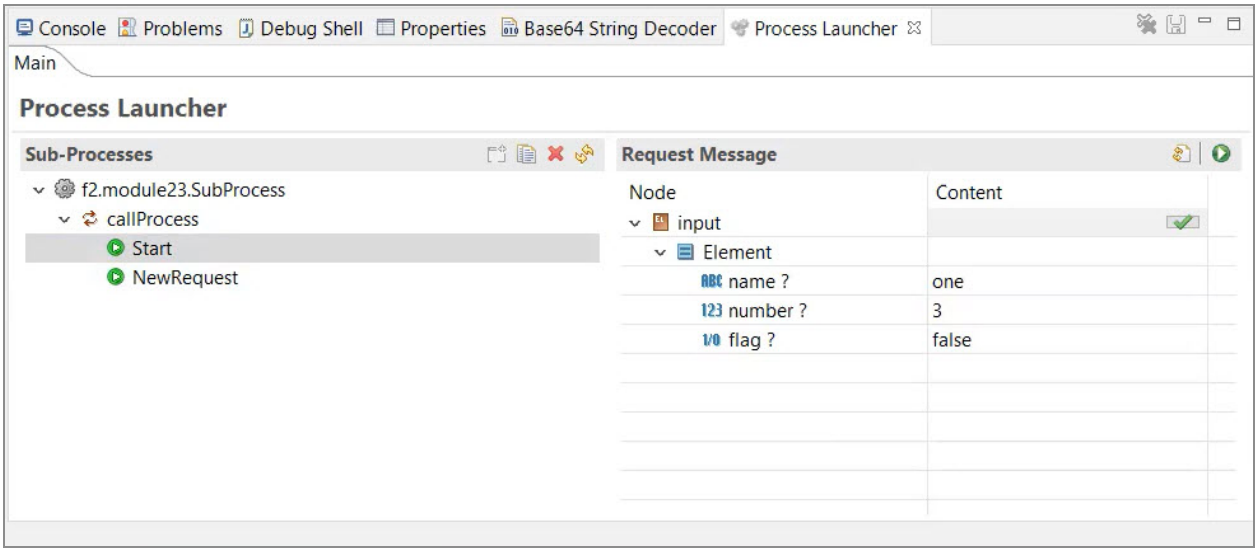
Process Launcher

Using the process launcher feature, you can see and test the subprocesses, which can accept the input. The input to the process instance is provided through the process launcher to test the subprocess individually.

You can use the process launcher by running the application in the debug mode. That is navigate to **Run > Debug Configurations**, navigate to the Process Launcher tab, and select the subprocess to test.

In the Process Launcher tab, all the requests created for a subprocess are listed under the respective subprocesses. The invoke button (▶) for the listed request messages is enabled only after initiating the process, which called the respective subprocess to be tested".

Process Launcher view:



Note:

- To perform the unit testing of a subprocess from the process launcher, first load the subprocess along with the corresponding starter process in the debug mode.
- A subprocess has to be called at least once before you call it separately from the process launcher.
- You cannot test a subprocess directly or independently without initiating and invoking it.

You can also launch the process launcher from the **Window > Show View > Other...** option (design view).

On the Show View window, navigate to **BusinessWorks > Process Launcher**. Or type process launcher in the text box.

Note:

- When you launch the process launcher using design view, the subprocess requests are disabled and you cannot invoke them. However you can add or delete as many requests as required and validate schemas, but there should be at least one request present to provide the input to the subprocess.
- To initiate a process or subprocess, you have to launch the process using debug mode and then you can invoke the requests to verify the subprocess functionality.
- You can validate the schemas of the request message in debug mode and design view by clicking the **Launch schema validation dialog** icon ( ) present next to the input.

Collaborative Application Development

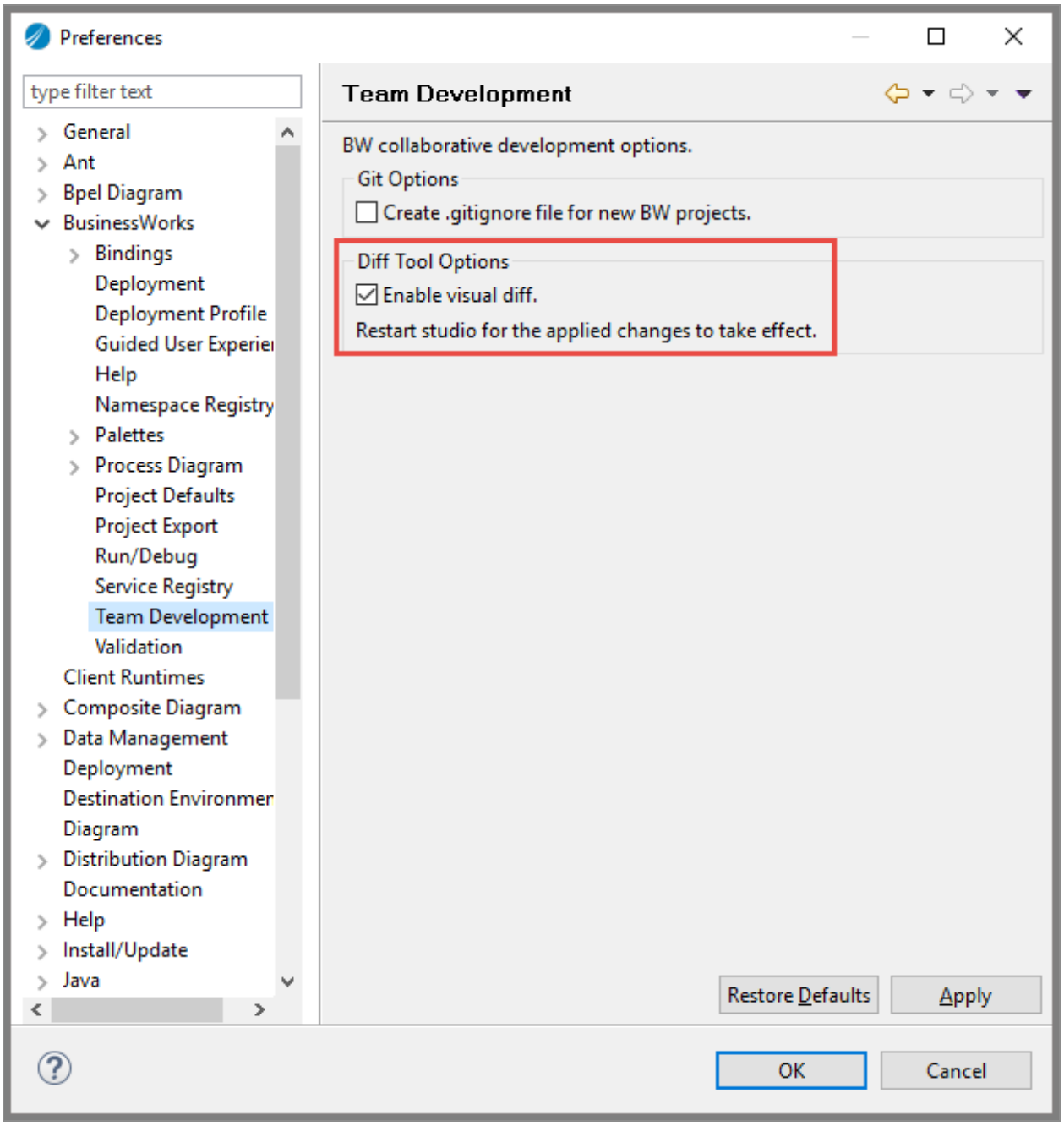
Collaborative application development process helps multiple process designers to design a process simultaneously.

To keep the track of collaborative development work, configure TIBCO Business Studio for BusinessWorks with Git plug-in. Once you configure Git, you can commit special folders of an application module by adding .gitignore files. For more information, see Generating gitignore files.

Diff Viewer

The Visual Process Diff tool provides the ability to view changes made to process files visually for different revisions.

To enable diff viewer, navigate to **Windows > Preferences > BusinessWorks > Team Development**. Select the **Enable visual diff** checkbox under **Diff Tool Options**. By default, this checkbox is always selected.

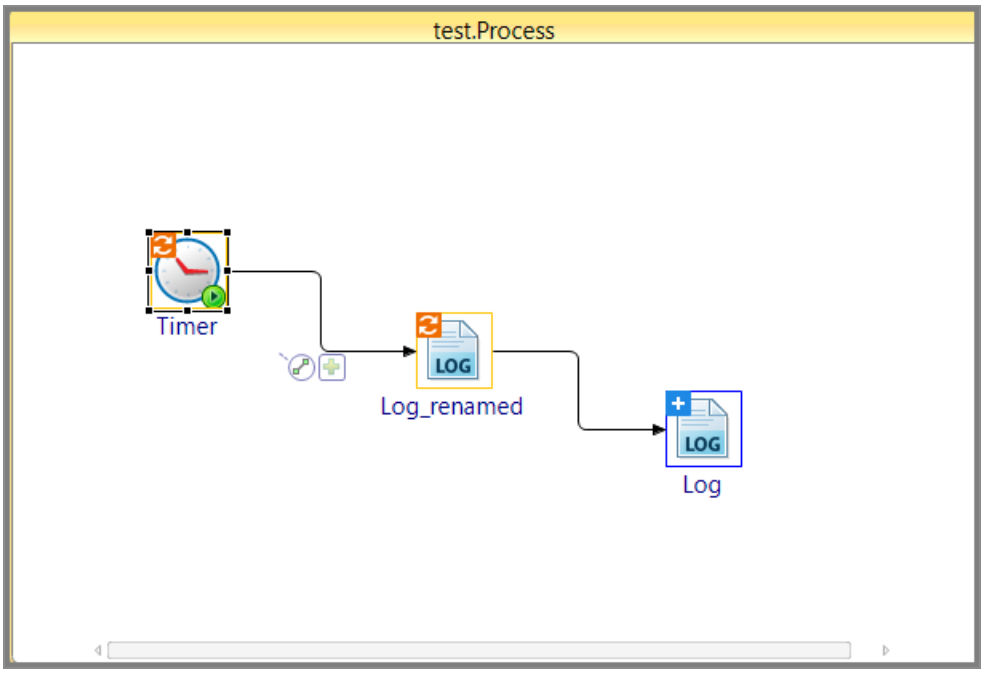


The Visual Process Diff tool is used to compare:

- Process or Scope Variables (primitive types only)
- BW Palette Activities
- Process Properties (Application and Module Properties)
- Shared Resources (Resources which are shipped only with the product)

- Groups
- Fault Handlers

After comparing the different revisions of a process, the diff viewer displays the process with decorations to indicate the changes in activities. It also navigates controls through the modified activities and their properties.



Change Indicators- These are small indicators displayed on the process to indicate the activity change.

Change Indicators	
	Blue indicates a newly added activity.
	Red indicates deletion of an activity.

Change Indicators	
	Orange indicates changes in one or more properties of that activity.

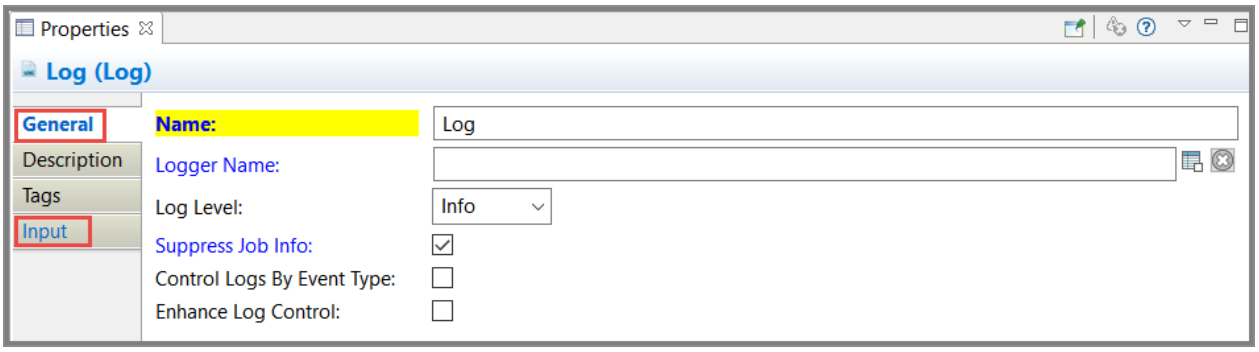
Navigation Controls: The navigation controls help to navigate through the property changes.

Activity Navigation Control	
	Navigate to the next difference. It navigates to the next activity which has changed.
	Navigate back to the previous difference. It navigates back to the previous activity which has changed.

While navigating, current activity change is marked by selection indicators and property view for respective activities are displayed on each side.

Navigation Property Changes	
	**Next Property Difference**- Navigate to the next property change. It navigates to the next property within the current object which has changed.
	**Previous Property Difference**- Navigate back to the previous property change. It navigates back to the previous property within the current object which has changed.

All the modified properties and their corresponding tabs are shown in blue color to indicate the change. While navigating through the properties, current property is highlighted in Yellow.



In the above example, only the **General** and **Input** tabs are highlighted, since the properties displayed on these tabs are changed.

There are different modes of operations by which diff viewer is viewed:

- **Compare with Local History**
- **Compare with Each Other**
- **Compare with another revision from an SVN Repository**
- **Compare with another revision from Git Repository**

Attention:

- The diff viewer is not supported for:
  - LDAP Authentication
  - **Password** field in a shared resource
  - REST and SOAP Reference
- The namespace registry feature auto-populates namespace prefixes, due to which change is indicated in the input of an activity, but there is no visible change on the UI. This change may come as a side effect of changing input mapping of any activity in the process. The auto-population of prefixes can be disabled by selecting the **Auto populate process namespace registry with generated prefixes for namespaces** checkbox under **Windows > Preferences > BusinessWorks > Namespace Registry**.
- It is not possible to make any edits in the **BW Compare** view. To make new changes, select the **Design** view.

Process Diff Viewer

The following are the different modes of operation by which diff viewer is viewed for a process.

- Compare with Local History
- Compare with Each Other
- Compare with another revision from SVN Repository
- Compare with another revision from Git Repository

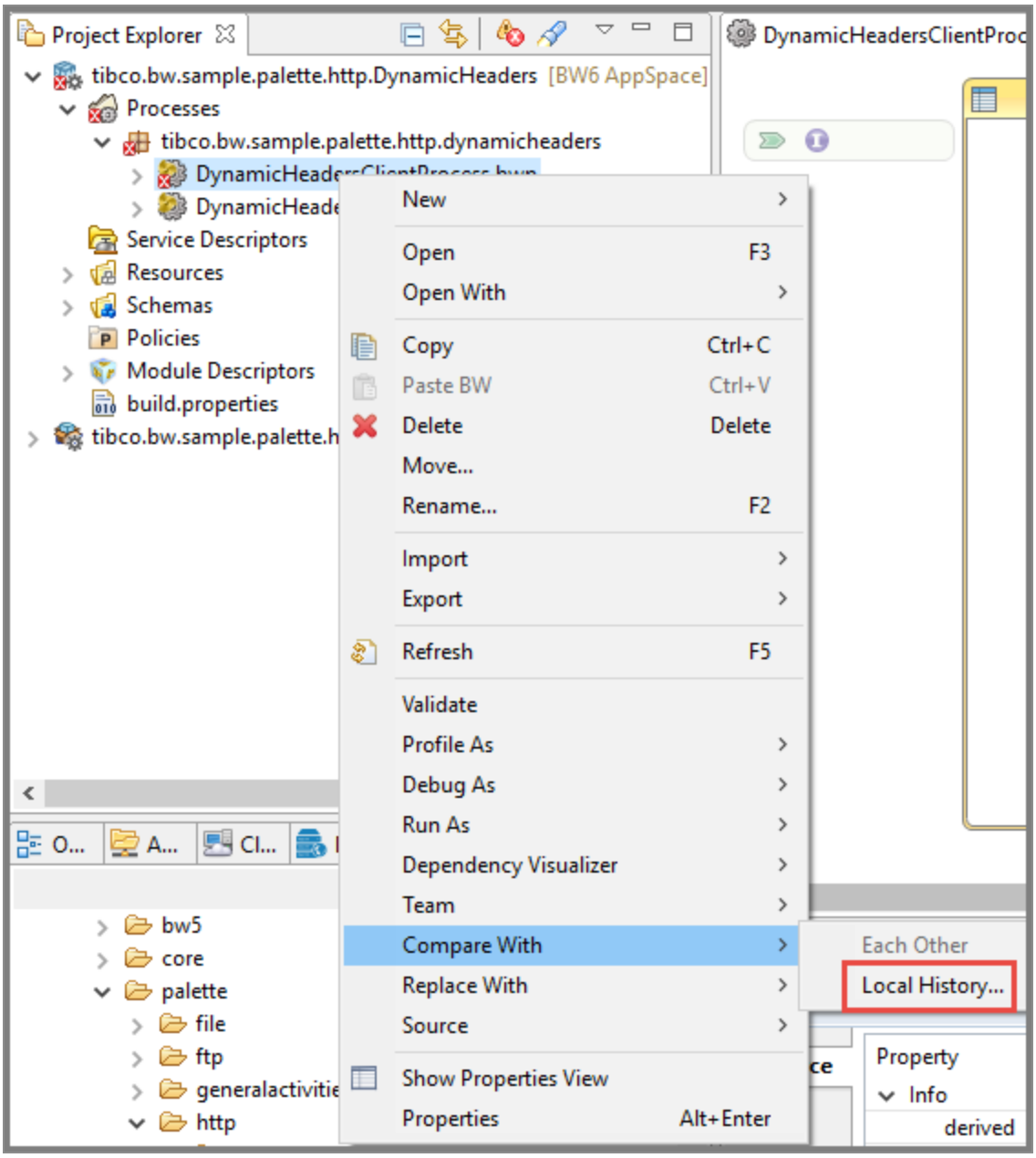
Compare with Local History

Before you begin

- Import an already existing sample or create a new project.
- The selected process should be modified and saved at least once before comparing the two revisions. This ensures that there is a local history available in the workspace.

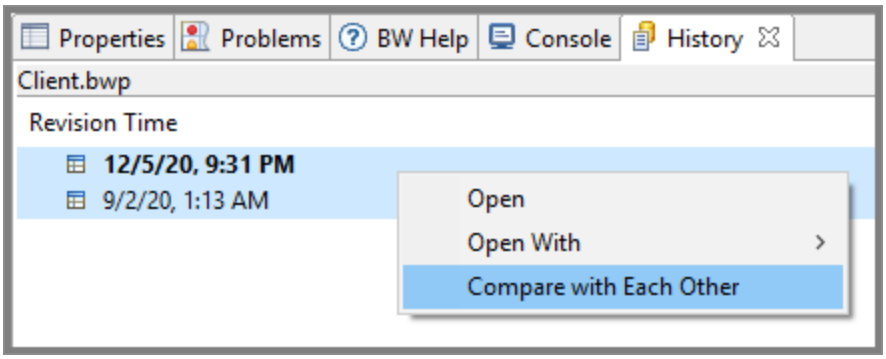
Procedure

1. In **Project Explorer** view, right-click on the selected process and select **Compare With > Local History** or **Team > Show Local History**

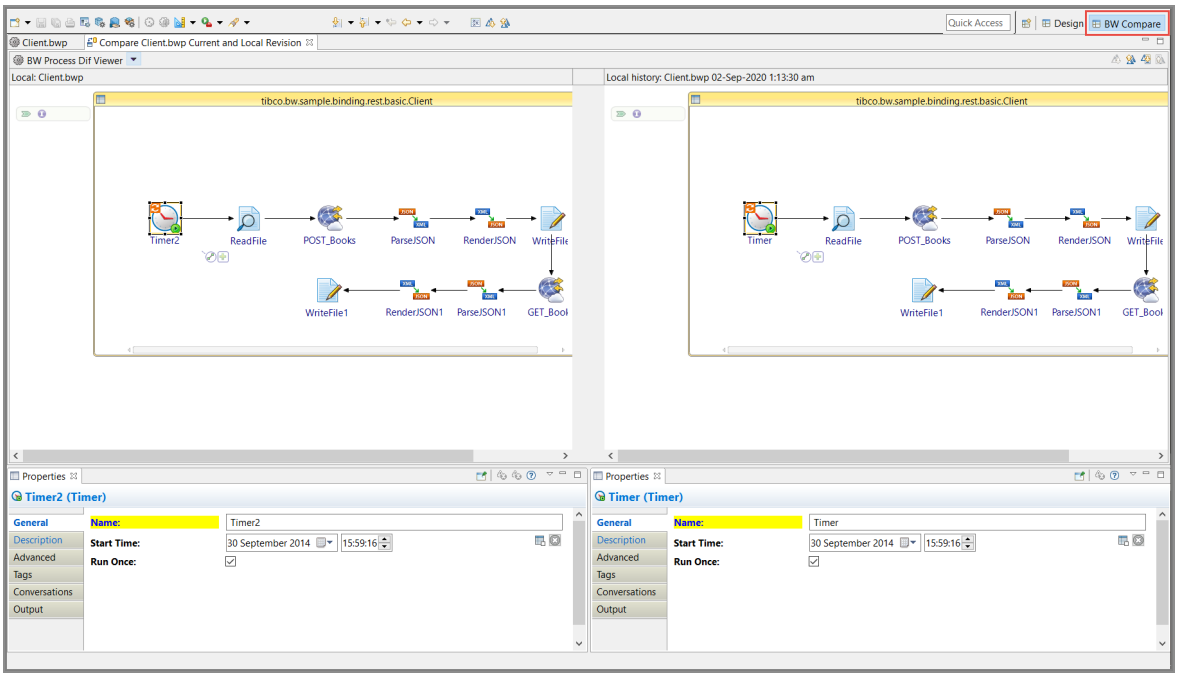


The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.



A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected process.



In the above example, there is a change in the **General** and **Description** tab hence they are marked in blue.

On the **General** tab, a change is made in the **Name** field, hence it is highlighted in yellow.

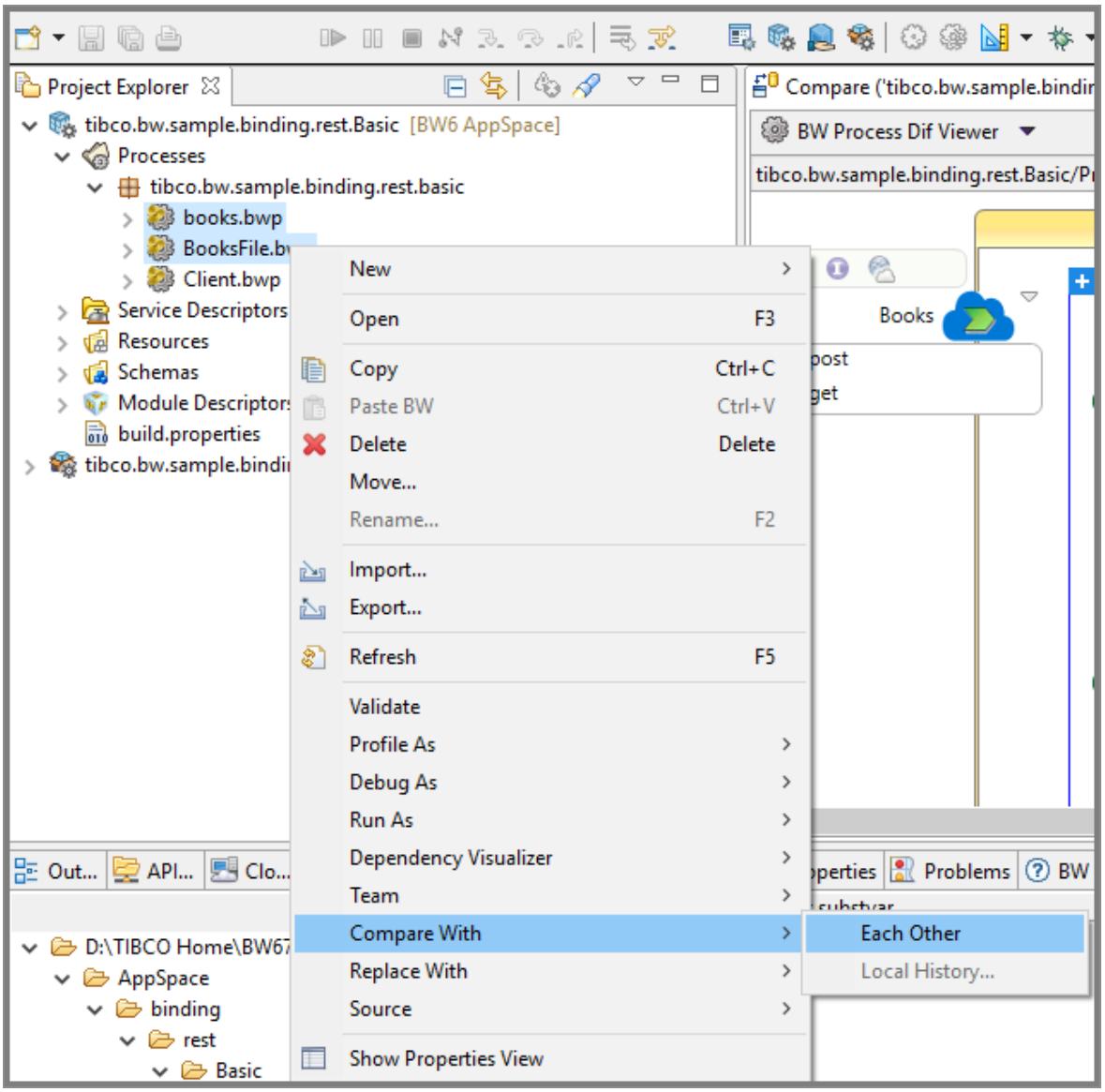
Compare with Each Other

Before you begin

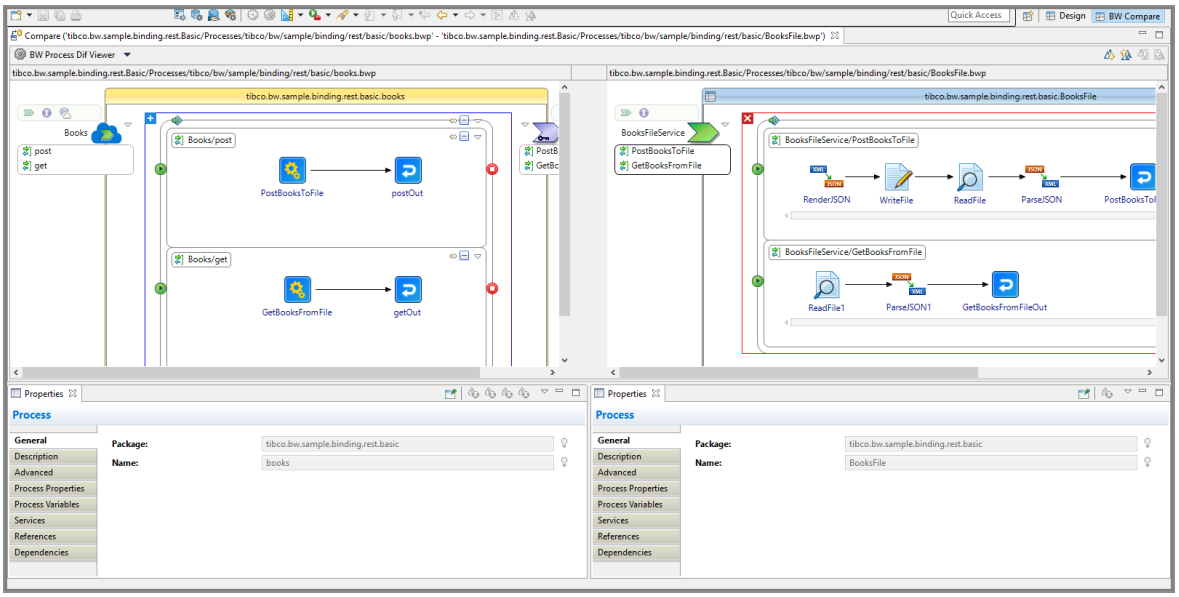
Import an existing sample or create a new project.

Procedure

1. In Project Explorer view, select the two different processes. Right-click and select **Compare With > Each Other**.



A **BW Compare** view is displayed, that displays the visual diff between the two different processes.



> **Caution:** If a process in an application is copied to the same application or shared module or a different application, the diff viewer does not highlight any changes unless the changes are explicitly made. If a new process is created with the same activities, the diff viewer highlights the changes for the activities even though the process flow is the same.

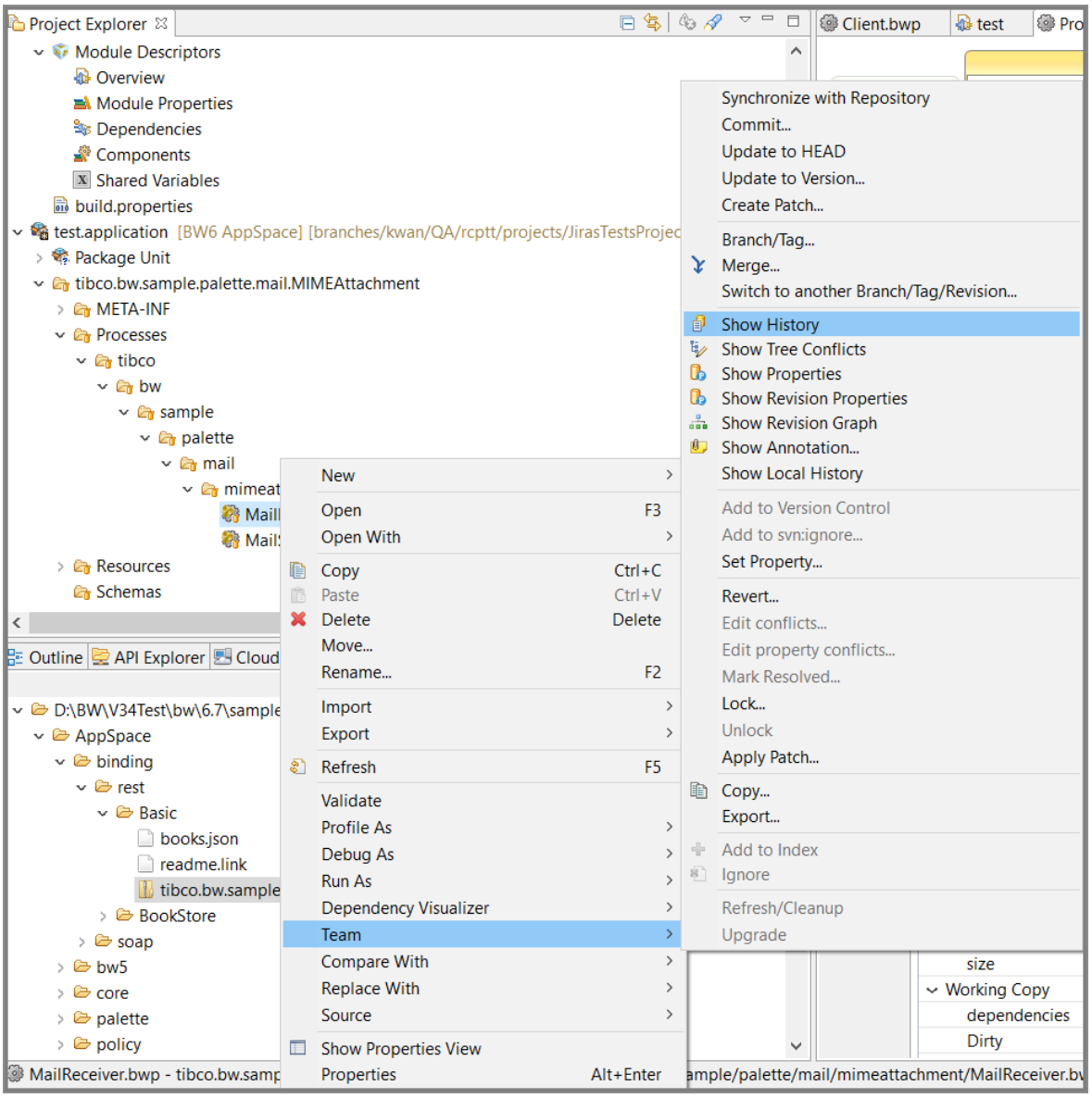
Compare with another revision from SVN Repository

Before you begin

- Import a project from SVN repository.
- Ensure the project is imported from SVN repository and a previous version of the project is saved and available to compare the two revisions.

Procedure

1. In the **Project Explorer** view, right-click on the selected process imported from the SVN repository and select **Team > Show History**



The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare** option.

   The **Compare** window is displayed.

3. In the **Compare** window, select the SVN versions to compare. Click **OK**.

   A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected process.

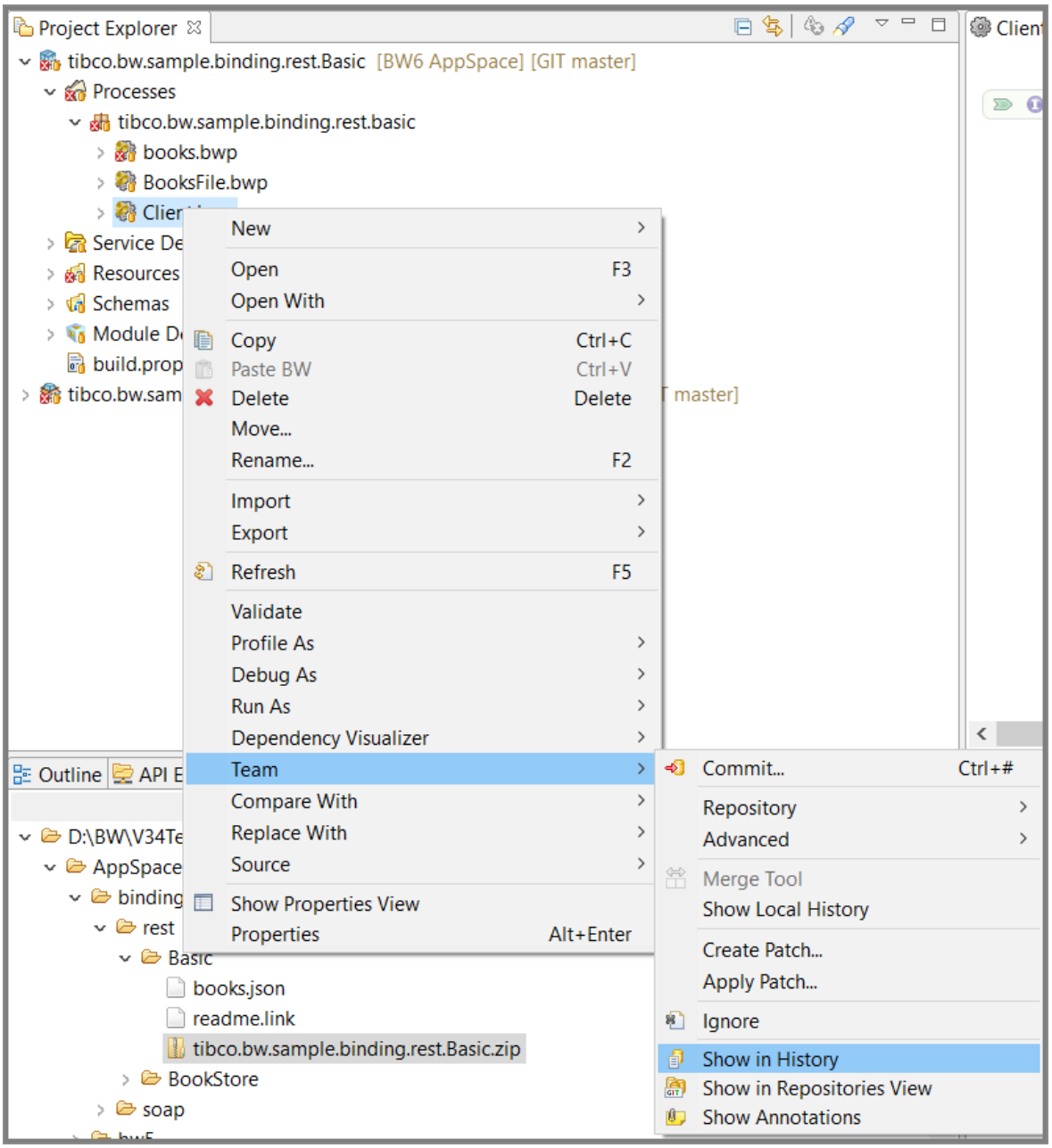
Compare with another revision from Git Repository

Before you begin

- Import a project from Git repository.
- Ensure the project imported from Git repository has a previous version and the project is saved and available to compare the two revisions.

Procedure

1. In Project Explorer view, right-click on the selected process imported from the Git repository and select **Team > Show History**



The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.

   A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected process.

Shared Resource Diff Viewer

The following are the different modes of operation by which diff viewer is viewed for shared resources.

- Compare with Local History
- Compare with Each Other
- Compare with another revision from SVN Repository
- Compare with another revision from Git Repository

Note: The Diff Viewer feature is supported only for the shared resources that are shipped with TIBCO BusinessWorks™ Container Edition.

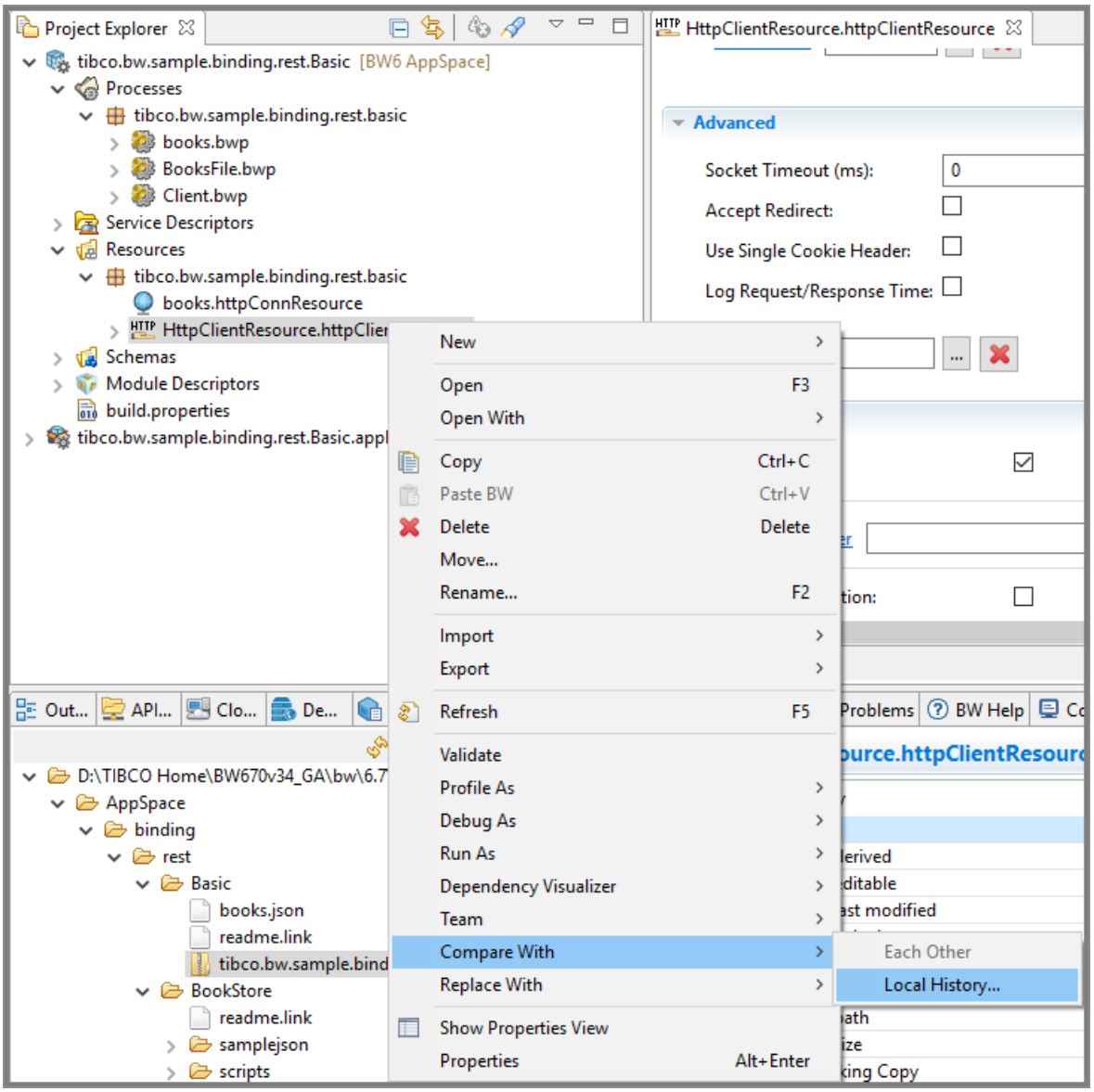
Compare with Local History

Before you begin

- Import an already existing sample or create a new project.
- The selected shared resource should be modified and saved at least once before comparing the two revisions. This ensures that there is a local history available in the workspace.

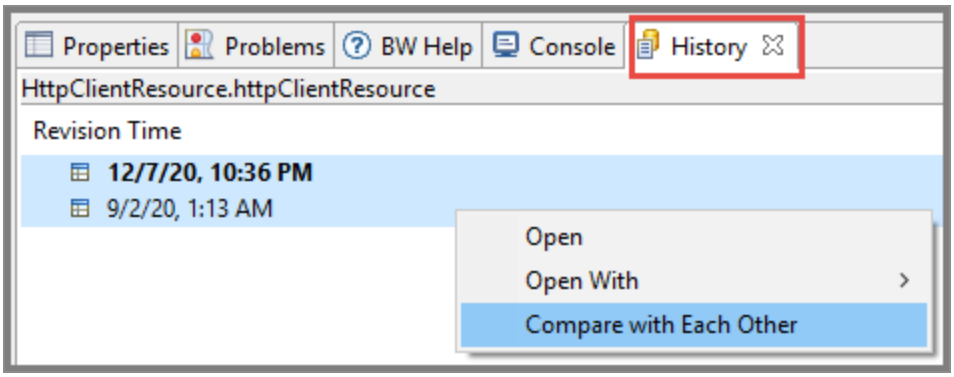
Procedure

1. In **Project Explorer view**, right-click on the selected shared resource and select **Compare With > Local History** or **Team > Show Local History**

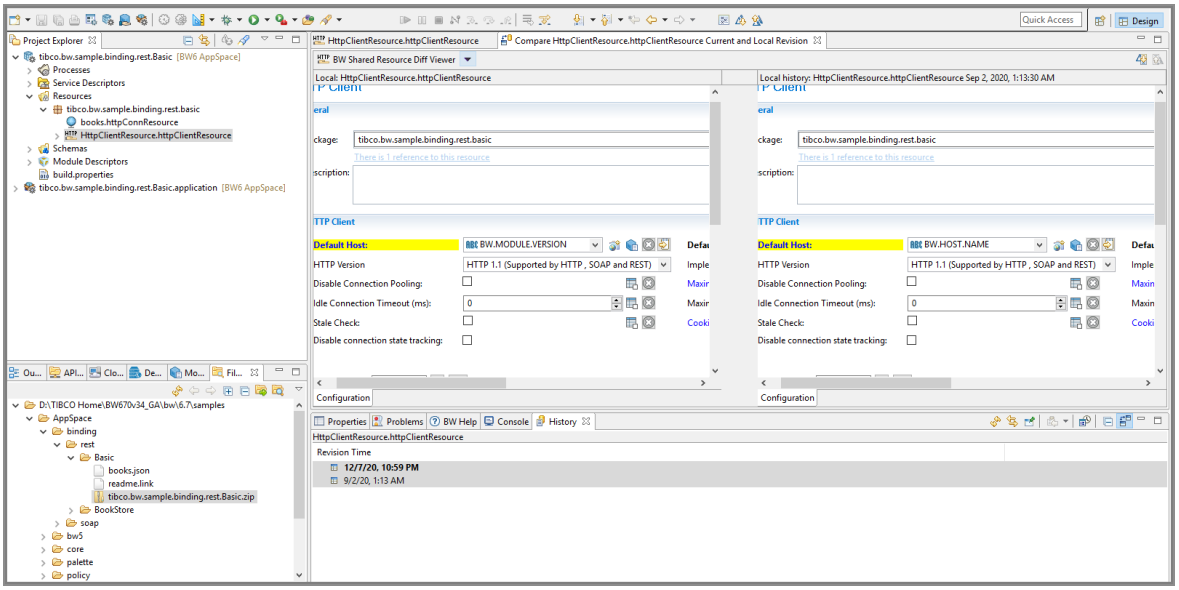


The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.



The **Design** view is displayed, that displays the visual diff between the two different revisions of the selected shared resource.



In the above example, since there is a change in the **Default Host**, **Maximum Total Connections** and **Cookie Policy** fields hence they are marked in blue and highlighted in yellow.

Compare with Each Other

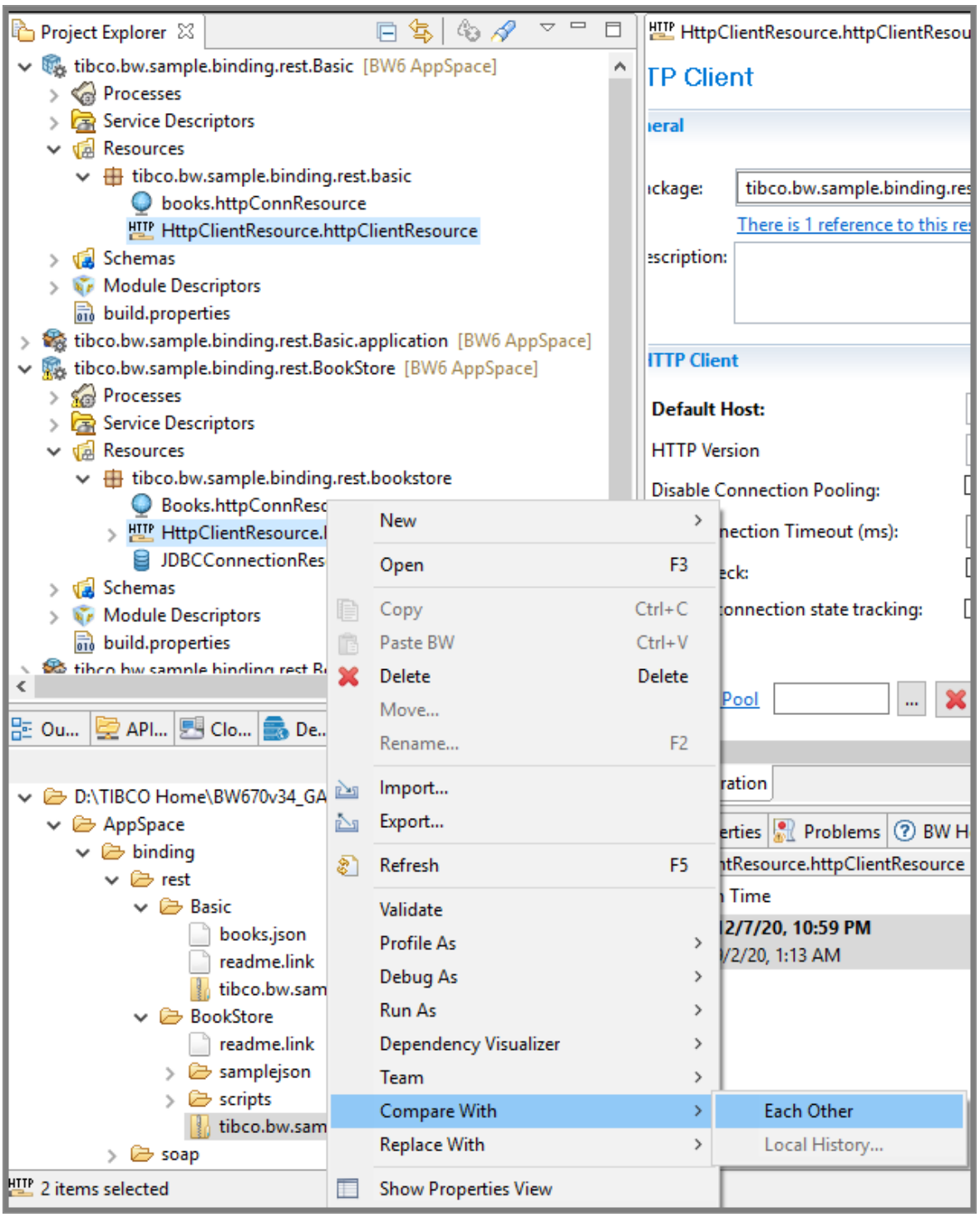
Before you begin

Import an existing sample or create a new project.

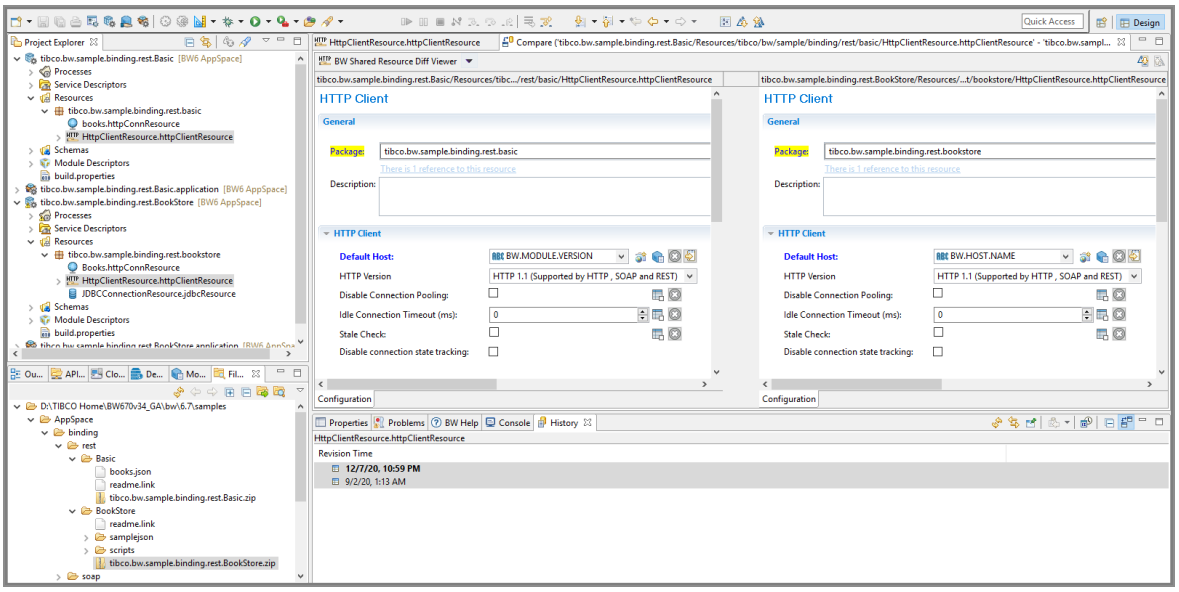
Procedure

1. In **Project Explorer** view, select two different shared resources. Right-click and select

Compare With > Each Other



A **Design** view is displayed, that displays the visual diff between the two different shared resources.



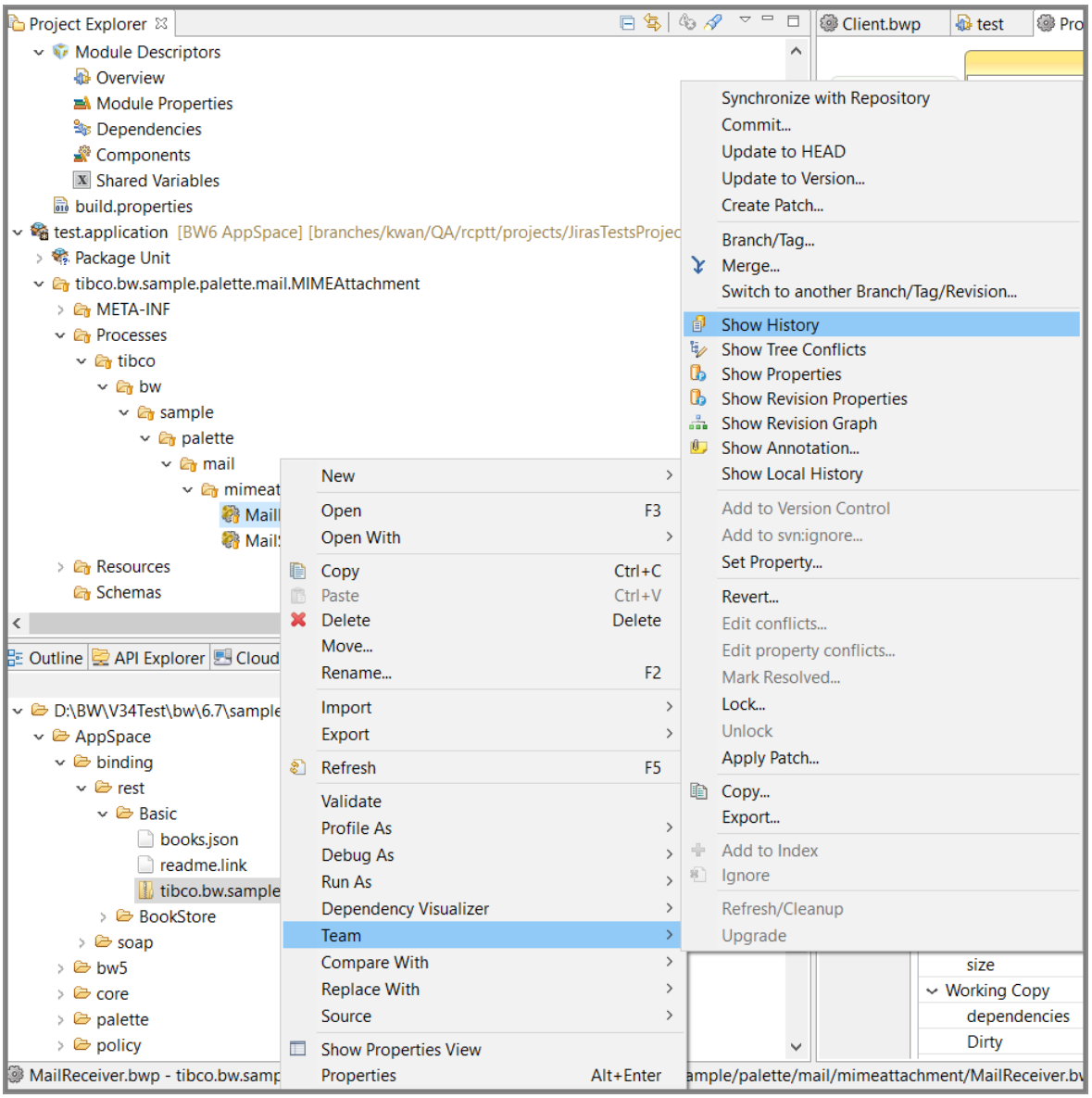
Compare with another revision from SVN Repository

Before you begin

- Import a project from SVN repository.
- Ensure the project is imported from SVN repository and a previous version of the project is saved and available to compare the two revisions.

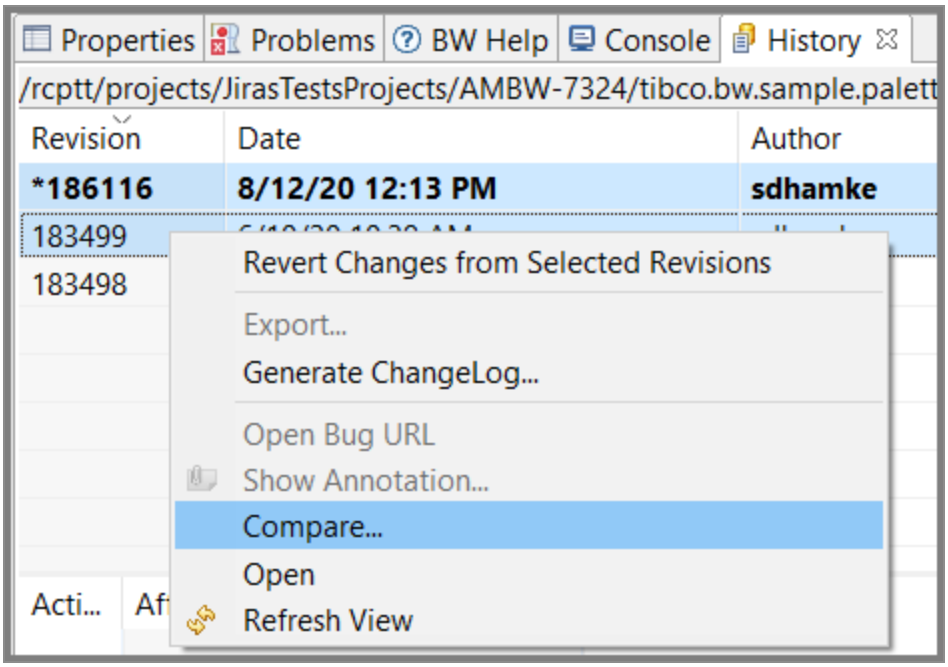
Procedure

1. In **Project Explorer** view, right-click on the selected shared resource imported from the SVN repository and select **Team > Show History**

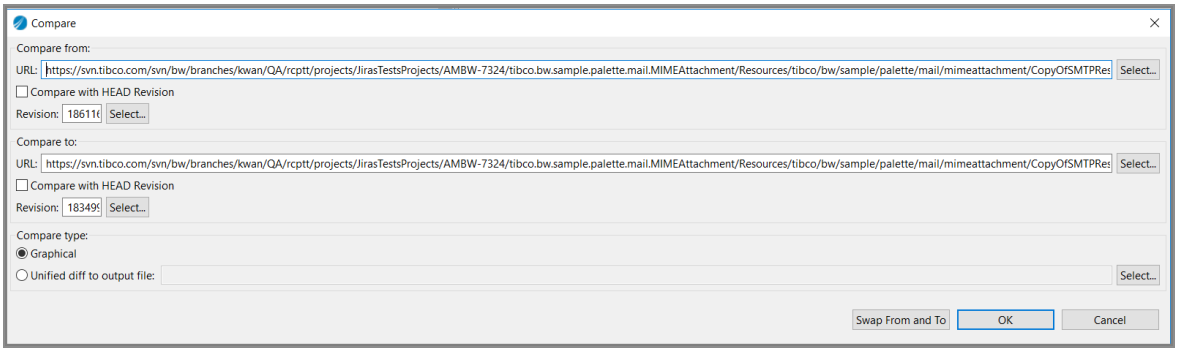


The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare** option.

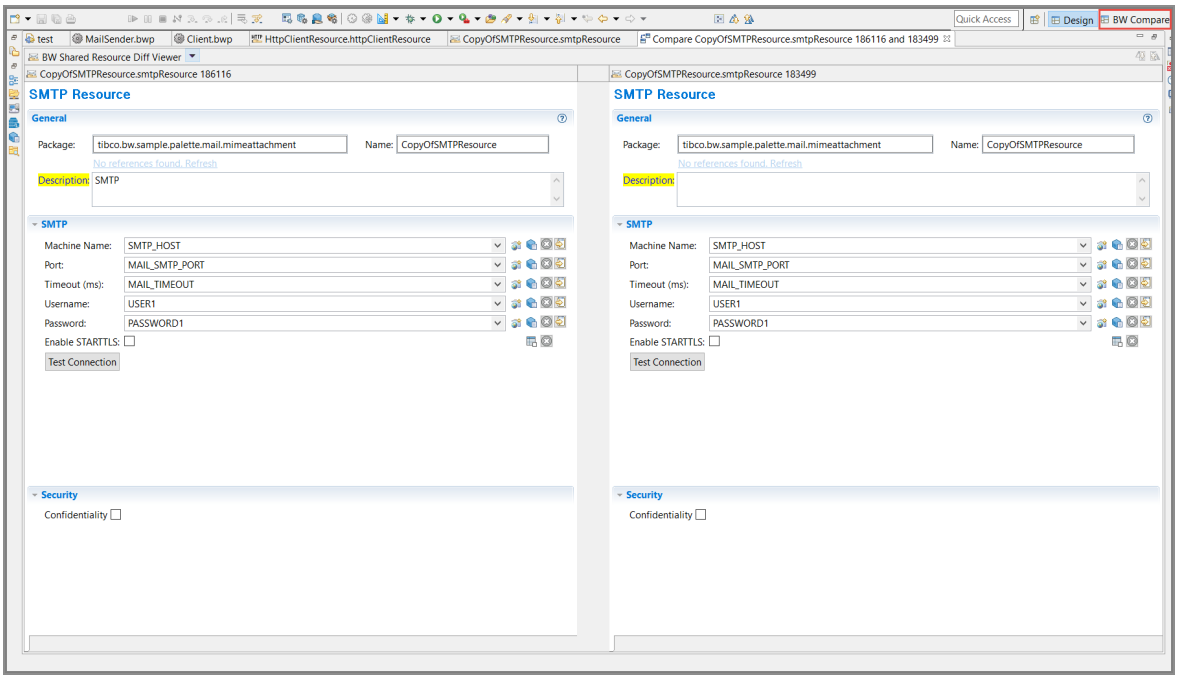


The **Compare** window is displayed.



3. In the **Compare** window, select the SVN versions to compare. Click **OK**.

   A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected shared resource.



In the above example, since there is a change in the **Description** field it is marked in blue and highlighted in yellow.

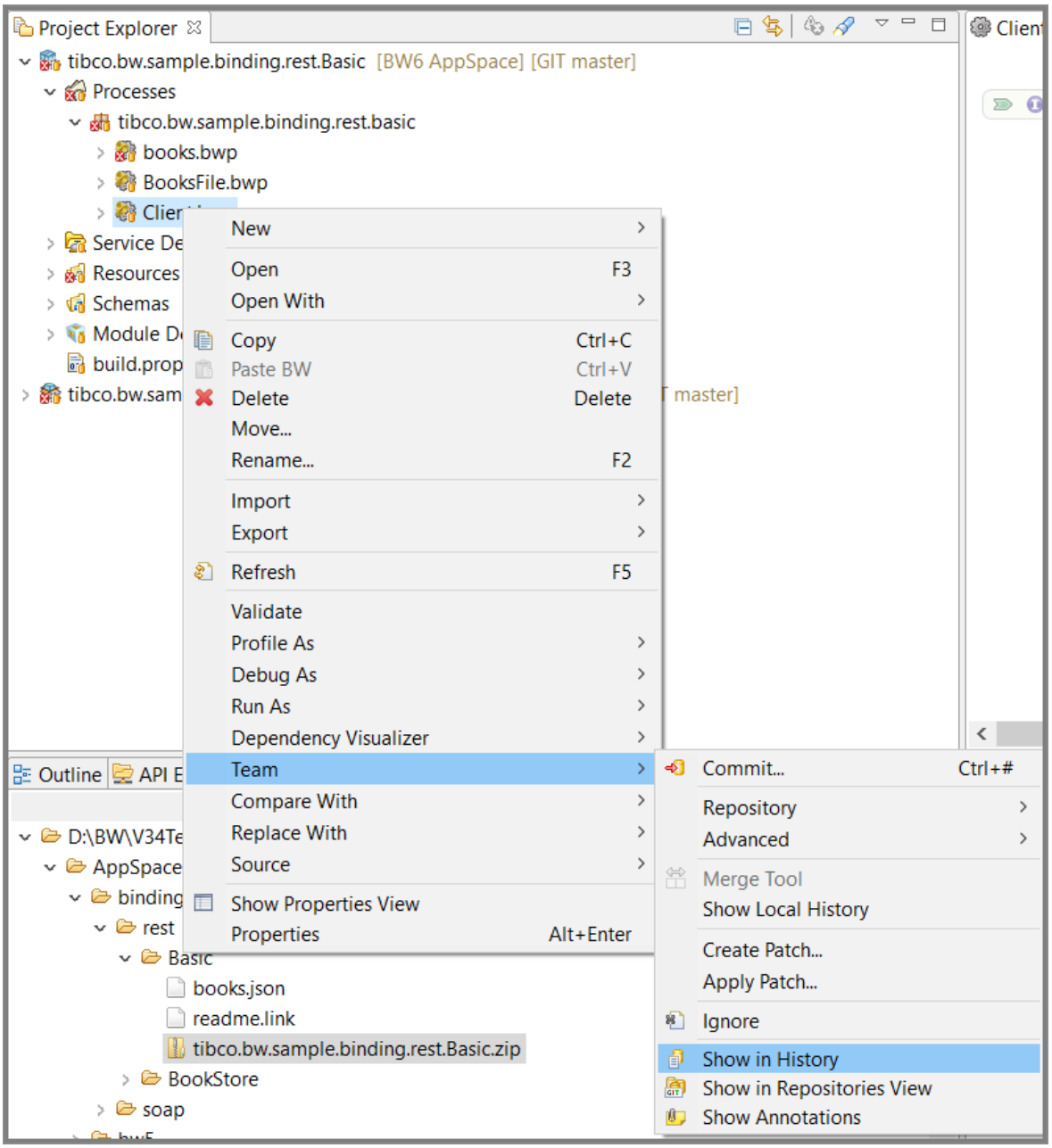
Compare with another revision from Git Repository

Before you begin

- Import a project from Git repository.
- Ensure the project imported from Git repository has a previous version and the project is saved and available to compare the two revisions.

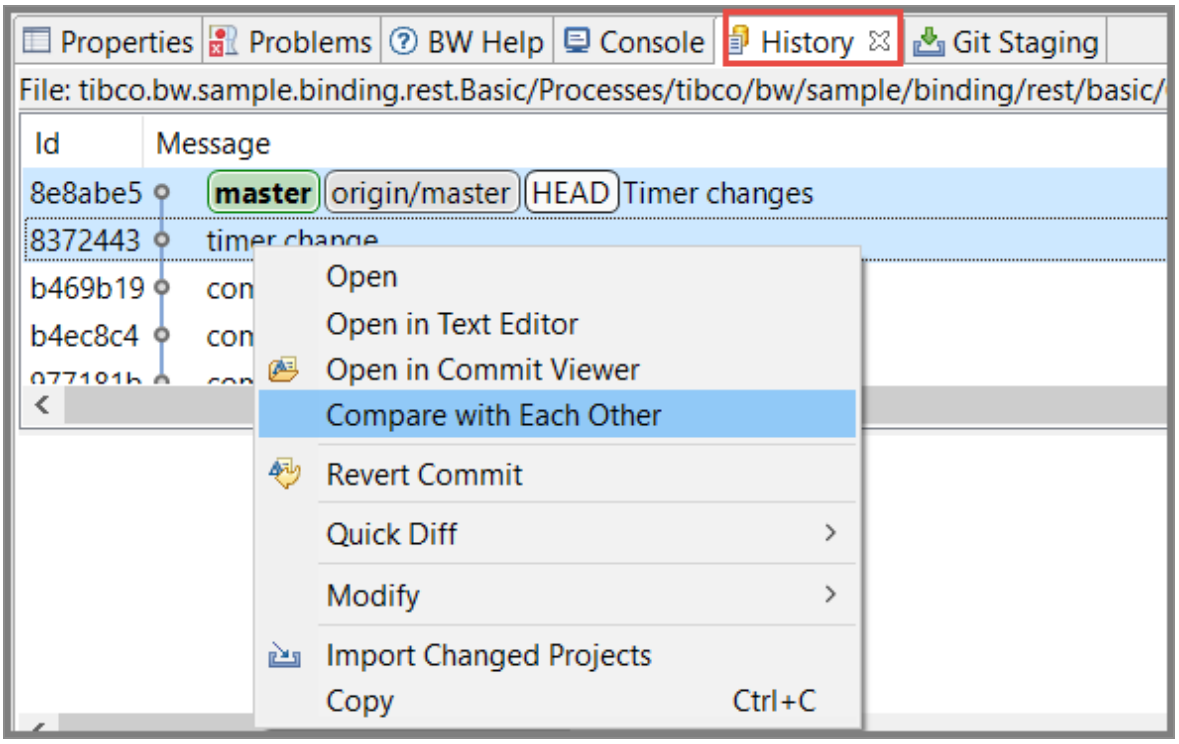
Procedure

1. In **Project Explorer** view, right-click on the selected shared resource imported from the Git repository and select **Team > Show History**

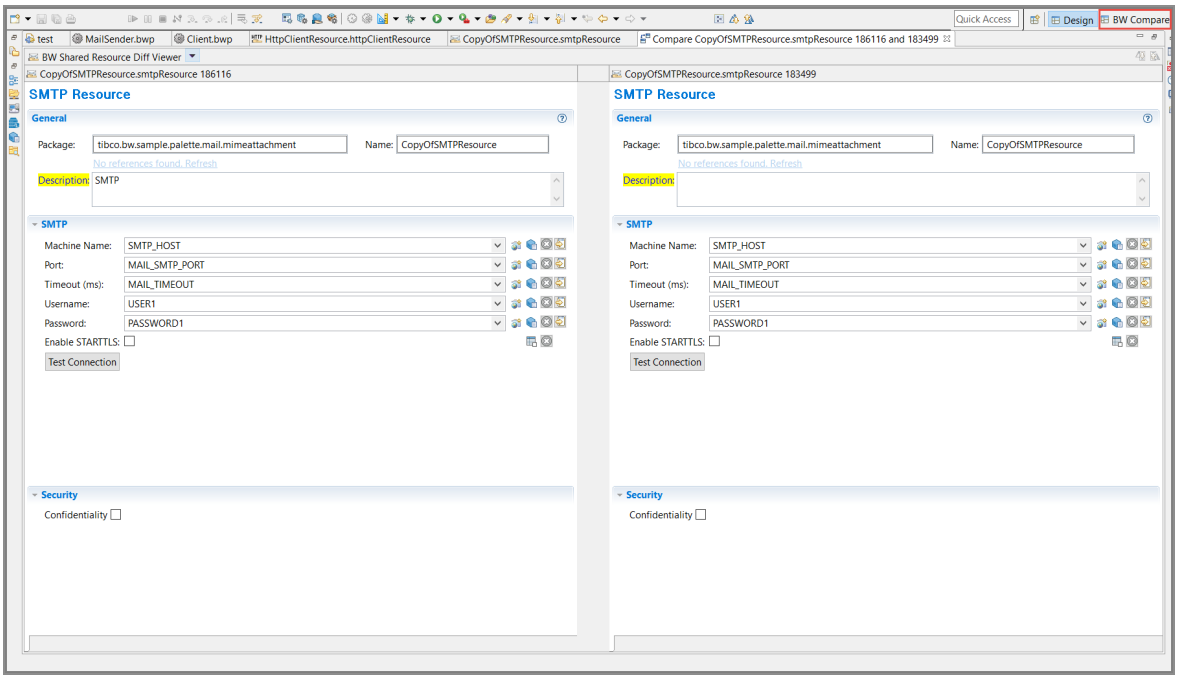


The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.



A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected shared resource.



In the above example, since there is a change in the **Description** field it is marked in blue and highlighted in yellow.

Module Property Diff Viewer

The following are the different modes of operation by which diff viewer is viewed for module properties.

- Compare with Local History
- Compare with Each Other
- Compare with another revision from SVN Repository
- Compare with another revision from Git Repository

Compare with Local History

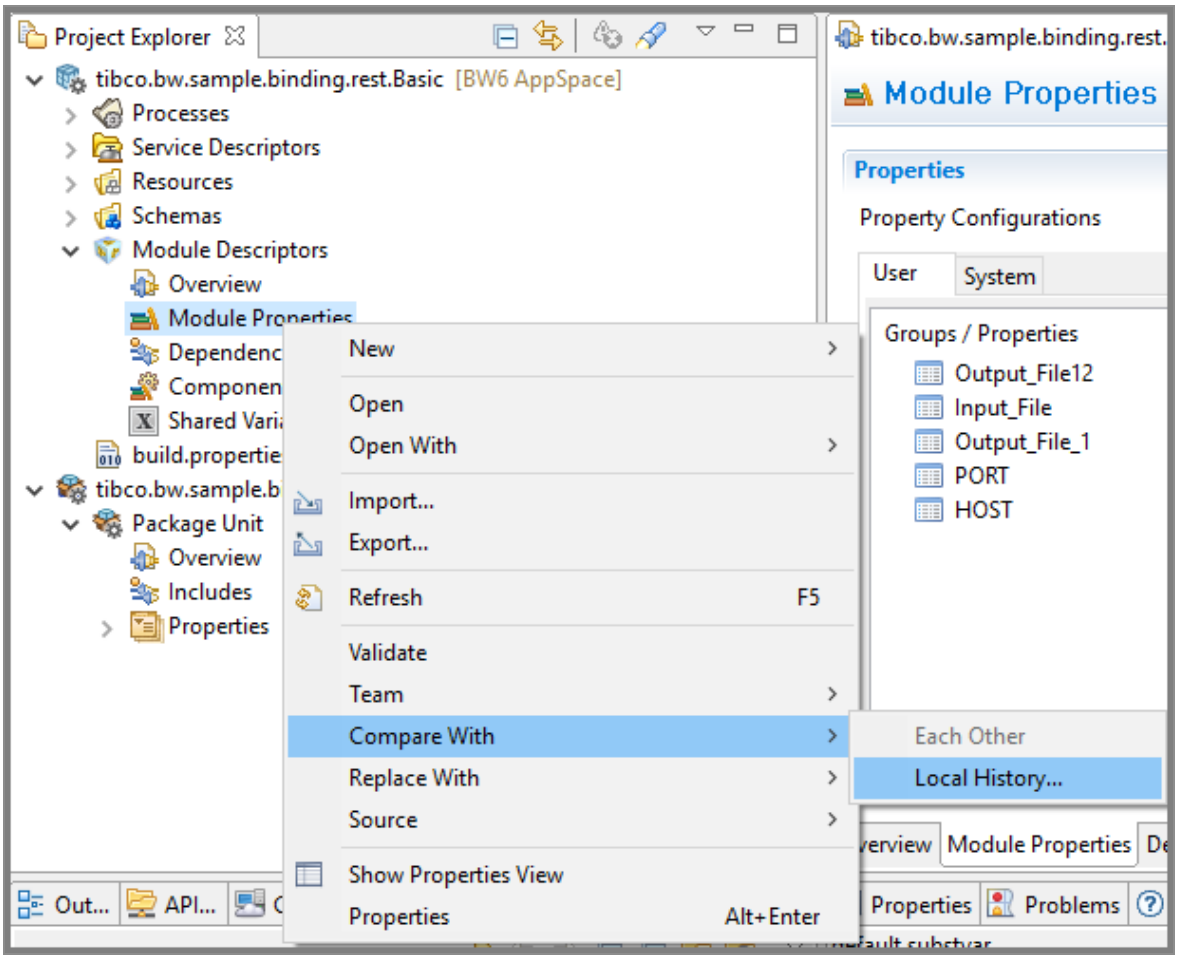
Before you begin

- Import an already existing sample or create a new project.
- For the selected module property, one of the should be modified and saved at least

once before comparing the two revisions. This ensures that there is a local history available in the workspace.

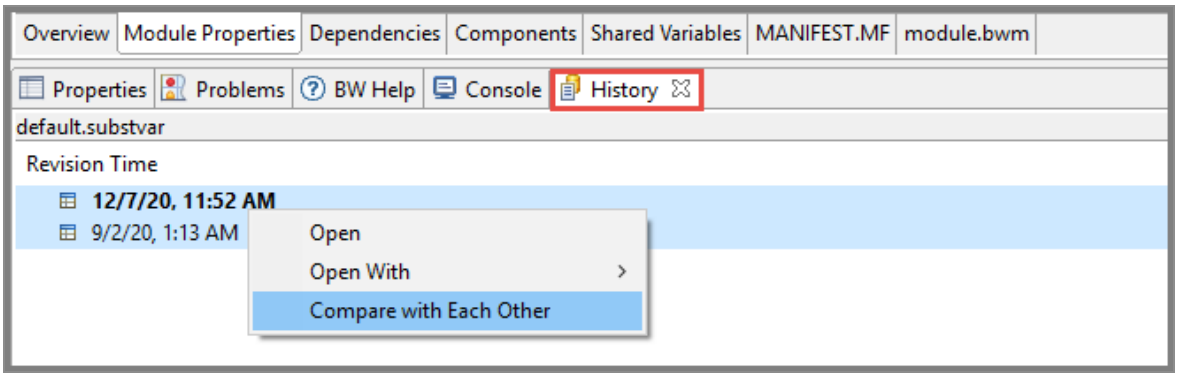
Procedure

1. In **Project Explorer** view, on the selected module property right-click and select **Compare With > Local History** or **Team > Show Local History**

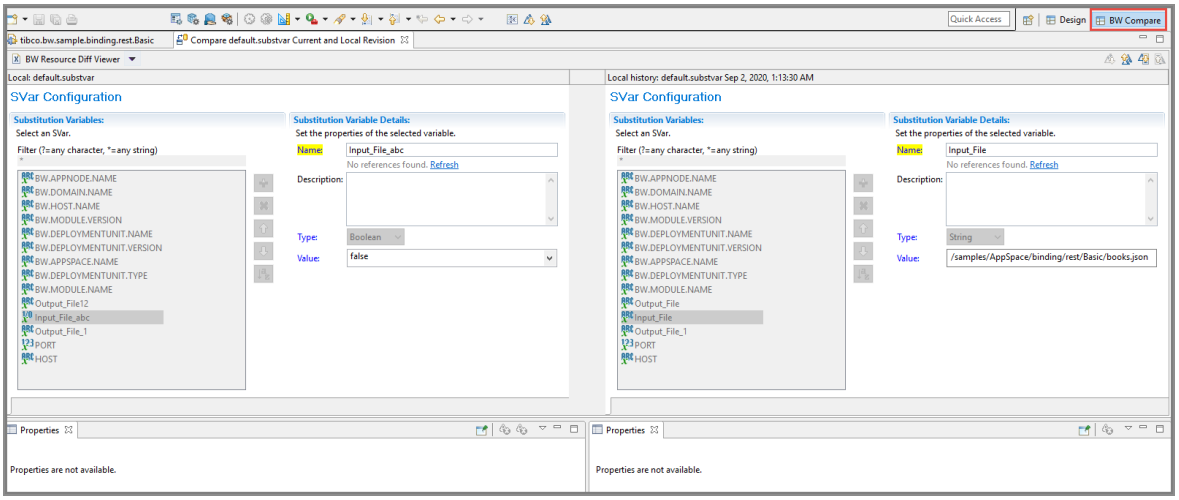


   The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.



A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected module property.



In the above example, since there is a change in the properties of the **Name**, **Type** and **Value** fields, they are marked in blue and highlighted in yellow.

For **Type** field, since there is a change in the data type from Boolean to String, TIBCO Business Studio™ for BusinessWorks™ automatically changes the value of the data type in **Value** field as well, although there was no change explicitly made to the **Value** field.

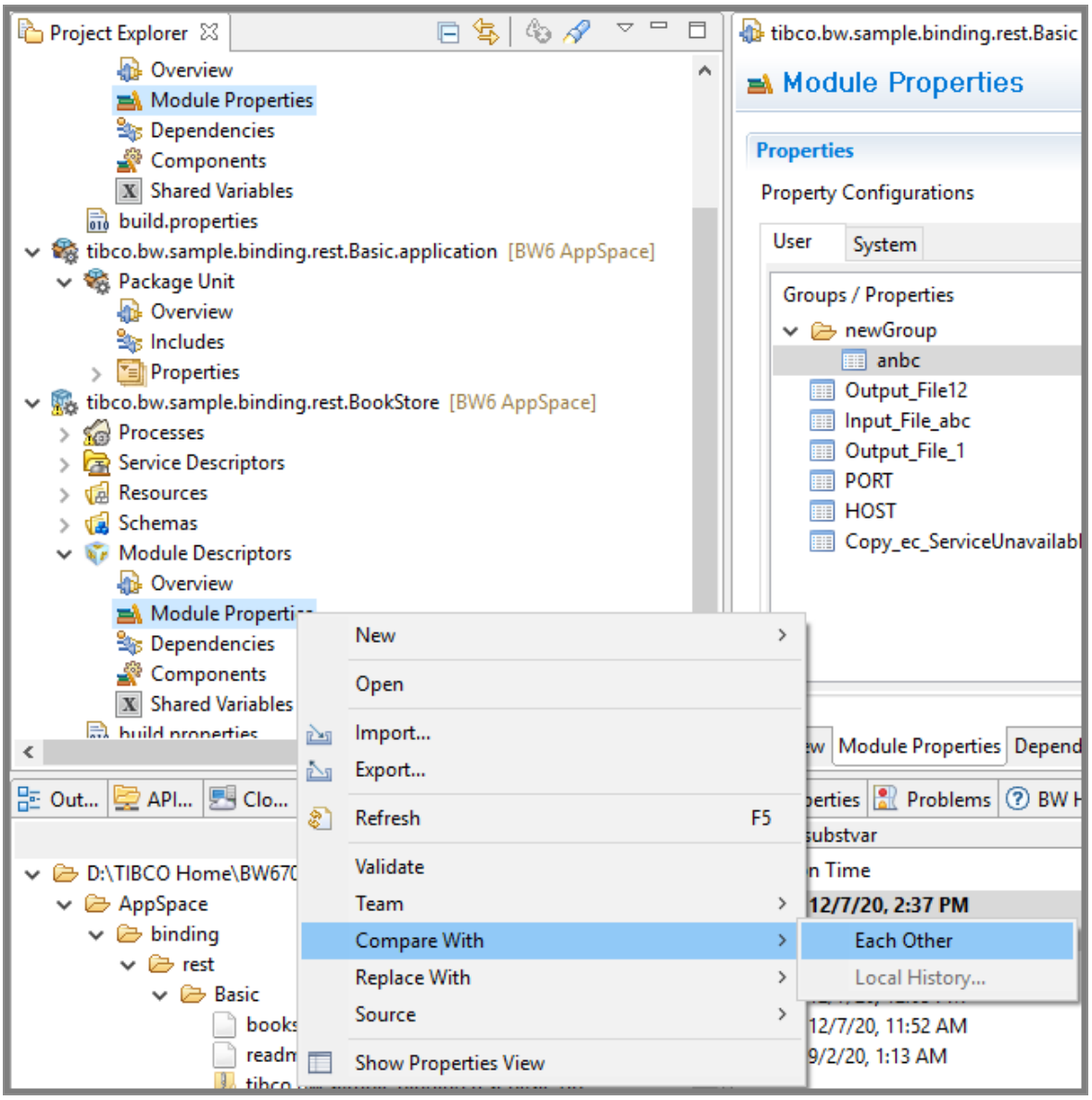
Compare with Each Other

Before you begin

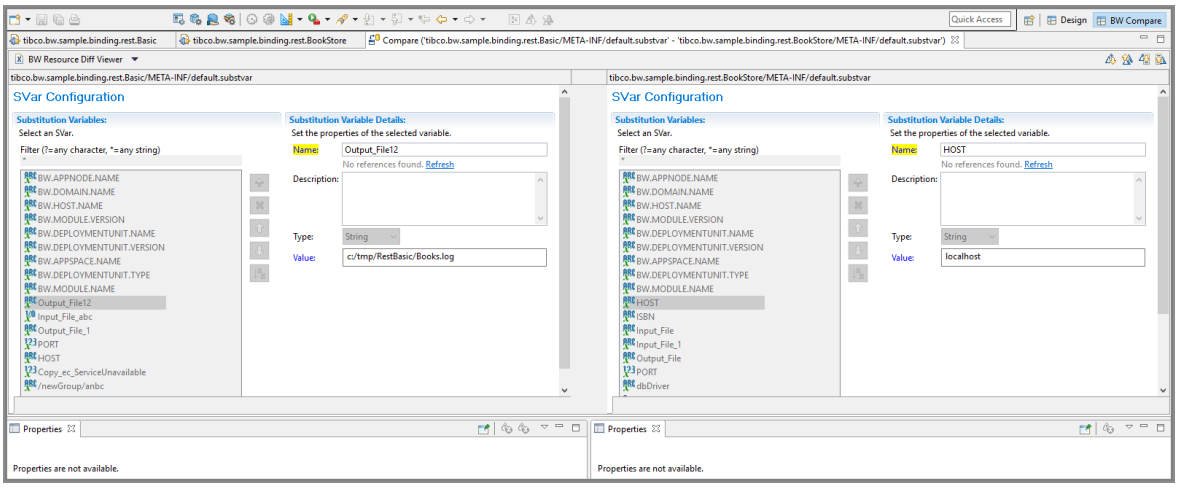
Import an existing sample or create a new project.

Procedure

1. In **Project Explorer** view, select the two different module properties of two different projects. Right-click and select **Compare With > Each Other**.



A **BW Compare** view is displayed, that displays the visual diff between the two different module properties of two different projects.



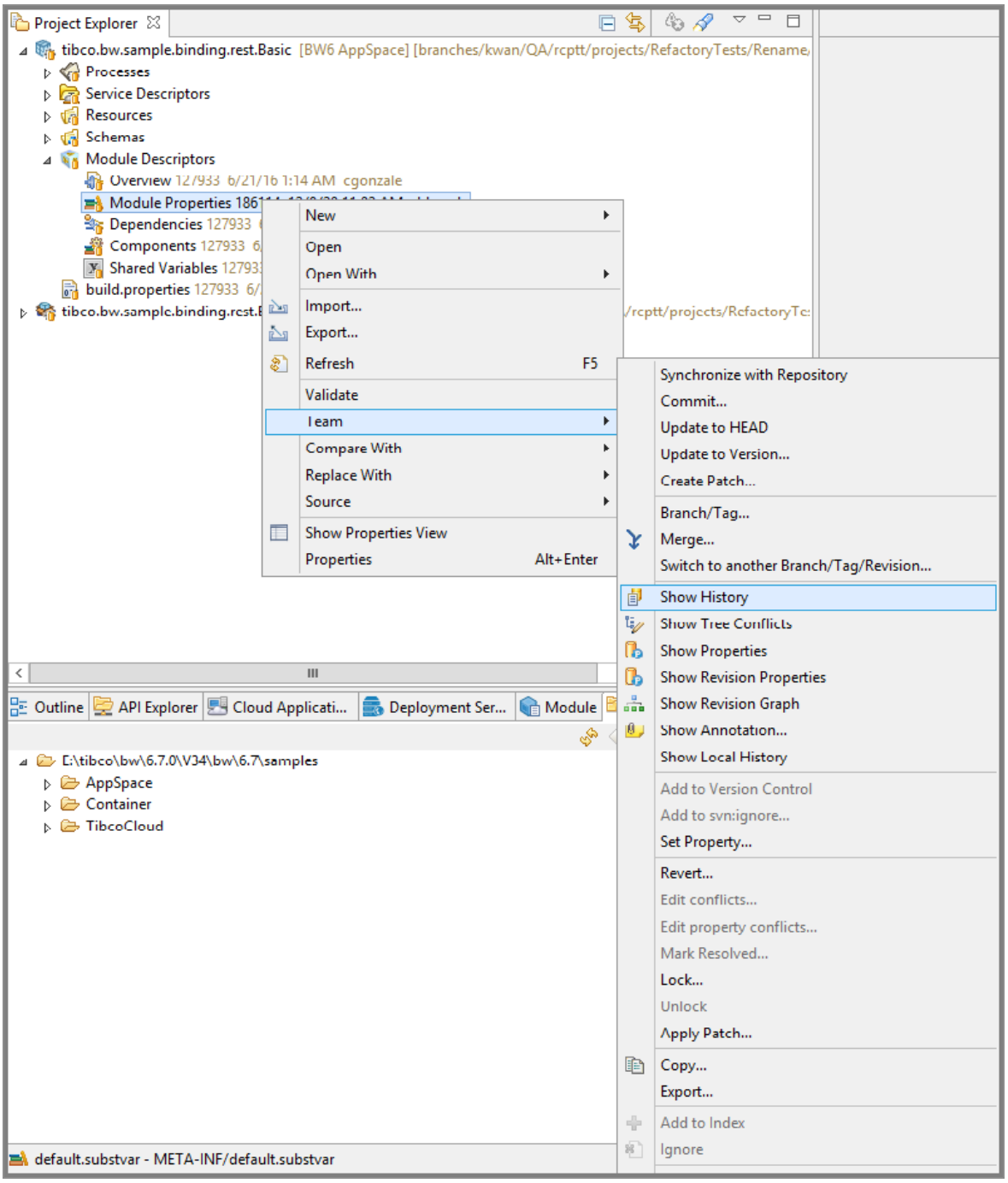
Compare with another revision from SVN Repository

Before you begin

- Import a project from SVN repository.
- Ensure the project is imported from SVN repository and a previous version of the project is saved and available to compare the two revisions.

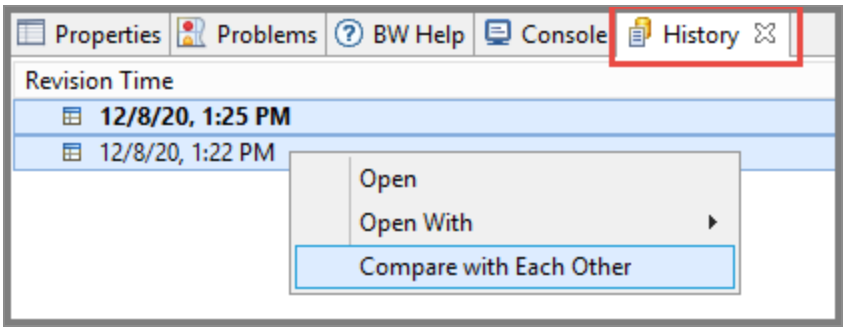
Procedure

1. In **Project Explorer** view, right-click on the selected module property imported from the SVN repository and select **Team > Show History** or **Compare with > Local History**

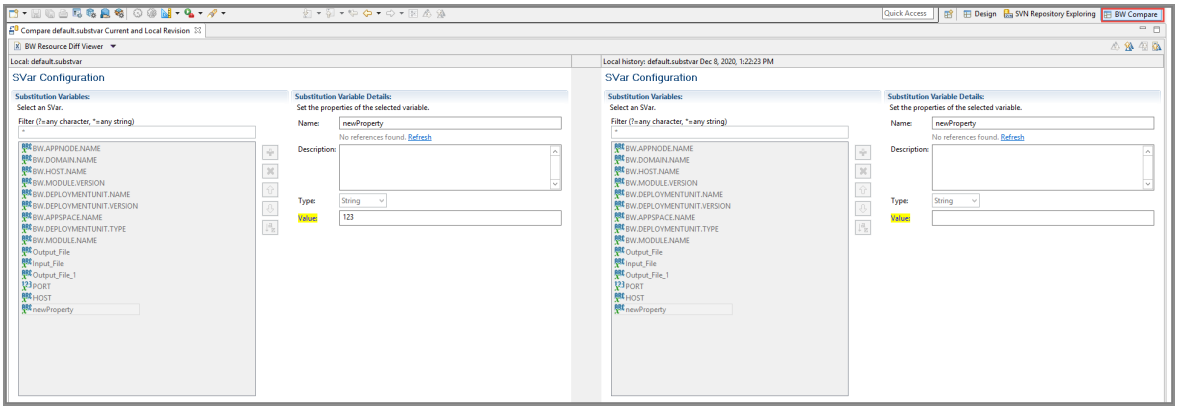


The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.



A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected module property imported from SVN repository.



In the above example, since there is a change in the properties of the **Value** field, it is marked in blue and highlighted in yellow.

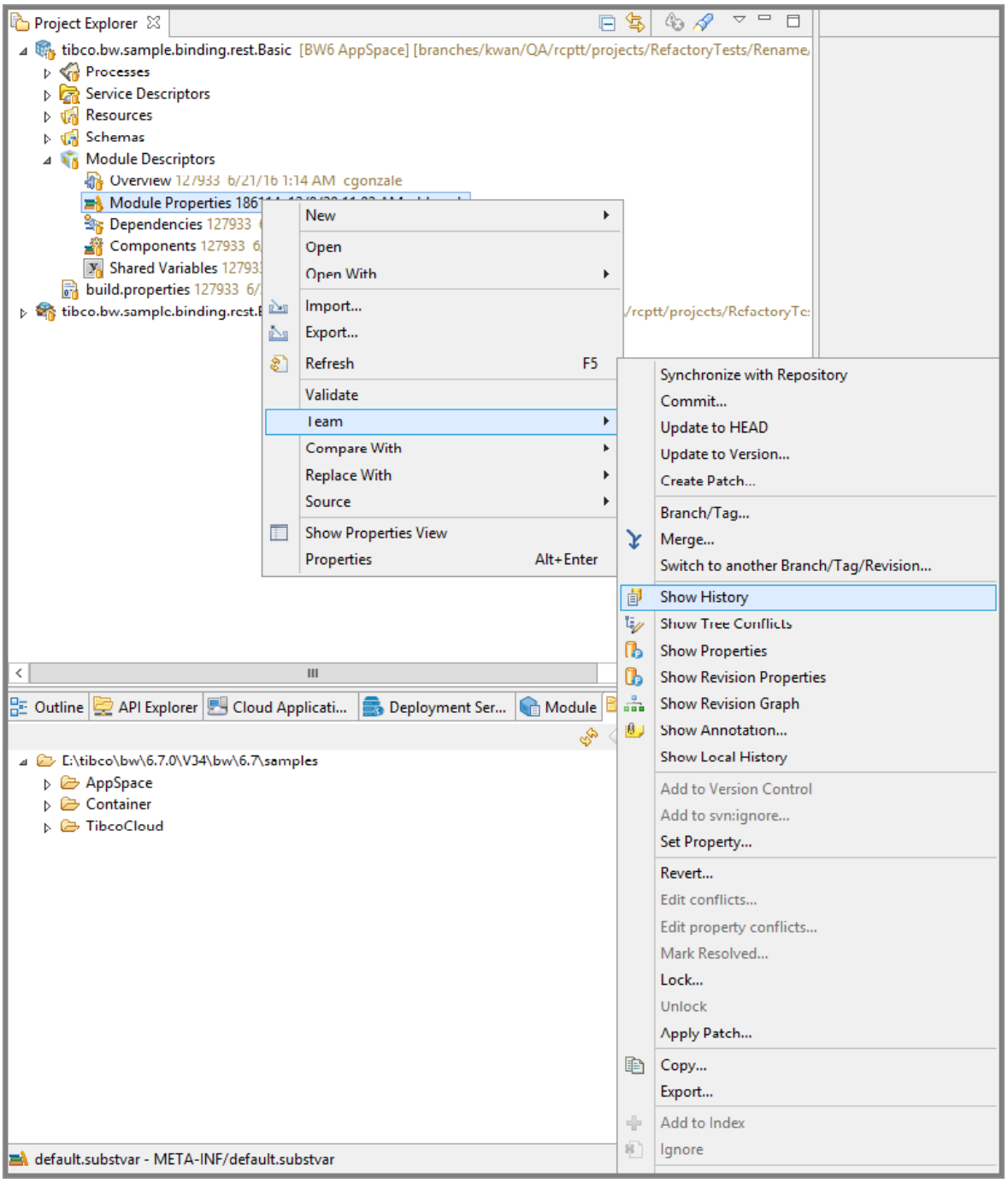
Compare with another revision from Git Repository

Before you begin

- Import a project from Git repository.
- Ensure the project imported from Git repository has a previous version and the project is saved and available to compare the two revisions.

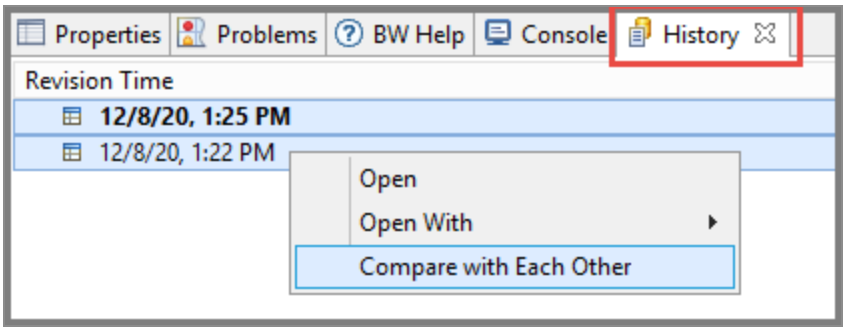
Procedure

1. In **Project Explorer** view, right-click on the selected module property imported from the Git repository and select **Team > Show History**

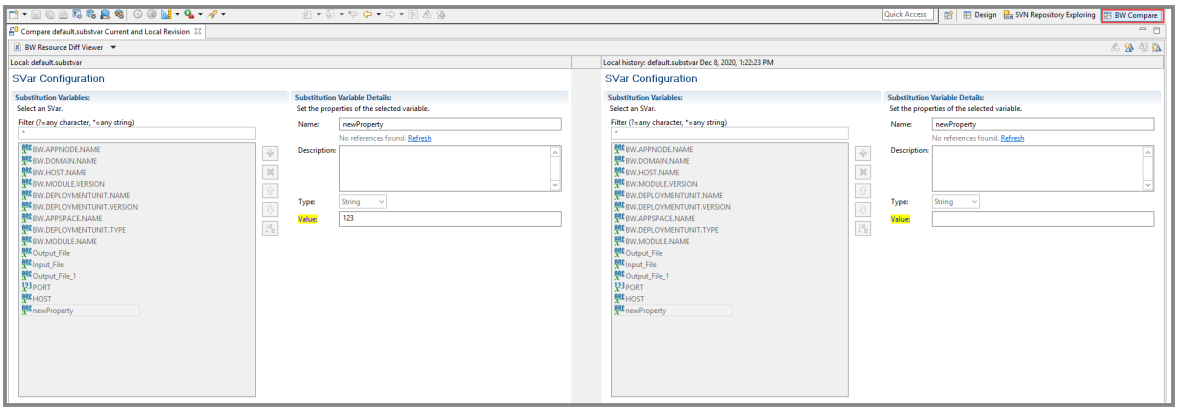


The **History** tab is displayed.

2. On the **History** tab, select the two different revisions to compare. Right-click and select **Compare with Each Other** option.



A **BW Compare** view is displayed, that displays the visual diff between the two different revisions of the selected module property imported from Git repository



In the above example, since there is a change in the properties of the **Value** field, it is marked in blue and highlighted in yellow.

Important:

- If the order in which the modules properties displayed is changed, after any add or delete operation performed on them, the diff viewer feature does not compare the correct module properties with each other.
- Since the module properties are mapped directly with application properties, for any number of changes that are made in the module properties, the same changes are reflected in the application properties as well.
- For module properties of shared resources type, the data type is always denoted as a String value.

Merge

The merge feature is used to merge changes made from one branch to another branch or from the main branch to feature branch and vice versa.

For version control system, where a branch is used to maintain separate lines of development, at some point it is required to merge the changes from one branch to another or main branch.

To use the merge option, right-click on an application or application module, select **Team > Merge**. The merge feature is supported for both GIT and SVN.

Merge using SVN

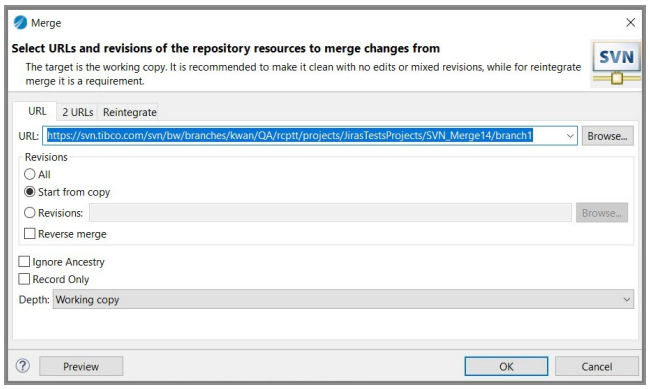
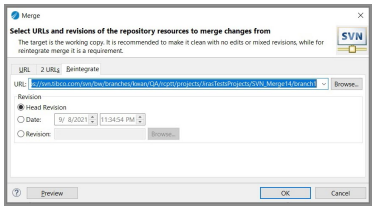
The merge feature using SVN merges changes from a single path or URL of a branch to a working copy of another branch where the changes need to be merged.

To use SVN merge, ensure the SVN plug-in is installed. To install the plug-in, see Configuring TIBCO Business Studio for BusinessWorks with SVN plug-in.

Merge Modes:

The following are the supported modes for SVN merge:

Merge Mode	Description
Single path or URL merge	• This mode merges changes from all the eligible revision into the branch from its immediate ancestor. • In this mode, you can choose to merge all the eligible changes at once, called 'sync' merge or merge explicitly defined set of changes using revision selection controls, called 'cherrypick' merge. • This option is used to catch-up feature branch with changes in trunk or another branch.

Merge Mode	Description
	
Reintegrate merge	• This mode is used to merge changes from a feature branch back into the trunk. By default, the head revision is selected. • Any specific revision can be selected using the Revision option. 

Follow the steps to merge changes from branch A to branch B:

1. In the Project Explorer, right-click on the project and select **Team > Merge**. The **Merge** window is displayed.
2. On the **Merge** window select one of the merge modes, **URL merge** or **Reintegrate merge**.
3. In the **URL** field, add the source URL from where you need to merge.
4. Select any one type of revision to use in merge.
    - **All**: Consider all eligible revisions for merge.
    - **Start from Copy**: Merge the changes from the recent revision copy.
    - **Revisions**: Specify a single or a set of revisions.

    By default, **the Start from Copy** option is selected.
5. Select any one of the **Depth** options. This specifies the depth of the merge to be

covered. The following are the options:

- **Only Folder**.
- **Files in Folder**
- **Directly Children**
- **Recursively**
- **Working Copy**

By default, the **Working Copy** option is selected.

6. Select the **Preview** button to inspect the consequences of the merge operation without applying any actual changes to the working copy.
7. Click **Ok** to complete the merge.

Note:
- As a best practice before merging, ensure the working copy does not have any changes and should be clean and updated.
- For a successful merge, avoid conflicts by using a separate branch and keep the branch updated regularly.

Merge using GIT

The merge feature using GIT, merges changes across two branches or from feature branch to main branch and vice versa.

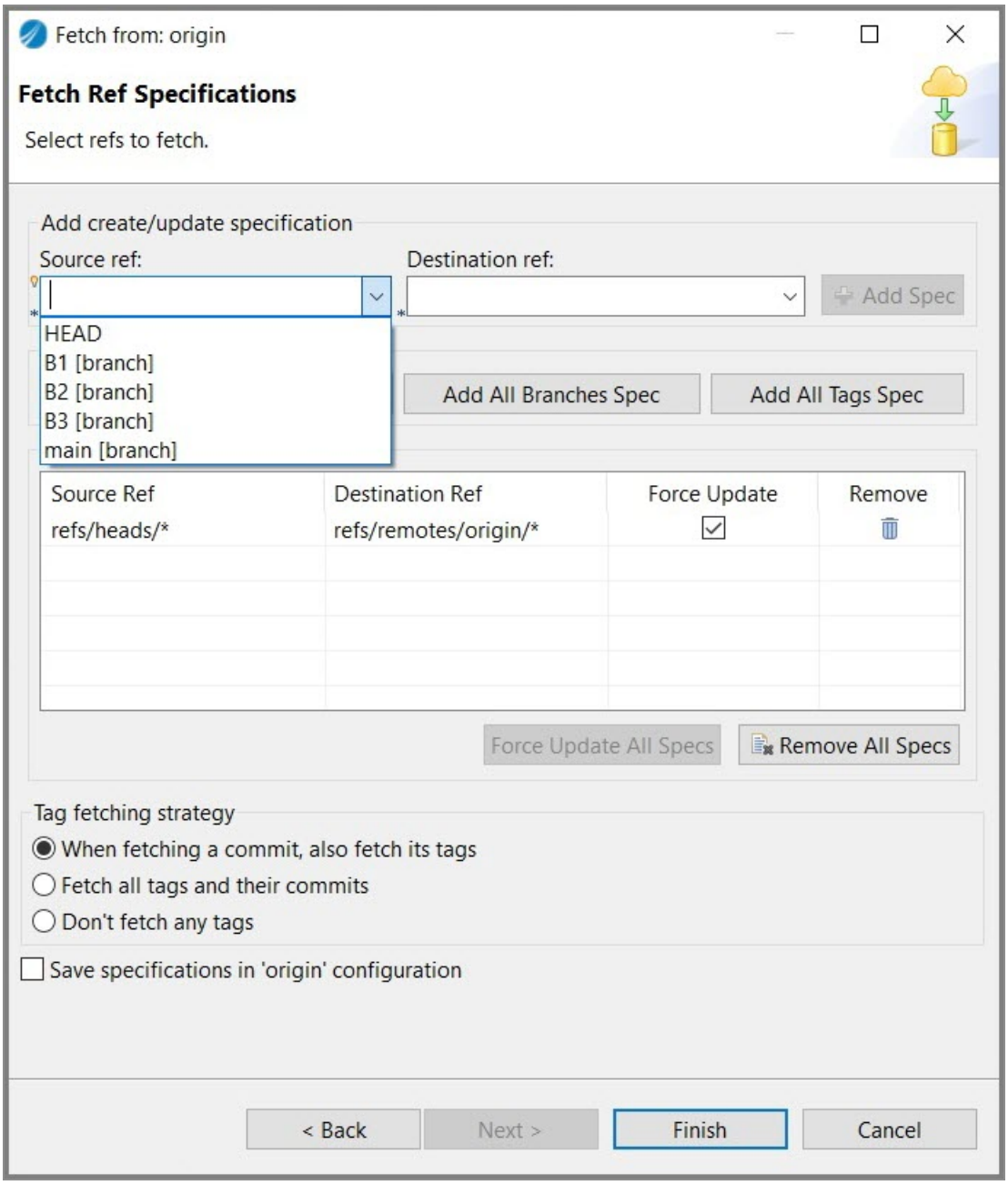
Before you begin

Before merging from one branch to another, you need to configure source branch in fetch list and fetch changes from the source branch. This is required to fetch the unmerge changes from the source branch.

Follow the steps to add the source branch to the fetch list:

1. In the Project Explorer, right-click on a project and select **Team > Remote > Fetch From**. The **Fetch from Another Repository** window is displayed.
2. In the **Fetch from Another Repository** window, configure the repository by enabling the **Configured remote repository** button, then click **Next**. The **Fetch from: origin** window is displayed.
3. In the **Fetch from: origin** window, select the source branch from the **Source ref:** drop-down in the **Add create/update specification** section, then click **Finish**.
4. In the **Fetch Results:** window select the **Configure...** button and then click **Add**.
5. In the **Source** field, select the name of the source branch and click **Next > Finish > Save and Fetch**.

   All the unmerge changes of the source branch are now fetched and displayed in the **Merge** window.



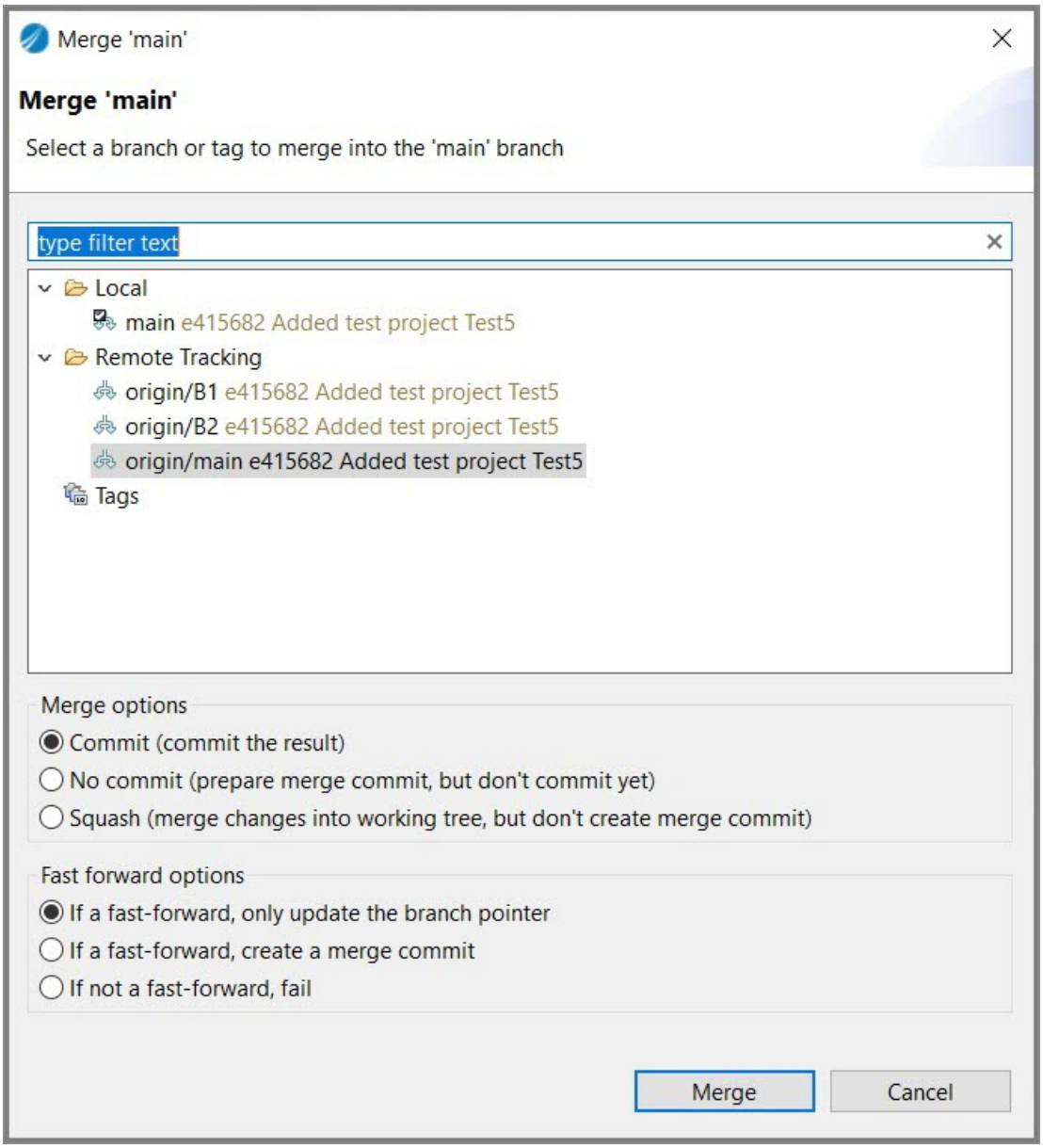
Follow the steps to merge changes from Branch A to Branch B:

1. In the Project Explorer, right-click on a project and select **Team > Merge**. After fetching, all the changes or commits of the source branch which are not merged are displayed in the fetch list.
2. In the **Merge Options** section, select one of the merge options. By default, the **Commit (Commit the result)** option is selected.

3. In the **Fast Forward** options, select one of the fast forward options. By default, the **If a fast forward, only update the branch pointer** option is selected.
4. Click **Merge**.

   The merge is completed, and the Merge Result is displayed.

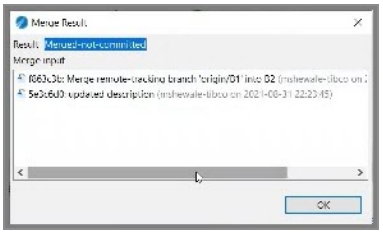
> **Note:**
> - As a best practice before merging, ensure the working copy does not have any changes and should be clean and updated.
> - For a successful merge, avoid conflicts by using a separate branch and keep the branch updated regularly.



Merge Options:

The following are the merge options for GIT:

Merge Options	Description and Merge Result
Commits	This option on merging prepares the commit message and commits the changes.

Merge Options	Description and Merge Result
	
No Commit	This option on merging only prepares the commit message but does not commit the changes. 
Squash	This option on merging only adds the changes to the working copy but does not prepare any commit. 

Fast Forward Options:

In GIT, when two branches are not diverged and there is a linear path from the target branch to the source branch , then GIT runs a fast forward merge.

Fast Forward Options	Description
If fast forward, only update pointer	For this option, instead of creating a new merge commit, the head pointer is moved to the current commit.
If fast forward, create a merge commit	For this option, a snapshot of all the merge commit history and its details is created.
If not a fast-forward, fail	For this option, the merge is not performed and is aborted.

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

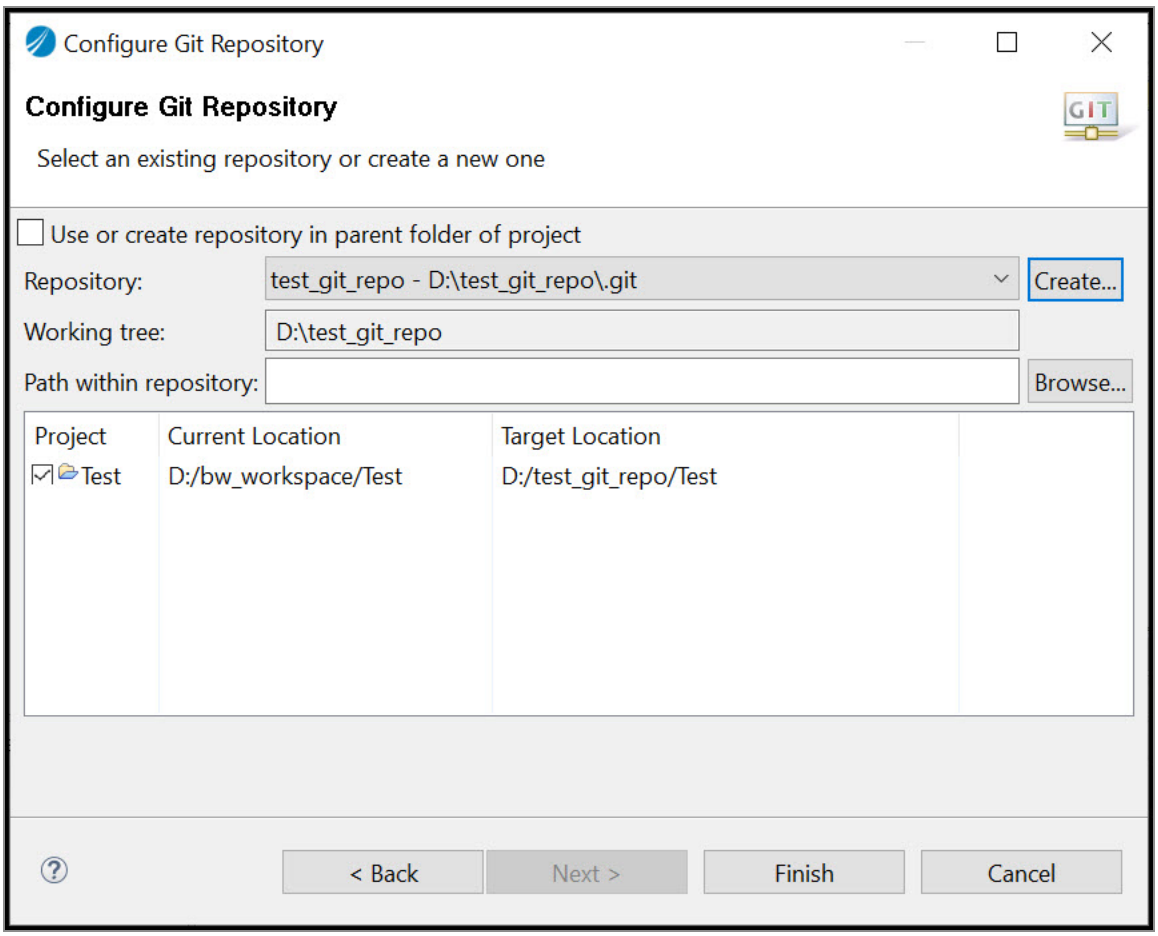
Before you begin

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, select **Git** and click **Next**.
3. In the Configure Git Repository dialog, either create a new Git repository or select an

existing Git repository from a drop-down list.



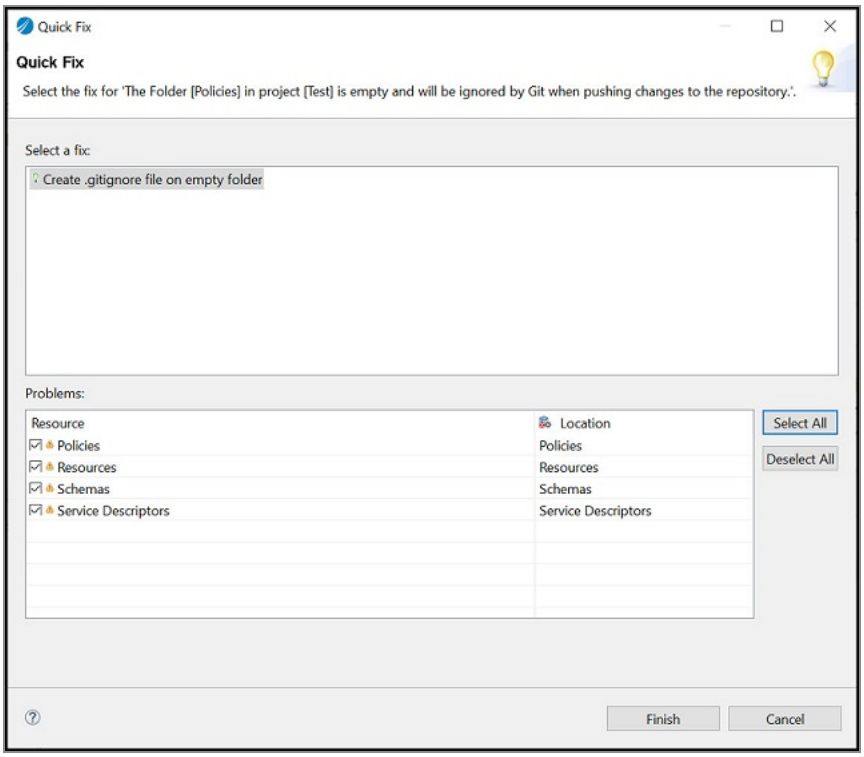
4. Click **Finish** to share the project.

   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**.

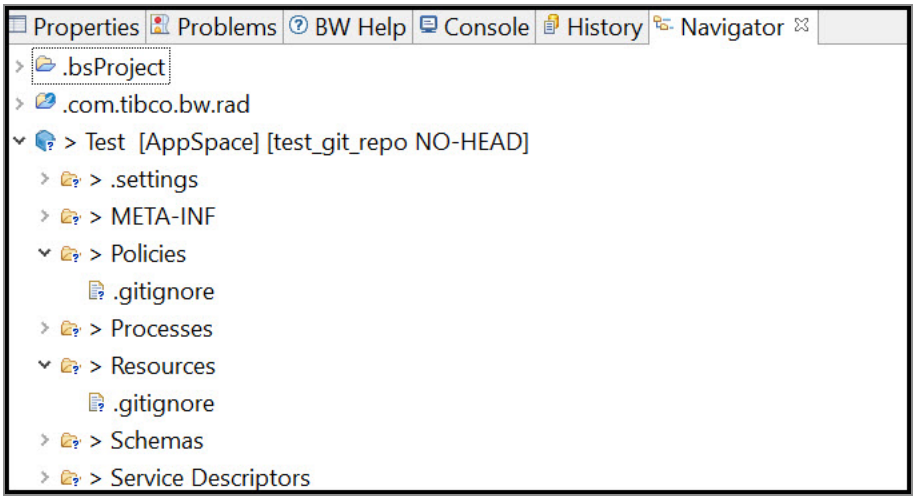
   The Quick Fix dialog opens. In the **Select a fix** section, **Create .gitignore file on empty folder** option is selected. List of Resource folders is displayed along with the Location in the **Problems** section on the dialog.

6. Click **Select All** to select all checkboxes 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...**.

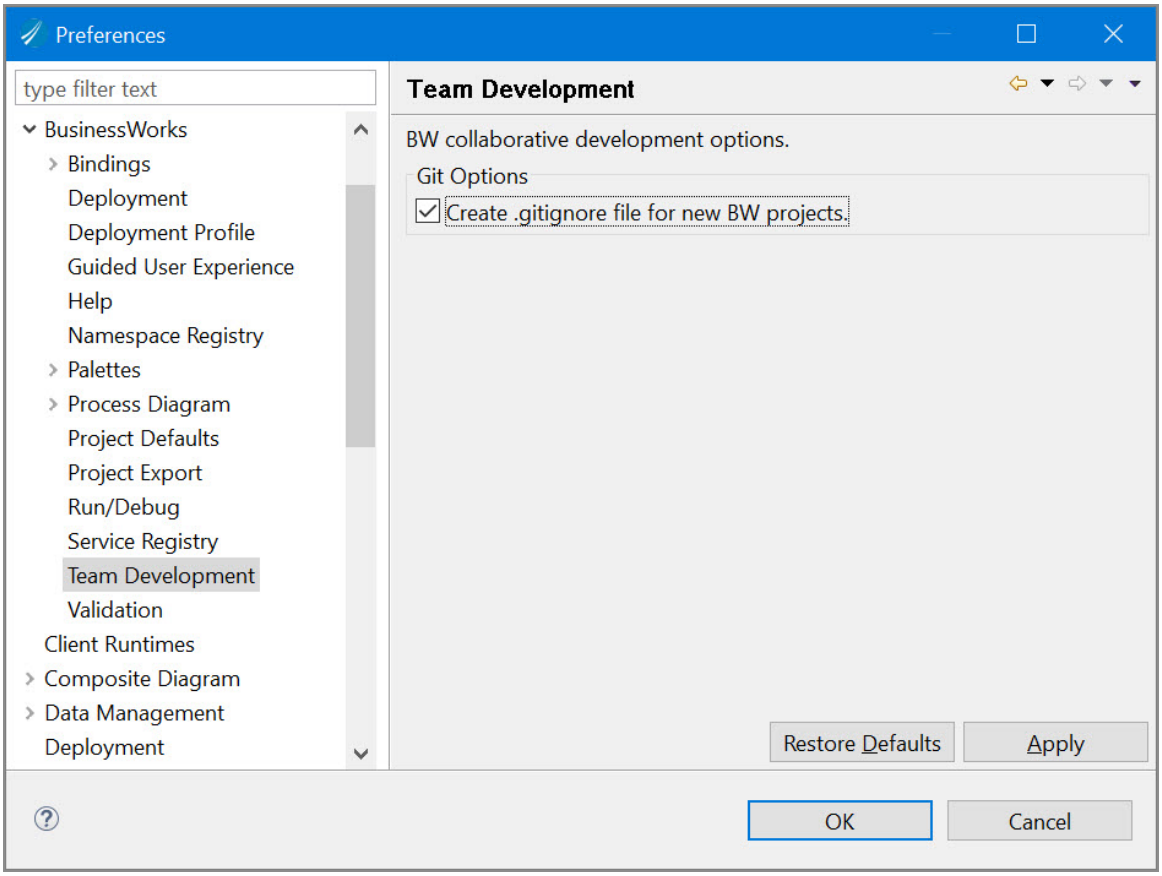
   Show View dialog is displayed.
9. Select **General > Navigator** and click **Ok**.



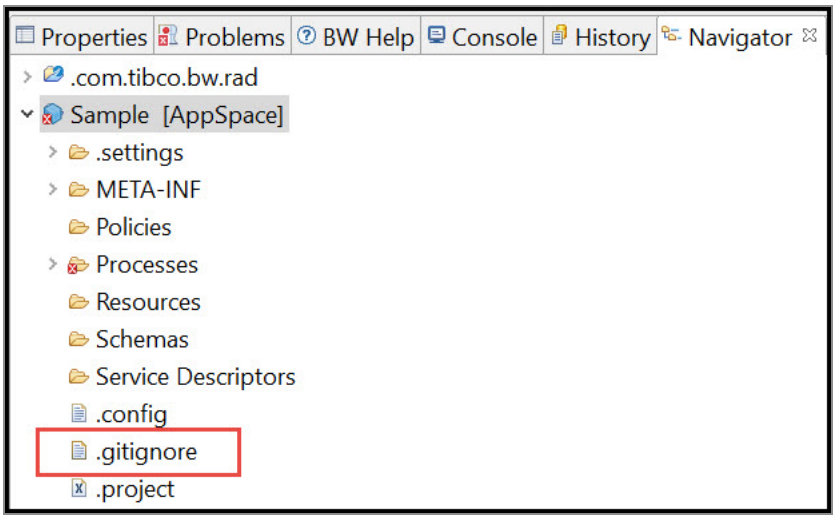
Generating gitignore Files at Application Module Level

Procedure

1. Select **Window > Preferences > BusinessWorks > Team Development** menu.
2. Select the **Create .gitignore file for new BW projects.** checkbox. 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.



Note: 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.

Configuring TIBCO Business Studio for BusinessWorks with SVN plug-in

The Subversive plug-in installation consists of two parts, installing Subversive plug-in and the Subversive SVN Connectors.

To use Subversive, install the Subversive plug-in and at least one Subversive SVN Connector compatible with your OS and the used SVN version.

The Subversive plug-in includes references available for Subversive SVN Connectors, so after installing the Subversive plug-in, you are automatically prompted to install one or multiple connectors.

Before you begin

Ensure that TIBCO BusinessWorks Container Edition is open.

Part 1: Installing Subversive Plug-in:

Procedure

1. From the menu, select **Help > Install New Software...** to open Eclipse Update Manager.
2. In the Install dialog, click **Add**.

   The **Add Repository** dialog opens.
3. Add name and location as the path to an online or archived available Subversive update site from `https://download.eclipse.org/technology/subversive/4.8/release/latest/`. After adding the location, click **OK**.
4. From the list of available components, select the components you want to install and click **Next**.
5. In the Review Licenses dialog, review the licenses, and click **I accept the terms of the license agreement**.
6. Click **Finish** to start the installation of the plug-in.

What to do next

After installing the software, restart TIBCO Business Studio for BusinessWorks. This restart is necessary for the software to install completely.

Part 2: Installing Subversive SVN Connectors

Subversive SVN Connectors are SVN libraries used by Subversive to communicate with SVN repositories.

Once the Subversive plug-in is installed and TIBCO Business Studio for BusinessWorks is rebooted, on the Welcome page select the **Installing SVN connector** option under Subversive and follow the steps to install SVN Connectors compatible with the installed version of the plug-in OR follow the steps below to install using Preferences in TIBCO Business Studio for BusinessWorks.

Procedure

1. Run Eclipse and select **Help > Install New Software...** from the main menu.
2. In the **Install** dialog that appears, click **Add...** and enter the URL of the following update site:
   https://osspit.org/eclipse/subversive-connectors/
3. Select the required features to install and follow the standard plug-in installation procedure.

What to do next

After installing the software, restart TIBCO Business Studio for BusinessWorks. This restart is necessary for the software to install completely.

Using the BWDesign Utility

The `bwdesign` utility provides a command-line interface for creating, validating, importing, or exporting resources stored in a workspace.

Before you begin

1. To use the `bwdesign` utility, open a terminal and navigate to the `BW_HOME\bin`.
2. In the command-line type: bwdesign -data <*TIBCO_BusinessStudio_workspace_absolutePath*>. For example, bwdesign -data C:\myWorkspace.
3. To view the arguments and options for a command, open a terminal, navigate to the `BW_HOME\bin` folder, and type the `bwdesign help` command at the command line.

Command name and Syntax	Description

`cd`
SYNTAX:
`cd path`
	Changes the current working directory to the specified folder.
	Arguments:
	• `path` - The path of the new current working directory

`clear` SYNTAX: `clear`	Clears the command-line console.

`diagram:gen_diagrams`
SYNTAX:
`diagram:gen_diagrams [project]`
	Save each process diagram of a project in a `.bwd` format.
	Arguments:
	• `outputfolder` - Optional, it is

Command name and Syntax	Description
	used to save the diagrams in a given path.
`edition` SYNTAX: `edition`	Prints out the edition of this TIBCO Business Studio for BusinessWorks.

`execute`
SYNTAX:
`execute file`
	Runs a batch script file containing a set of commands to run in sequence.
	Arguments:
	• `file` - Script file, which contains a set of commands run in sequence.

`exit` SYNTAX: `exit`	Exits the command-line console.

`generate_manifest_json`
SYNTAX:
`generate_manifest_json [options] [ear_location] [manifest_location]`
	Creates a `manifest.json` or `ear` from the ActiveMatrix BusinessWorks EAR file.
	Arguments:
	• `manifest_location or ear_location` - The destination folder that contains the created `manifest.json` or `ear` file.
	• `ear_location` - The location of the ActiveMatrix BusinessWorks EAR file.
	Options:
	• `-project` - Name of the project

Command name and Syntax	Description
	for which the `manifest.json` file is created. • `-ear` - Use this flag to generate a new EAR file from an EAR file created on earlier versions of TIBCO Business Studio for BusinessWorks.

`ls`
SYNTAX:
`ls [-f|-p] [-a]`
	List the projects in the current workspace or the files in the current working directory.
	Arguments:
	• `a` - List all the entities including hidden ones.
	• `f` - List the files in the file system.
	• `p` - List the projects in the current workspace.

`pwd` SYNTAX: `pwd`	Prints the location of the current working directory.
`quit` SYNTAX: `quit`	Exits the command-line console.
`setedition` SYNTAX:	Converts projects from their existing editions to this edition of TIBCO Business Studio for BusinessWorks. If the option `-name` is not selected, this

Command name and Syntax	Description

`setedition -name -t`
 	
EXAMPLE:
 	
`setedition -name test.application -t bwcf`
	command sets the edition of all the projects in the workspace to the current edition of TIBCO Business Studio for BusinessWorks.
	
	Select the option `-name`, and provide the names of the projects to be converted.
	
	Provide comma-separated values to convert multiple projects. The `-t` tag changes the edition to the specified edition. The values to be used for the editions are `bwcf`, `bwe` and `bwcloud`.
	
	Optional. If the `-f` option is used, the following message is not prompted:
	
	`Are you sure you want to change the edition of the given project?`

`system:create`
 	
SYNTAX:
 	
`system:create [options]`
 	
Alternatively, you can use the `create` command.
 	
EXAMPLE:
 	
`create application test2.application test`
 	
Note: This example generates `test2.application` for the test application module.
	Creates resources in the workspace.
	
	Options:
	
	• `--help` - Display this help message.
	• `-verbose` - This parameter prints information regarding the workspace that is used for the `create` command.
	
	The success or failure message is printed at all times, even if the verbose flag is not used.
	• `application [name] [modules] -v [version]`
	
	Create an application project with the given name, including the given modules.

Command name and Syntax	Description
	Optionally, specify the application version using the -v argument. Version format: `major.minor.micro.qualifier` For example: `'1.0.0.qualifier'` • `-f` - This parameter checks if an application with the same name exists, if it does, it deletes the application and creates a new application. If the application with the same name does not exist, it creates a new application.

`system:export`
 	
SYNTAX:
 	
`system:export [options] [projects] [output folder]`
 	
Alternatively, you can use the `export` command.
 	
EXAMPLE:
 	
`export -ear test2.application -removeunused -path D:\Samples`
	Exports BW artifacts from the specified projects in the workspace to a folder. The artifacts can be ZIP or EAR files.
	
	Arguments:
	
	• `projects` - The name of the projects to export, separated by commas, for example, project [,project]*., Must specify at least one project. BW Applications can be exported as EAR files.
	
	• `-path` -This flag is used to give an output or destination folder, where the content is exported. Defaults to local folder.
	
	Options:
	
	• `-alsomoduleproperty` - This parameter adds the module property in a property file.
	
	• `-bin`, `-binary` - Export shared model as binary shared module. Can be used with `-zip` option.

Command name and Syntax	Description
	Cannot be used with `-ear` option. • `-cxf <CXF_Project_Name>` - This command must run in the workspace where the CXF project exists. It installs the custom XPath function in the project *TIBCO_HOME*. • `-e`, `-ear` - Export the application as a deployable ear file (default). Can be used with application projects. Cannot be used with module projects. • `-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. • `--help` - Display this help message. • `-includesystem` - This parameter adds all the predefined properties in a property file. For example, *BW.APPNODE.NAME* . • `-name [name]` - Use the supplied name for the exported module. • `-noprofile` - This parameter removes all the profiles from the EAR file. • `-pf {profile name}` - Export the named profile of the given module. Multiple profiles can be exported by a comma-separated format.

Command name and Syntax	Description
	• `-pr {profile name}` - Export the profile in property files. Multiple profiles can be exported by a comma-separated format. • `-removeunused` - Exclude unused resources from the application when creating the EAR. • `-removediagraminfo` - Removes process diagram information when creating the .EAR file. • `-substvar` - Export the substvar file of a given profile to a given destination folder. For example, `export -substvar {profile name} {application or module name} -path {path where substvar file should be exported}` If no path is given, then the default path is taken. • `-t` - This parameter tokenizes the property file if a project is deployed in TIBCO BusinessWorks™ Container Edition. • `-z, -zip` - Export the model as a zip file. Cannot be used with the `-ear` option. • `-dot` - This parameter exports the profile using the dot separator.
`system:import` SYNTAX:	Imports flat or ZIP projects into the current workspace.

Command name and Syntax	Description
`system:import [options] files` Alternatively, you can use the `import` command.	Arguments: • `files` - The names of the folders, which contain the target flat projects to import. All the flat projects found in the specified folders are imported. The folders are separated by commas. By default, zip files are ignored. If the items to import are zip archives, use the `-z`, `-zip`, `-fz`, `-fzip` options. Options: • `-fz`, `-fzip` - The specified items to import are zip archives in the folders specified by the arguments. All the zip projects in these folders are imported, while flat projects are ignored. Multiple folders are separated by commas. • `-z`, `-zip` - The specified items to import are zip archives specified by the arguments. Multiple zip files are separated by commas. Output: `file`, `status` • `file` - Name of the project • `status` - Result of the import, either "imported", "ignored", or "failed {message}"
`system:importpreferences` SYNTAX:	This command imports the preferences set in the preferences file. Arguments:

Command name and Syntax	Description
`system:importpreferences [options] file` Alternatively, you can use the `importpreferences` command.	• `file` - Absolute path of the preferences file to be imported. Output: `file, status` • `file` - Name of the project • `status` - Result of the import, either "imported", "ignored", or "failed {message}"

`system:validate`
 	
SYNTAX:
 	
`system:validate [options] [modules]`
	This command validates BW modules in the current workspace. If you do not provide any module name, by default, it validates all modules.
	
	Arguments:
	
	• `modules` -
	
	The name of the modules to validate, separated by commas, for example, module[,module]*. Defaults to all modules in the workspace.
	
	Options:
	
	• `-h,--help` - Display help for this command.
	• `-d <Directory path>,--directory <Directory path>` - Path of the directory to store the validation result.
	
	At the end of the validation report, this command displays the number of errors and warnings for a project.

`system:clean`	This command cleans specific projects

Command name and Syntax	Description
SYNTAX: `system:clean project1, project2, project3`	(all the projects if none is specified in the command) in the workspace.

`system:delete`
SYNTAX:
`delete -f`
`system:delete project1, project2, project3`
	Deletes projects from the workspace.
	`-f` - This parameter deletes all the projects in the workspace without any message prompts.

`system:generate_manifest_json -project`
SYNTAX:
`generate_manifest_json -project <application name present in workspace> <folder location where JSON is created>`
	This command generates a JSON file without creating an EAR file in the workspace.

`export_to_consul`
SYNTAX:
`export_to_consul [options]`
	Export the properties from the specified profile to the Consul key-value store.
	Options:
	• `-profile` - Name of the profile to export.
	• `-project` - Name of the ActiveMatrix BusinessWorks application project in which the profile exists.

Command name and Syntax	Description
	• `-consul` - The URL of the Consul. For example, http://127.0.0.1:8500. • `-consultoken` - Optional. The Consul token used for authentication. • `-customkey` - Optional. The custom encryption key used for encrypting the application properties of type Password.

Messaging

You can now integrate TIBCO FTL® and TIBCO EMS with TIBCO BusinessWorks Container Edition. The following topics are covered in this section:

- Integrating with TIBCO FTL®.
- Integrating with TIBCO EMS.

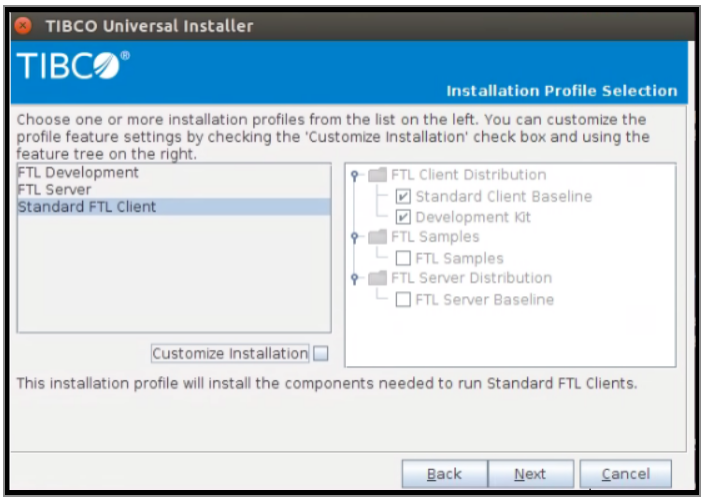
Integrating with TIBCO FTL®

This topic describes how to add the TIBCO FTL 6.x Native Libraries to the TIBCO BusinessWorks Container Edition runtime and design time environment.

Runtime

To add TIBCO FTL 6.x native libraries into TIBCO BusinessWorks Container Edition runtime environment:

1. Download TIBCO FTL 6.x on a 64-bit Linux machine.
2. During the installation process select the **Standard FTL Client** profile as shown.



3. Create the TIBCO FTL client libraries.

   From a temporary folder execute:

   ```
   ls <FTL-TIBCO-HOME>/lib/*.so* | zip -j ftl_linux_client_libraries -@
   ```

4. Cloud Foundry only

   a. Extract the TIBCO BusinessWorks Container Edition buildpack into a temporary folder, *<buildpack-temp-folder>* and copy `<temp folder>/ftl_linux_client_libraries.zip` into the `<buildpack-temp-folder>/resources/addons/lib` folder.

   b. Copy the TIBCO FTL OSGi client libraries from `<FTL-HOME>/components/shared/1.0.0/plugins into <temporary folder>/resources/addons/jars`.

   c. Zip the contents of the temporary folder and push the customized buildpack to your Cloud Foundry environment.

5. Docker only

   a. Build the TIBCO BusinessWorks Container Edition base Docker image. For more information, see Creating the TIBCO BusinessWorks™ Container Edition Base Docker Image.

   b. Copy the TIBCO FTL OSGi client libraries from `<FTL-HOME>/components/shared/1.0.0/plugins` into temporary folder created in

step 3.

c. From the temporary folder created in Step 3 above, use the Docker file given below to copy this zip file and FTL OSGi client libraries into the base Docker image:

```
FROM tibco/bwce:latest
COPY *.zip /resources/addons/lib
COPY com.tibco.ftl* /resources/addons/jars
COPY <FTL-TIBCO-HOME>_<version>_<build_number>_bwce-
runtime.zip /resources/addons/plugins
```

d. From the temporary folder, build the Docker image:

```
docker build -t TAG-NAME .
```

Note: Alpine Linux is not supported.

Designtime

This section describes how to add the TIBCO FTL 6.x Native Libraries to the TIBCO BusinessWorks Container Edition designtime environment.

Using the bwinstall Utility

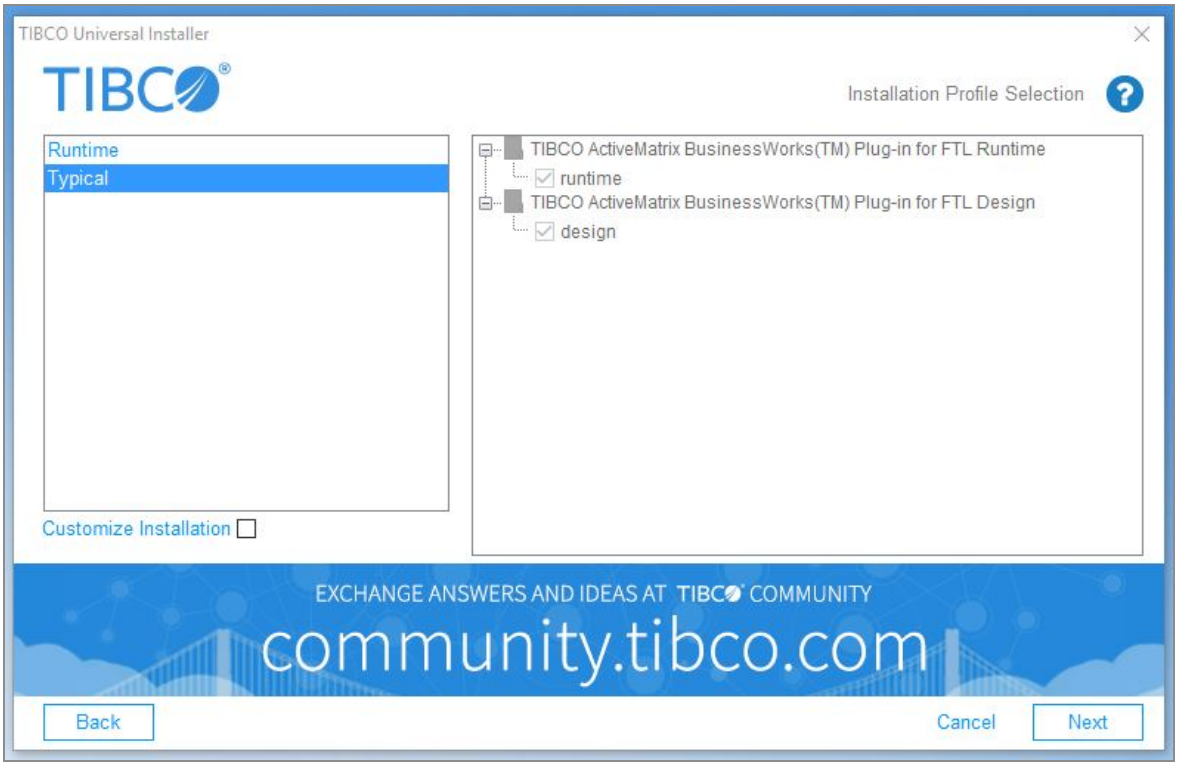
To install the TIBCO FTL client libraries for your TIBCO BusinessWorks Container Edition environment, run the following command from the *<BW_HOME>*`/bin` folder and follow the prompts:

```
bwinstall ftl-driver
```

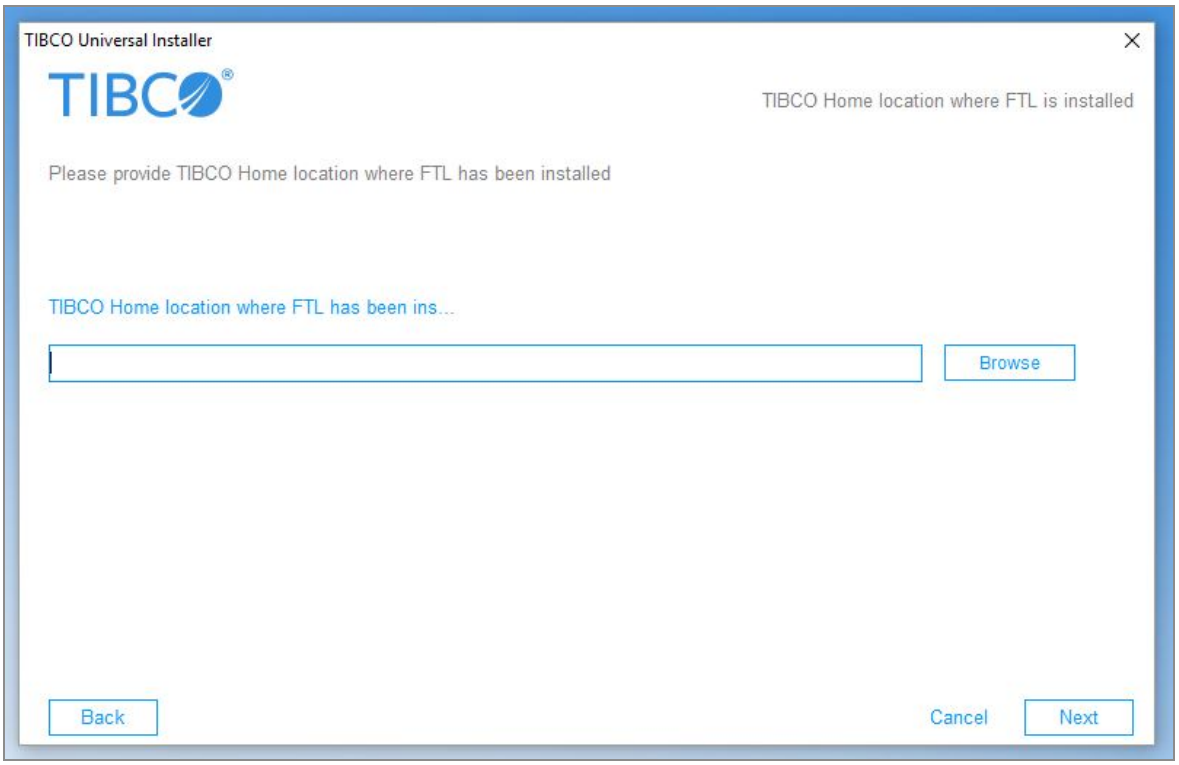
Using the TIBCO Universal Installer

Use the FTL plugin installer to install the TIBCO FTL client libraries for your TIBCO BusinessWorks Container Edition environment.

1. Run the TIBCOUniversalInstaller executable.
2. On the Welcome screen, click **Next**.
3. Read through the license agreement, select **I accept the terms of the license agreement** if you agree to the terms, and click **Next**.
4. Choose to create a new installation environment in an existing *TIBCO_HOME*. Specify the location that TIBCO BusinessWorks Container Edition is installed. Then click **Next**.
5. On the Installation Profile Selection screen, ensure Typical is selected, and click **Next**.



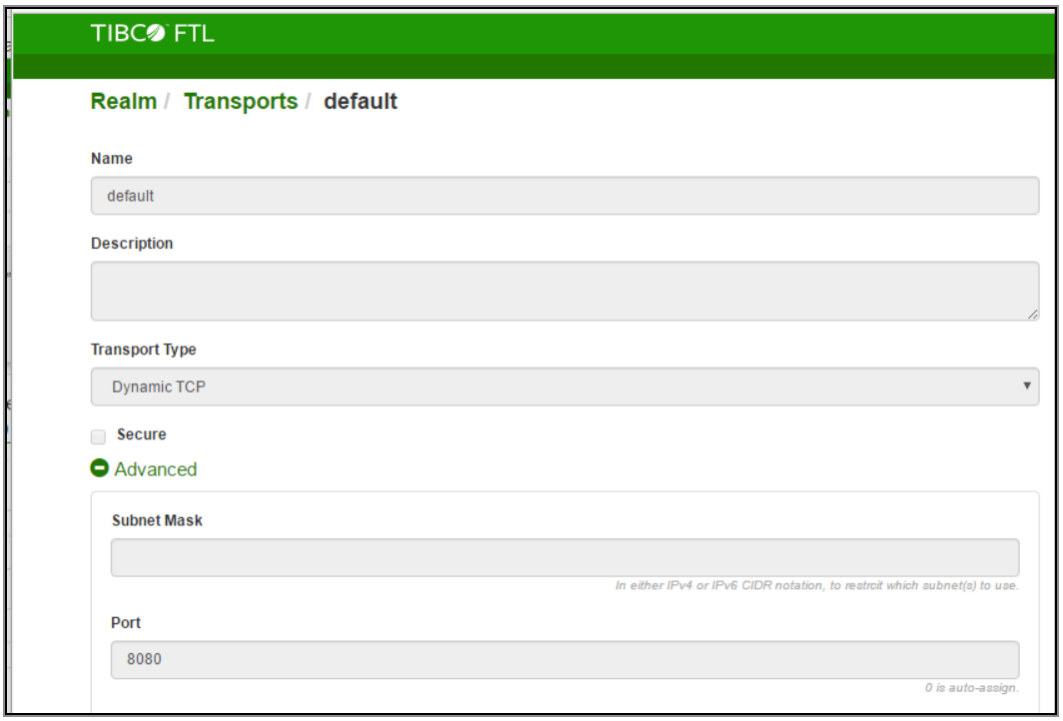
6. Provide the *TIBCO_HOME* that TIBCO FTL has been installed, and click **Next**.



7. On the Pre-Install Summary screen, verify the list of products and components selected for installation, and click **Install**.
8. Verify the Post-Install Summary, click **Finish** to complete the installation process, and close the installer window.

Limitations

- Due to ephemeral nature of containers, only the **Dynamic TCP** transport type should be used on both Docker and Cloud Foundry environments.
- The following limitations are due to the single port restriction of Cloud Foundry:
  - Activities from the FTL palette should not be used along with the HTTP Receiver activity or with a REST service.
  - An application should not contain more than one instance of the FTL Shared Resource.
  - Port `8080` must be configured for **Dynamic TCP**.



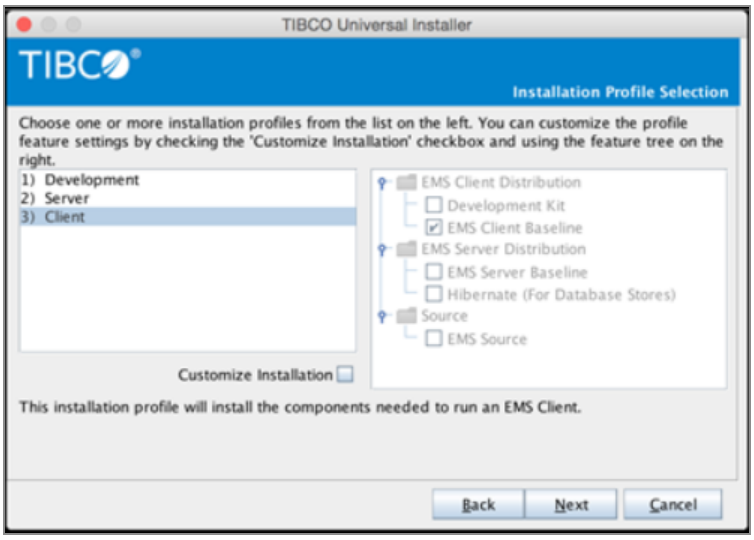
Integrating with TIBCO EMS

Before you begin

Ensure that the TIBCO Enterprise Message Service™ client OSGi bundles are present in the `<EMS-HOME>/components/shared/1.0/plugins` folder.

If you do not have these client libraries, install the TIBCO Enterprise Message Service™ libraries by selecting either the Development or Client installation profiles.

Alternatively, copy the `<EMS-HOME>/components` folder from an existing installation to your local machine.



Design Time

To install the TIBCO Enterprise Message Service™ Client libraries, run the following command from the `<BWCE_HOME>/bin` folder and follow the prompts:

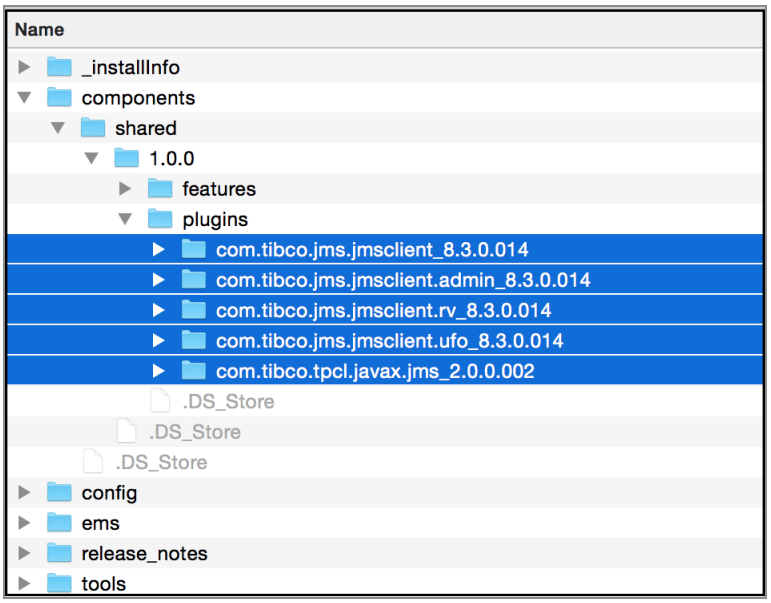
```
bwinstall ems-driver
```

Runtime

To add the TIBCO Enterprise Message Service™ client libraries to the TIBCO BusinessWorks Container Edition runtime environment:

Cloud Foundry

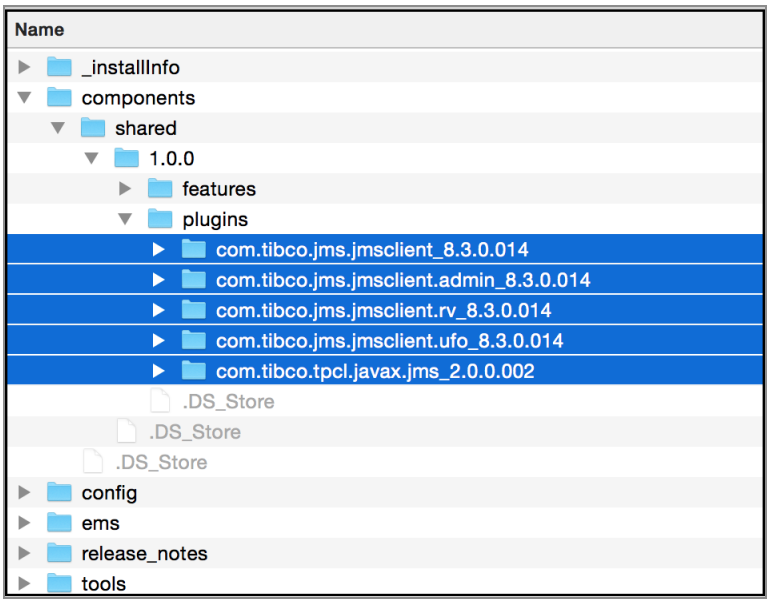
1. Extract the TIBCO BusinessWorks Container Edition buildpack to a temporary folder
2. Copy the TIBCO EMS OSGi client libraries from `<EMS-HOME>/components/1.0/plugins` into `<temporary folder>/resources/addons/jars`.



3. Zip the contents of the temporary folder and push the customized buildpack to your Cloud Foundry environment.

Docker

1. Build the TIBCO BusinessWorks Container Edition base Docker image. For more information, refer to Creating the TIBCO BusinessWorks™ Container Edition Base Docker Image.
2. Copy the TIBCO Enterprise Message Service™ OSGi client libraries from `<EMS-HOME>/components/1.0/plugins` into a temporary folder.



3. From the temporary folder use the Docker file given below to copy these jars into the base Docker image:

```
FROM tibco/bwce:latest
COPY . /resources/addons/jars
```

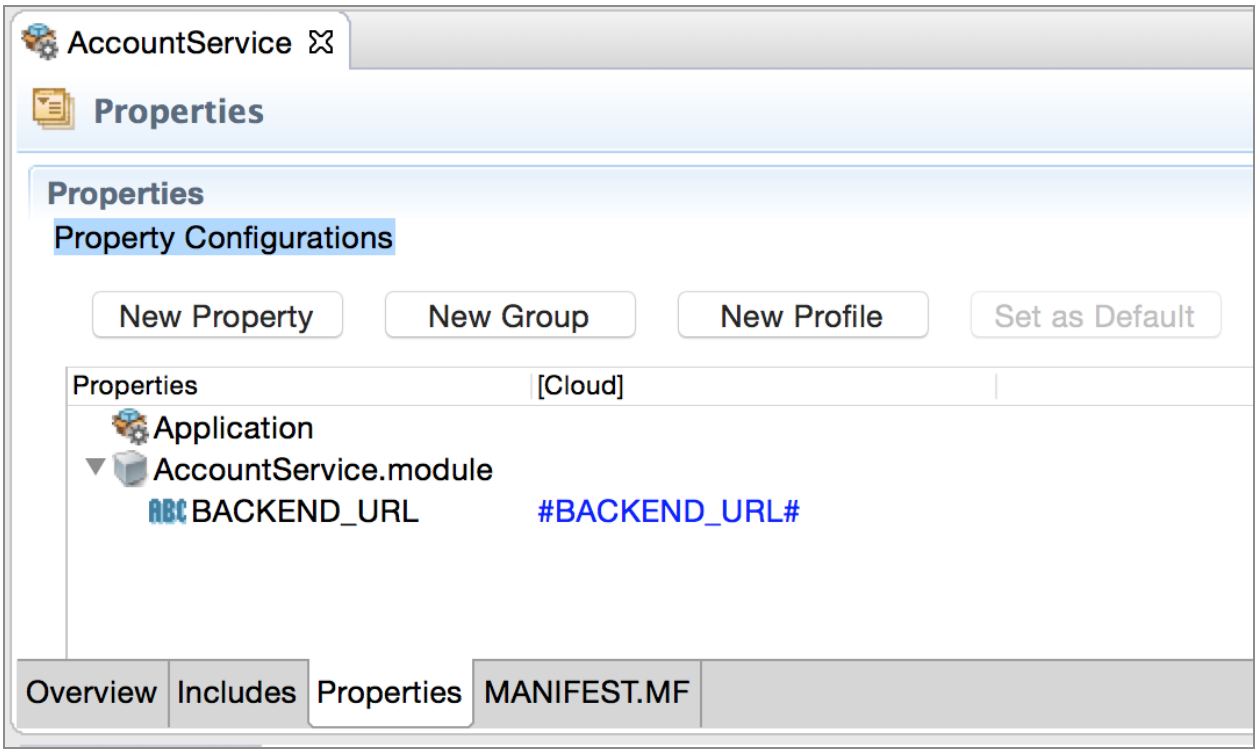
4. From the temporary folder, build the Docker image:

```
docker build -t TAG-NAME .
```

Using Configurations from Configuration Management Services

Use configurations from the configuration management services such as Consul and Spring-Cloud-Config, by defining a token such as `#<property name>#` in the application properties, where *<property name>* is the name of the configuration parameter.

For example, `#BACKEND_URL#`.

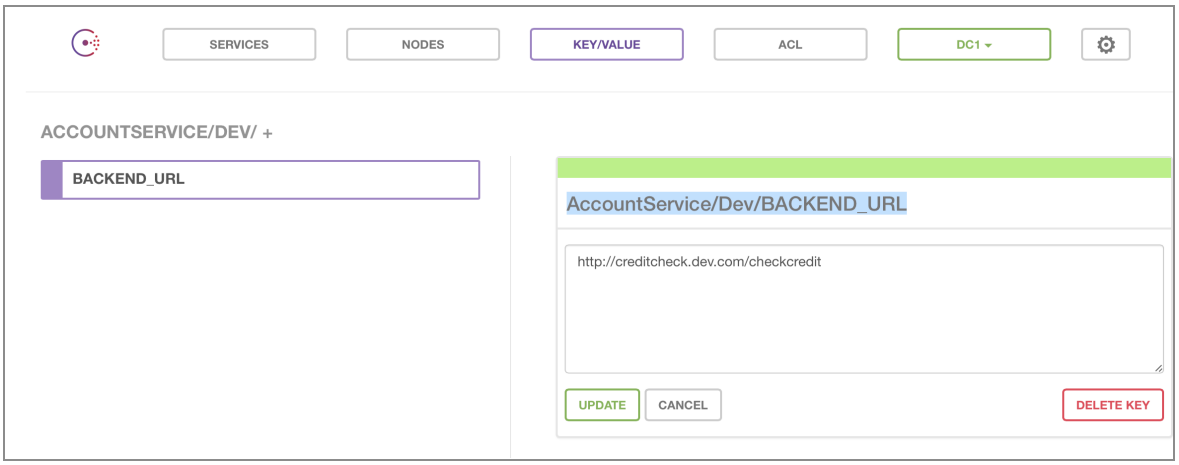


Using Consul for Configuration Management Services

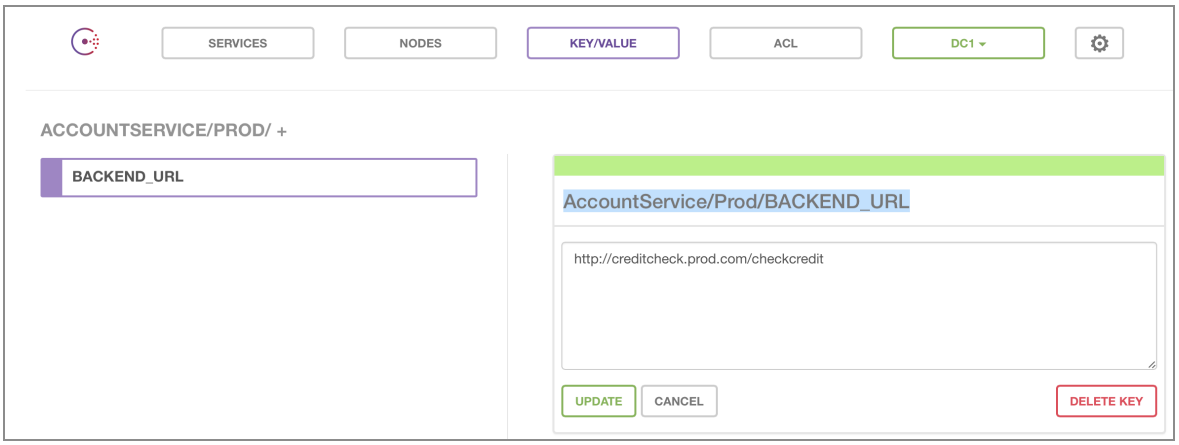
Follow these steps to use configurations from the Consul:

1. Set the environment variable `CONSUL_SERVER_URL`. For more information, see Environment Variables.
2. In your Consul service, define the keys using the format `<BWCE_APP_NAME>/<PROFILE NAME>/<KEY Name>`.

   For example, `AccountService/Dev/BACKEND_URL`



`AccountService/Prod/BACKEND_URL`



Note: The X_CONSUL_TOKEN environment variable should be used to when authenticatication is enabled on the Consul server.

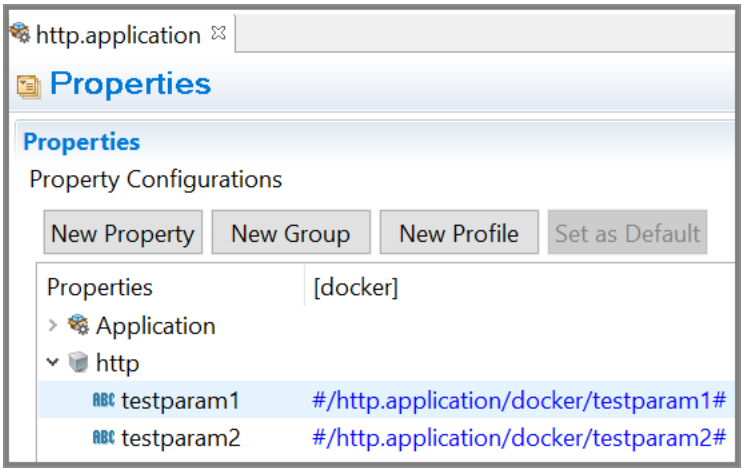
Using AWS for Configuration Management Services

You can use the AWS Systems Manager Parameter Store for configuration and secret management services. This is defined by the token `#/<BWCE_APP_NAME>/<PROFILE_NAME>/<property name>#` in the application properties using Container Configuration. Tokenize the application properties, where the *<property name>* is the name of the configuration parameter.

For example, `#/http.application/docker/testparam1#`.

Note:

- AWS Systems Manager Parameter Store is only supported on the Docker platform.
- The format `#/<BWCE_APP_NAME>/<PROFILE_NAME>/<property name>#` is not a mandatory format that needs to be used. This format is used where the name of the properties is the same in two different applications whereas their values might be different. Although this is a recommended format to use.



Prerequisites

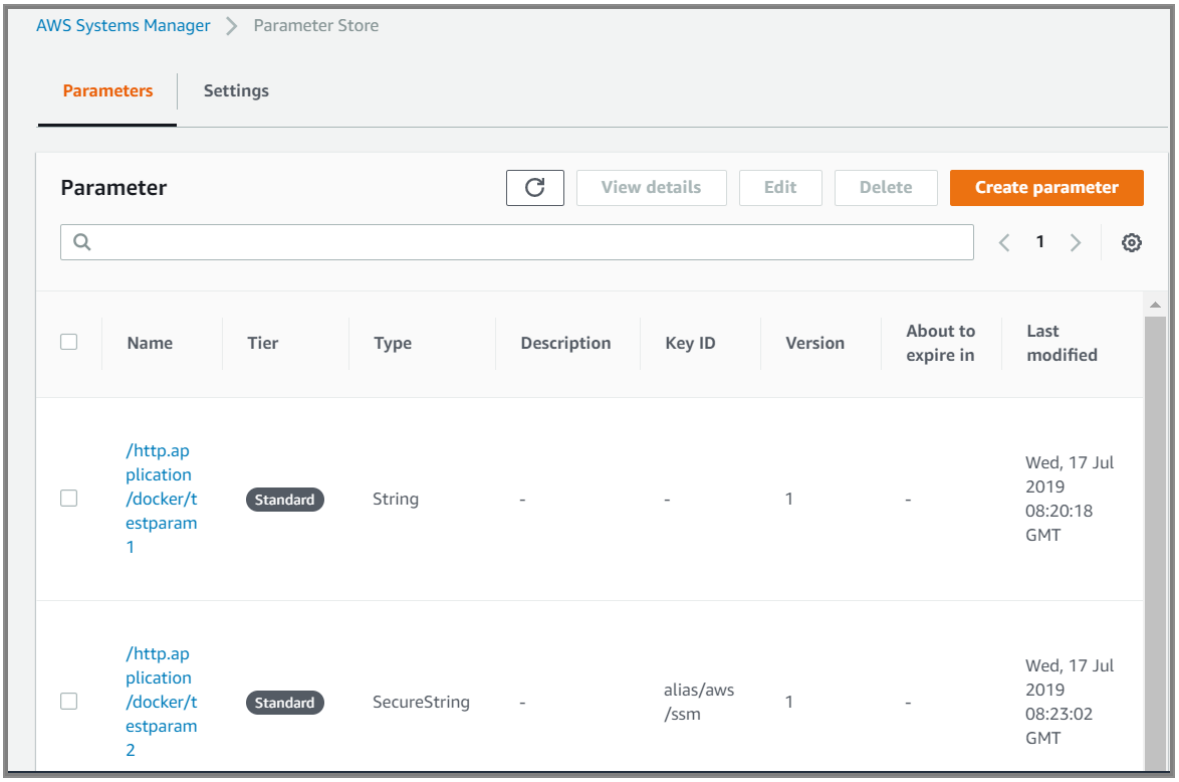
Ensure that the TIBCO ActiveMatrix BusinessWorks™ Plug-in Component for AWS Common Services Software plug-in is added in the `resources\addons\plugins` path in your Docker installation.

Procedure

Follow these steps to use configurations from the AWS Systems Manager Parameter Store:

1. In TIBCO Business Studio for BusinessWorks, navigate to the **Menu** bar and select **Run > Run Configurations > Environment** and set the environment variables `AWS_ACCESS_KEY`, `AWS_SECRET_KEY`,`AWS_REGION`, and `APP_CONFIG_PROFILE`. For more information, see Environment Variables.
2. Select the desired profile as the default profile for TIBCO Business Studio for BusinessWorks to fetch the values from the AWS Systems Manager Parameter Store.
3. In your AWS parameter store, define the keys using the format `/<BWCE_APP_NAME>/<PROFILE NAME>/<property name>`.

   For example, `/http.application/docker/testparam1`



Using Kubernetes ConfigMap for Configuration Management Services

Follow the steps to use configuration management with ConfigMaps

1. Export the application properties file for the profile containing the module properties. For more information, see Tokenizing Application Properties for exporting in the Properties file.
2. Add a profile name to the exported properties file. For example, BW_PROFILE=Prod.
3. Create a Kubernetes ConfigMap using the following command:

```
kubectl create configmap <Name-of-config-map> --from-env-
file=<name-of-exported-file>.properties
```

4. Add the created ConfigMap to the Kubernetes pod specification as follows:

```
apiVersion: v1
kind: Pod
metadata:
  name: bookstore-demo
  labels:
    app: bookstore-demo
spec:
  containers:
  - name: bookstore-demo
    image: bookstore-demo:2.4.4
    imagePullPolicy: Never
    envFrom:
    - configMapRef:
      name: <Name-of-config-mapbt>
```

5. When the created ConfigMap is added to the Kubernetes pod, the application picks the values of the tokenized module properties in the application from ConfigMap.

You can refer values from one ConfigMap to another by using the following format: `configmap_1_key: ${configmap_2_key}`.

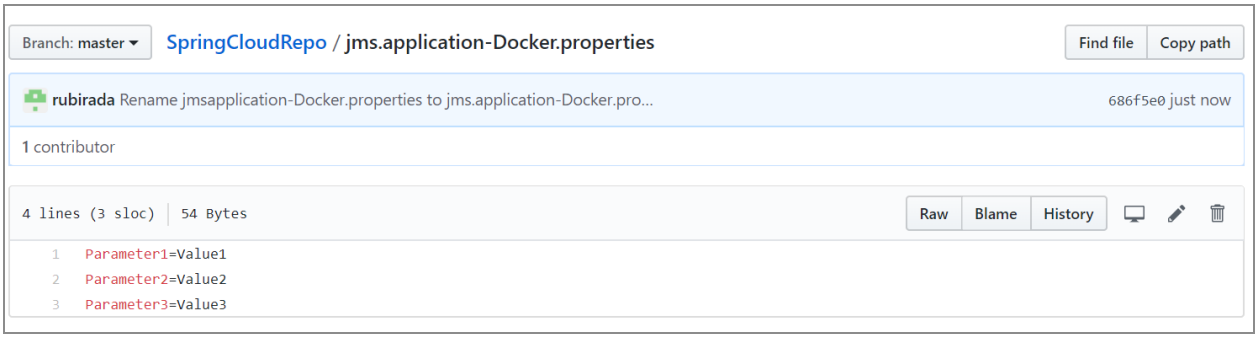
Using Spring Cloud Config for Configuration Management Services

Follow these steps to use configurations from Spring Cloud Config:

1. Set the environment variable `SPRING_CLOUD_CONFIG_SERVER_URL` For more information, see Environment Variables.
2. Create `.properties` file and add the application properties and their values as key-value pairs in the file. The filename for application-specific properties should be `<APPLICATION_NAME>-<PROFILE_NAME>.properties`.

   For example, `jms.application-Docker.properties`

   For common properties, the filename should be `application.properties`.
3. Add the `.properties` files to the Spring Cloud Config Server backend.



Note: Using the ExternalApplicationProperties interface provided in the `com.tibco.bw.runtime.customProps` package, users can write their own integration with custom config management servers.

Using Custom Config Management Server for Configuration Management Services

Follow the steps below to use custom config management server for configuration management services:

1. Create a new package and inside the new package create a new java class.
2. Import `com.tibco.bw.runtime.customProps` package in this java class and implement the custom config management by implementing the interface `ExternalApplicationProperties` provided in this package.

   The implementation of custom configuration management should register an OSGi service implementing the `ExternalApplicationProperties` interface.

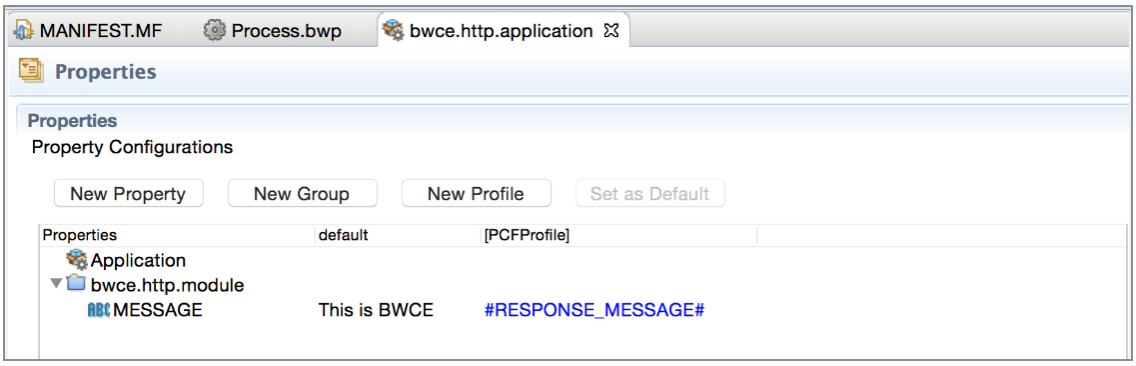
   The interface has a method `getExternalApplicationProperties()` of return type `Map<String, String>` that should return the properties and their corresponding values fetched from the custom configuration management server.
3. Export the project as a plug-in by right-clicking on the newly created package and select **Export > Plug-in Development > Deployable plug-ins and fragments**.
4. In the **Export** wizard, ensure that the package is selected, and specify a location to export the plug-in. Get the JAR file from the specified location in the plug-ins folder and place the JAR in `TIBCO_HOME/bwce/2.5/docker/resources/addons/jars` folder of the TIBCO BusinessWorks Container Edition installation.
5. While deploying the application, make sure to set the value of the property `bw.application.config.management.type` same as the value of the property **external.config** in your OSGi service.
   a. While deploying the application in the studio, set the property as a VM argument.
   b. For docker, set the property using the following syntax:

      ```
      -e BW_JAVA_OPTS="-Dbw.application.config.management.type=custom"
      ```

Using Configurations from Configuration Management Services for Cloud Foundry

You can use configurations from the configuration management services such as ZUUL or Spring Cloud Config by defining a token such as `#<property name>#` in the application properties, where *<property name>* is the name of the configuration parameter.

For example, `#RESPONSE_MESSAGE#`.



The environment variable `APP_CONFIG_PROFILE` can be used to identify the profile name for your deployment environment, for example `QA` or `Prod`. For more information, see Environment Variables for Cloud Foundry.

Integrating with Spring Cloud Config Server

Support for application configuration is provided through integration with the Spring Cloud Config Server.

You first create an instance of the service using the VMware Tanzu Apps Manager and then bind this service instance to your application.

Note: TIBCO BusinessWorks Container Edition communicates with Spring Cloud® Services using HTTPS. If your Cloud Foundry installation uses a self-signed SSL certificate, this certificate should be added to the buildpack before your application can be deployed. To add self-signed SSL certificate to the buildpack, see the *Certificate Management* section of Customize the Buildpack.

Using Custom User Provided Services (CUPS) for Config Servers

The following are required when integrate your application with custom user provided services (CUPS) Spring Cloud Config or ZUUL Configuration service in Cloud Foundry:

1. Name of the Cloud Foundry service instance must contain either `spring-cloud-config` or `zuul-config`.
2. The service configuration must contain the `uri` parameter.

   For example, for ZUUL

```
{
   "credentials": {
       "uri": "http://{ZUUL-SERVER-IP}:{PORT}/zuul"
     },
  "label":"user-provided",
  "name": "zuul-config-server",
  "syslog_drain_url": "",
  "tags": []
}
```

   For Spring Cloud

```
{
   "credentials": {
       "uri": "http://{SPRING-CLOUD-SERVER-IP}:{PORT}"
     },
  "label":"user-provided",
  "name": "spring-cloud-config-server",
  "syslog_drain_url": "",
  "tags": []
}
```

The environment variable `APP_CONFIG_PROFILE` can be used to set `ProfileName`. If this variable is not set, `default` name is used for the profile.

The environment variable `SPRING_CLOUD_CONFIG_VAULT_TOKEN` can be used to pass the Vault token for VMware Tanzu application, while using Vault as back end for Spring Cloud Config.

Note: Using the methods from the new `ProfileResolverHelper` class, users can write their own integration for config management servers.

- getKeysForConfig()- This method returns the list of all the tokens that are required to fetch from the config management server.
- replaceProfileValues()- This method replaces values of the corresponding keys, fetched from the users, in the `pcf.substvar` file.

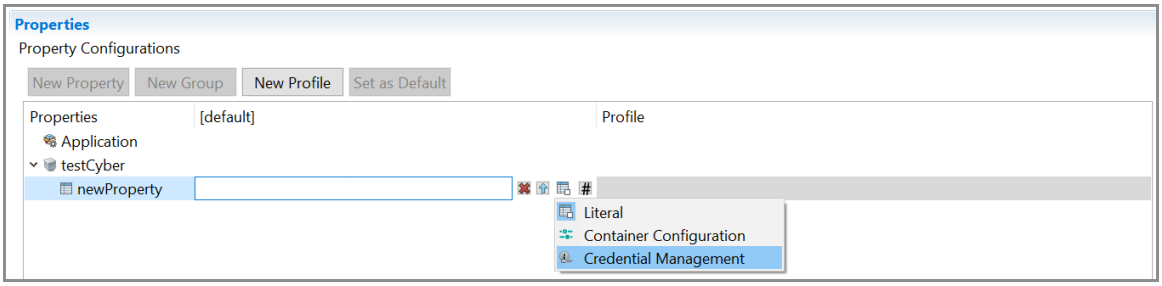
Using Credential Management Service for Properties of Type Password

It is possible to store the passwords externally instead of storing them within the application using external credential management systems. This makes it easier to manage these credentials in a better and more secure manner.

Follow these steps to modify any password properties to use with credential management systems.

Procedure

1. In the TIBCO BusinessWorks Container Edition application module, create a module property of type **Password**.
2. Open the application property editor by navigating to the **Application > Properties** and click the module property created in the above step and click  .
3. Choose the **Credential Management** option.



   The Set Credential Management icon  is displayed next to the password type field.
4. Click the **Set Credential Management** icon. The **Credential Management** window is displayed.
5. On the **Credential Management** window, select the required credential management service provider.

Using CyberArk Application Access Manager for Credential Management Service

Use the credential management service, CyberArk Application Access Manager, to achieve the security in storing the passwords, automatically replace the passwords, and so on. The CyberArk credential management service is more secure than the passwords in clear text format stored in the configuration files.

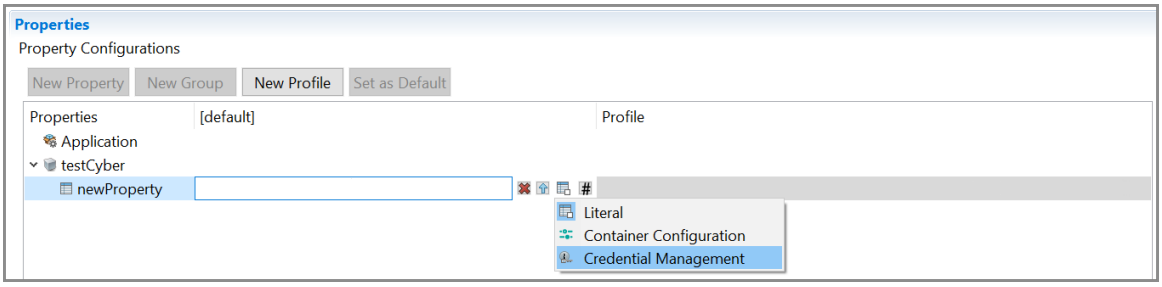
Before you begin

- Set up the CyberArk account for the TIBCO BusinessWorks Container Edition application to create a password. To know more, visit the https://www.cyberark.com/ website.
- Configure the following environment variables:
  - `APP_CONFIG_PROFILE`: Name of the application profile to be used
  - `CYBERARK`: Set to true to enable CyberArk credential management service

Follow these steps to modify the properties of field type password.

Procedure

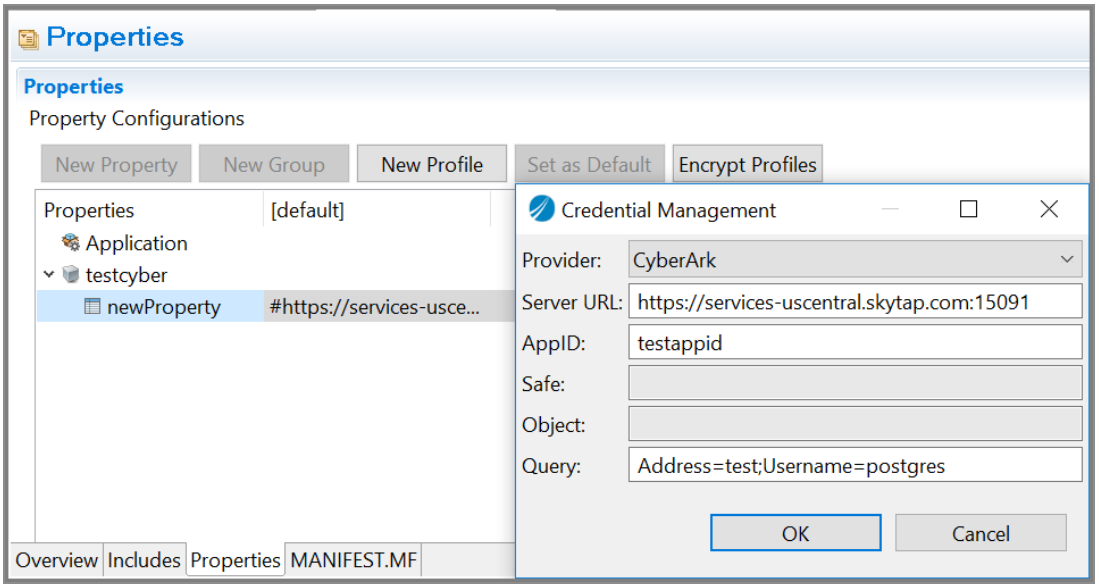
1. To fetch the password from the credential management service, in the application property editor, select the property of type **Password** and click  .
2. Choose the **Credential Management** option.



   The Set Credential Management icon  is displayed next to the password type field.
3. Click the **Set Credential Management** icon. The **Credential Management** window is displayed.
4. On the Credential Management window, select the credential management service

provider, **CyberArk**. Enter information in the following fields to form a Cyberark Query URL:

- **Server URL**: The base URL of the CyberArk environment in use. This parameter is mandatory.
- **AppID**: The unique ID of an application issuing password request. This parameter is mandatory.
- **Safe**: The name of the safe where the password is stored. The **`Safe`** field parameter is mandatory, if the parameter in the **`Query`** field is not defined.
- **Object**: The name of the object password to retrieve. The **`Object`** field parameter is mandatory, if the parameter in the **`Query`** field is not defined.
- **Query**: Define a free query using account properties, including **Safe** and **Object** field parameters. If a query is defined, then the **Safe** and **Object** fields are disabled.



5. Click **OK**.

Note:

- The credential management service, CyberArk, is supported for Docker only.
- The query URL configured for a property can be updated during deployment by passing it as an environment variable. The name of the environment variable should be same as the name of the property that needs to be updated. To update the CyberArk URL, the format for the environment variable is as follows:

  `<ApplicationModuleName>_<ApplicationProfileName>_<PropertyName>= NEW URL`. If the module property or property name has a slash (/) or dot (.) as a separator in between it should be replaced with the underscore (_) separator. For example, an application `cyberark.test.application` is having a property in under `newGroup/newProperty` Groups and the profile is set to default, the environment variable should be: `cyberark_test_default_newGroup_newProperty=<Updated URL>`

Client Authentication and SSL verification

- **Client certificate**: Convert the client certificate to a JKS format and add the certificate in `/Resources/addons/certs` folder for docker. Pass the following environment variables while executing the application
  - CYBERARK_KEYSTORE_PATH
  - CYBERARK_KEYSTORE_PASSWORD

  For more information on the environment variables, see Environment Variables for Docker.

  These environment variables can also be passed as module properties, to provide a secure way to pass the keystore password.

  To pass them as module properties, create the module properties with the following names
  - CYBERARK_KEYSTORE_PATH with type as string

    - CYBERARK_KEYSTORE_PASSWORD with type as password

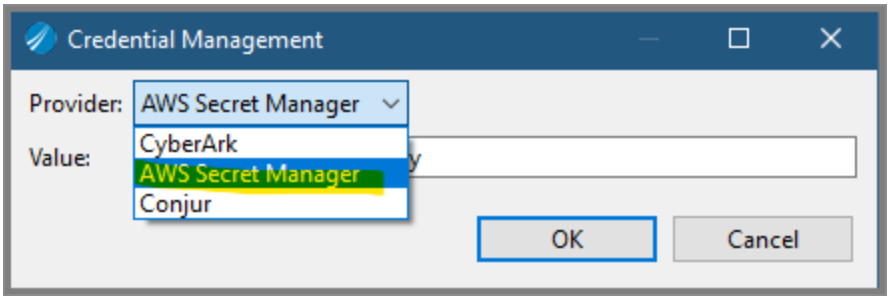
- **Root CA certificate**: Add the Root CA certificate in `/Resources/addons/certs` folder for docker. While running the application on TIBCO Business Studio for BusinessWorks, add this certificate to the CA keystore located at `$BW_HOME/tibcojre64/1.8.0/lib/security/cacerts`.

Using AWS Secret Manager for Credential Management Service

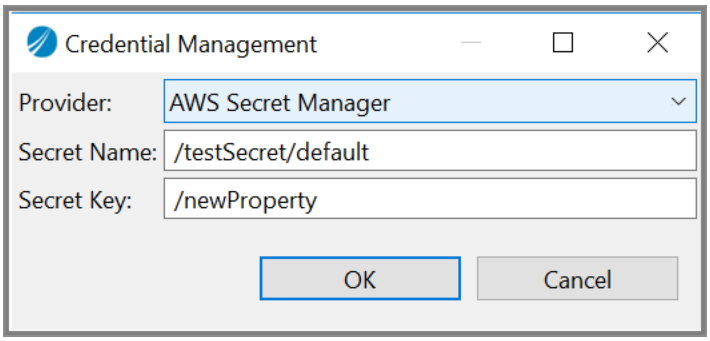
AWS provides an AWS Secret Manager Service for easier management of secrets. Secrets can be database credentials, passwords, third-party API keys, and arbitrary text. Secrets Manager is used to replace hard-coded credentials in the code, including passwords, with API calls to Secrets Manager to retrieve the secret programmatically.

For more information on how to modify password properties to use with credential management systems, see Using Credential Management Service for Properties of Type Password.

The **AWS Secret Manager** supports both Password and String type module properties.



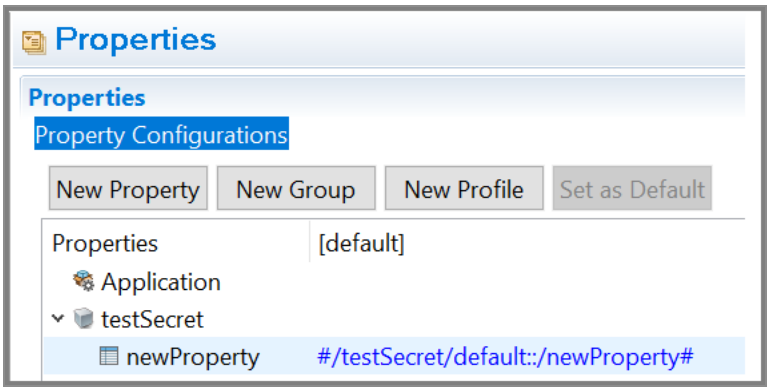
The **AWS Secret Manager** has two fields, **Secret Name** and **Secret Key**. In the **Secret Name** and **Secret Key** fields, add the secret name and secret key that is to be retrieved from the AWS Secret Manager.



On the AWS systems manager console or using AWS CLI, the password properties need to be stored in AWS secret manager. The recommended format is to have secret name as*/<applicationModule_name /<profile_name>*. This makes sure that the property names are unique within the AWS secret manager as well as within the application. However other names can be used. The recommended format is using secret key as the */<property_name>*.

The secret name can either be the application module name or the shared module name along with the profile name and secret key. The secret key is the key whose value is to be retrieved from AWS secret manager. On TIBCO Business Studio for BusinessWorks, the format is stored as `<secret_name>::<secret_key>`.

For example, `#/testSecret/default::/newProperty#`.



Procedure

Follow these steps to use AWS Secret Manager for Secrets Management Service:

1. Set the environment variables *AWS_ACCESS_KEY*, *AWS_SECRET_KEY*, *AWS_REGION*, *AWS_SECRET_MANAGER*, and *APP_CONFIG_PROFILE*. The *AWS_ACCESS_KEY* and *AWS_SECRET_KEY* environment variables are used as credentials for authentication. In order to enable assume role, the following additional environment variables should be passed, *AWS_ROLE_ARN*, *AWS_ROLE_SESSION_ARN*, *AWS_EXTERNAL_ID* (optional),

and *AWS_EXPIRATION_ID* (optional). For more information on the environment variables, see Environment Variables for Docker.

2. In the application properties section, select the **Credential Management** option, and select **AWS Secret Manager** and the value gets populated with the following format `<secret_name>::<secret_key>`.

Integrating ECS and EKS Services with AWS Secret Manager

Support for credential management service for AWS Secret Manager is provided while deploying application on ECS and EKS services.

Set the environment variables *APP_CONFIG_PROFILE* and *AWS_SECRET_MANAGER* to configure ECS and EKS services with AWS Secret Manager.

To use AWS Secret Manager with EKS service for passwordless solution, deploy the application on an EKS cluster configured to a service account, OIDC provider, and an IAM role associated with the cluster, pass the environment variables `APP_CONFIG_PROFILE` and `AWS_SECRET_MANAGER`. The AWS credentials do not need to be passed. For more information, see https://aws.amazon.com/it/blogs/opensource/introducing-fine-grained-iam-roles-service-accounts/

Note: The AWS credentials need to be provided while running the application.

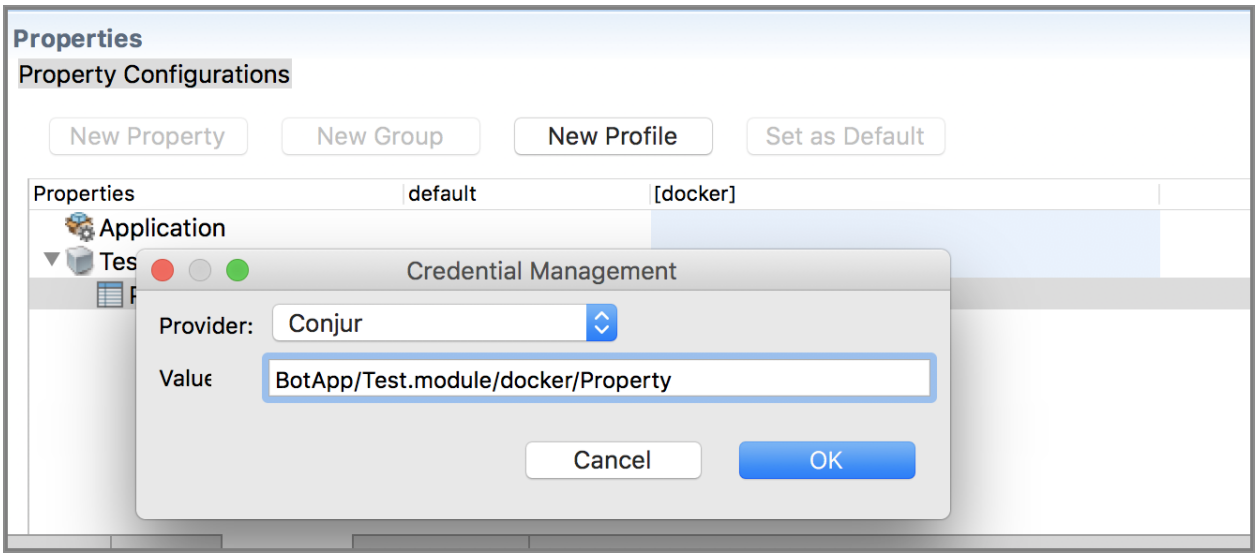
CyberArk Conjur for Credential Management Service

CyberArk Conjur is used to integrate with TIBCO BusinessWorks Container Edition for password management. It enables a TIBCO BusinessWorks Container Edition application to retrieve passwords from Conjur when a password is needed at runtime. This avoids adding any passwords during deploying the configuration or redeploying the application.

Conjur manages the secrets required by applications and other non-human identities to gain access to critical infrastructure, data, and other resources. Conjur secures this access by managing secrets with granular Role-Based Access Control (RBAC) and other security best practices and techniques.

On the Conjur Server, the recommended name for secrets is *<applicationModule_name>/<profile_name>/<property_name>*. This ensures that the property names are unique within the Conjur Server as well as in the application. However, other names can also be used.

In application properties, a new **Conjur** provider is added for Credential Management. To retrieve a secret from the Conjur Server, enter the name of the secret.



Note: Conjur is only supported on the Docker platform.

To retrieve the passwords from the Conjur Server, pass the following environment variables at runtime:

- *CONJUR_ACCOUNT*
- *CONJUR_APPLIANCE_URL*
- *CONJUR_AUTHN_LOGIN*
- *CONJUR_AUTHN_API_KEY*
- *APP_CONFIG_PROFILE*

For more information on the environment variables, see Environment Variables.

SSL Verification

By default, the Conjur appliance generates and uses self-signed SSL certificates (Java-specific certificates known as cacerts). This certificate should be loaded into Java's CA

keystore that holds the list of all the allowed certificates for HTTPS connections. This certificate can be obtained by using the Conjur CLI.

Run the following command from the Conjur CLI to initialize Conjur to retrieve Conjur certificates:

```
conjur init
```

While running the command on Docker, the certificate needs to be added in the `/resources/addons/certs` folder.

While running the command on TIBCO Business Studio for BusinessWorks, the certificate needs to be added to the Java CA keystore at `BW_HOME/tibcojre64/1.8.0/lib/security/cacerts`.

To disable the SSL verification, pass the following environment variable:

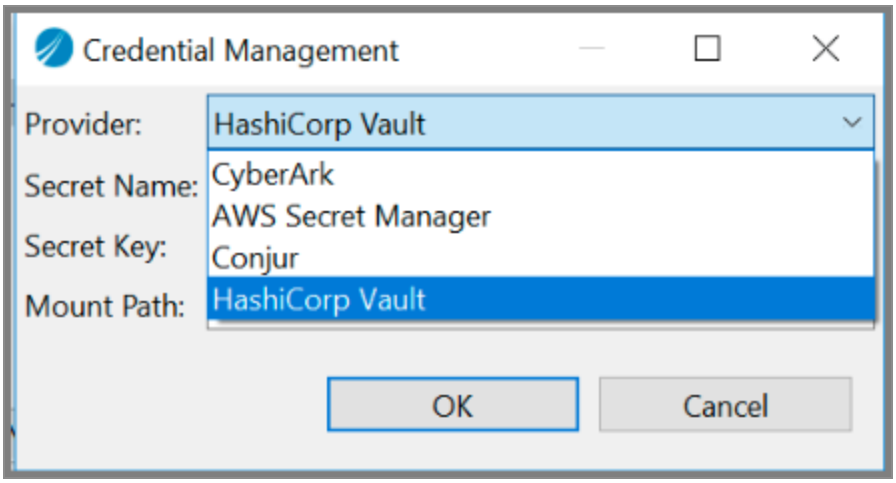
CONJUR_DISABLE_SSL= true

Note: This environment variable should be used only for testing and not in the production environment.

HashiCorp Vault for Credential Management Service

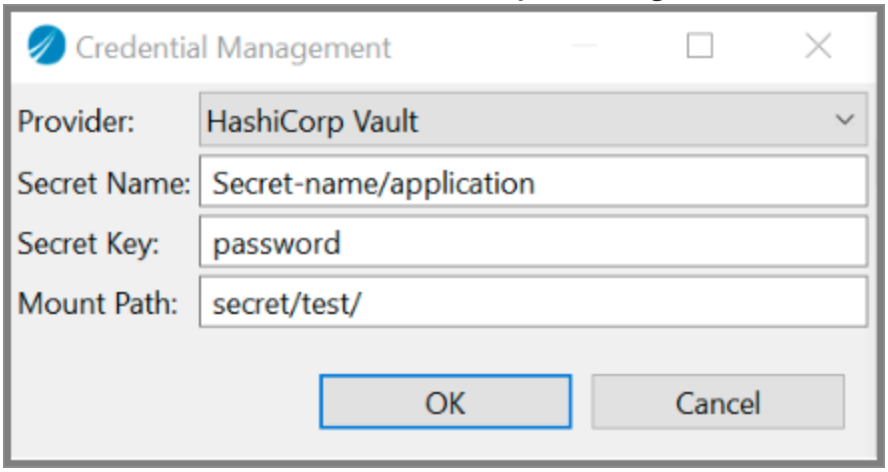
HashiCorp Vault is used to integrate with TIBCO BusinessWorks Container Edition for the credential management system to retrieve passwords from the vault to use it within the application at runtime when the password is required. This avoids adding any passwords in the deployment configuration and redeploying the application.

A new **HashiCorp Vault** provider is added for the credential management for property of type password.



HashiCorp Vault has the following fields:

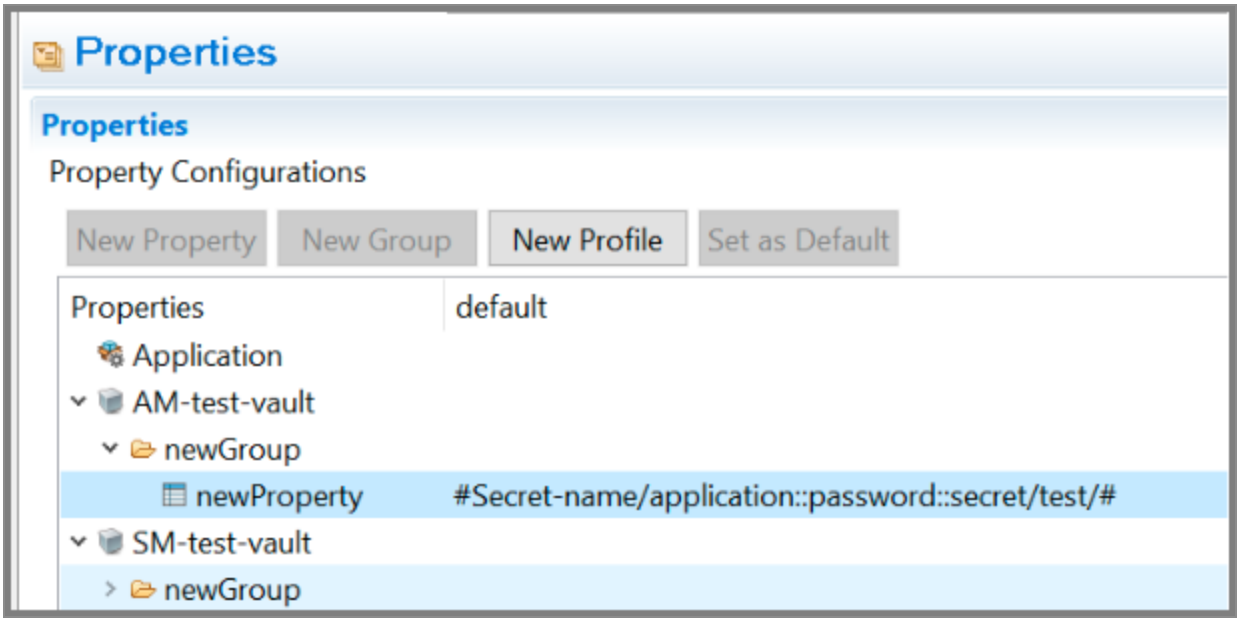
- **Secret Name**: Path of the secret.
- **Secret Key**: Key of the secret in the KeyValue engine.
- **Mount Path**: Path where the KeyValue engine is enabled.



On TIBCO Business Studio for BusinessWorks, the format is stored as `#Secret_Name::Secret_key::Mount_Path#`

You can use HashiCorp Vault as a credential management service for module properties. When using it to export the profile as a properties file, the property value uses the following format: `HashiCorpVault::secretName::secretKey::mountPath`.

Note: While creating a ConfigMap from the `deployment.yml` file, ensure that the value of the property must be in the format `HashiCorpVault::secretName::secretKey::mountPath`



To enable the HashiCorp Vault credential management system, pass the following environment variables at runtime.

- *HASHICORP_VAULT_ADDR*
- *HASHICORP_VAULT_AUTH*
- *VAULT_AUTH_PATH*
- *HASHICORP_VAULT_KV_VERSION*
- *APP_CONFIG_PROFILE*

To use the name of the namespace, pass the *HASHICORP_VAULT_NAMESPACE* environment variable.

For more information on the environment variables, see Environment Variables.

The authentication methods supported for HashiCorp Vault are Token, AppRole, and Userpass.

Authentication Method	Description	Environment Variables to enable the Authentication Method
Token	This authentication method allows users to	*HASHICORP_VAULT_*

Authentication Method	Description	Environment Variables to enable the Authentication Method
	authenticate using a token.	*TOKEN*
AppRole	This authentication method allows machines or applications to authenticate with vault-defined roles. The default path is approle/	If this authentication method is enabled at a different location, the environment variables used to enable this authentication method is: • *HASHICORP_VAULT_ROLE_ID* • *HASHICORP_VAULT_SECRET_ID*
Userpass	This authentication method allows users to authenticate with the vault using a username and password combinations. The username and password combinations are configured directly to the authentication method using the users or path.	• *HASHICORP_VAULT_USERNAME* • *HASHICORP_VAULT_PASSWORD*

Note: You can increase the security of the authentication methods by encrypting the parameters and passing them as environment variables.

For example, instead of passing Username and Password as plain text, you can encrypt the string and pass the string as an environment variable using the BWOfuscator Utility. For more information, see Password Obfuscator Utility.

In addition, to secure the string further, encrypt it using the custom key encryption feature of BWObfuscator Utility. In this case, you can pass the custom key to the application using the CUSTOM_ENCRYPTION_KEY environment variable. For more information, see Environment Variables for Docker.

Secrets engines are components that store, generate, or encrypt data. Some secrets engines simply store and read data while others connect to services and generate dynamic

credentials on demand. Other secrets engines also provide encryption as a service. Secrets engines are enabled at a "path" in the Vault. When a request comes to the Vault, the router automatically routes anything with the route prefix to the secrets engine. The supported secrets engines currently is the Key Value Engine.

The KeyValue (KV) engine is the supported secret engine for HashiCorp Vault. The default engine used is the KeyValue (KV) engine version 2. If the *HASHICORP_VAULT_KV_VERSION* environment variable is set to 1, KeyValue (KV) engine version 1 is used.

HTTPS Support:

The *HASHICORP_CACERT* environment variable needs to be set to the path of the certificate.

In TIBCO Business Studio for BusinessWorks provide the path of the certificate in the *HASHICORP_CACERT* environment variable.

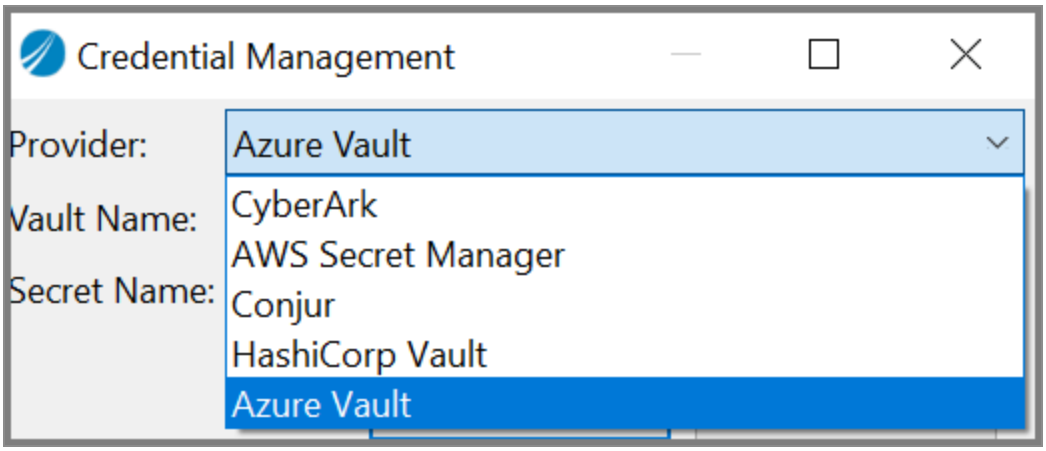
In Docker, place the certificate in the `/resources/addons/certs/` folder.

Note:
- The HashiCorp Vault is supported only on the Docker and Kubernetes platform.
- All the passwords fetched from the HashiCorp Vault are obfuscated.

Azure Vault for Credential Management Service

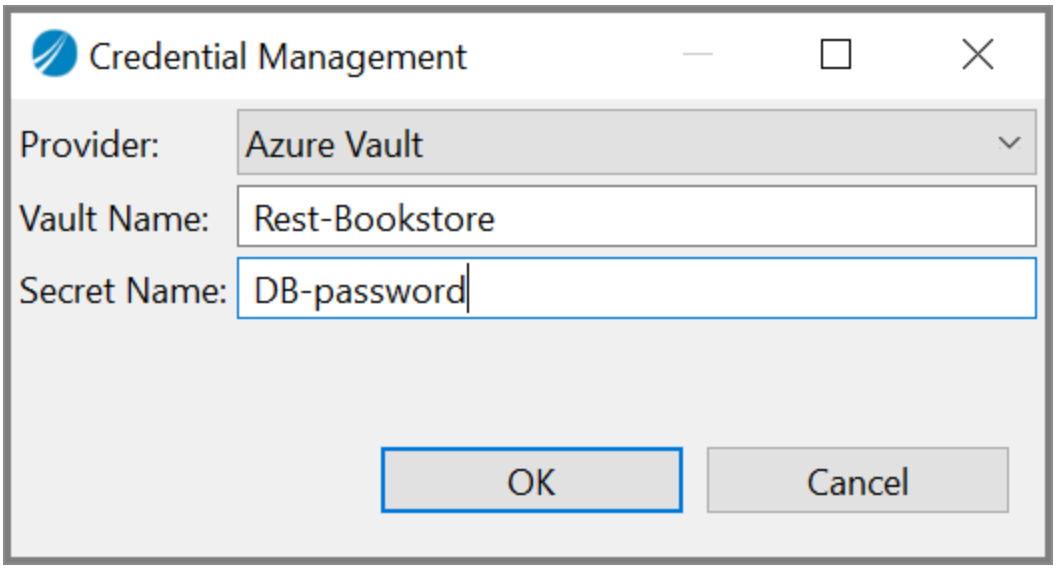
Azure Key Vault is a tool for securely storing and accessing secrets.

A new **Azure Vault** provider is added for the credential management for property of type password.

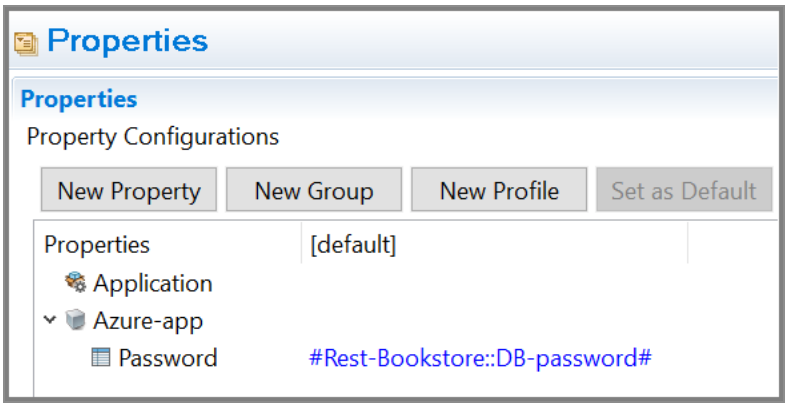


The **Azure Vault** has two fields:

- **Vault Name**: Name of the vault.
- **Secret Name**: Path of the Secret.



On TIBCO Business Studio for BusinessWorks, the format is stored as `#<AZURE_VAULT_NAME>::<AZURE_SECRET_KEY>#`.



You can use Azure Vault as a credential management service for module properties. When using it to export the profile as a properties file, the property value uses the following format: `AzureVault::vaultName::secretKey`.

> **Note:** While creating a ConfigMap from the `deployment.yml` file, ensure that the value of the property must be in the following format: `AzureVault::vaultName::secretKey`

TIBCO Business Studio for BusinessWorks supports two authorization methods to connect to the Azure Vault:

- Service Principle and secrets
- Managed identities for Azure Resources

To enable the Azure Vault credential management system, pass the following environment variables at runtime:

For Service Principle and Secrets

- *AZURE_VAULT*
- *APP_CONFIG_PROFILE*
- *AZURE_CLIENT_ID*
- *AZURE_CLIENT_SECRET*
- *AZURE_TENANT_ID*

For Managed identities for Azure Resources

- *AZURE_VAULT*
- *APP_CONFIG_PROFILE*

Note: Managed identities are used to when the application is running on Azure.

For more information on the environment variables, see Environment Variables.

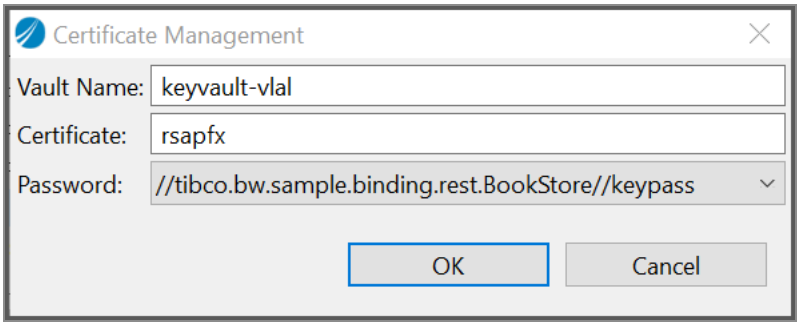
Azure Vault for Certificate Management

Azure Vault is used to integrate with TIBCO BusinessWorks Container Edition for certificate management to retrieve and store certificates from the vault to use within the applications.

This feature retrieves certificates and keys from the Azure Vault instead of maintaining a local copy of the certificate within the EAR application. With this configuration, the application is able to pull certificates from the Azure Vault and use that in the application.

The Azure Vault for certificate management has the following fields:

- **Vault Name**: Name of the vault.
- **Certificate**: Name of the certificate.
- **Password**: Name of the password field module property that specifies the password of the certificate. The module property can be selected from the dropdown menu.



To import the certificate in the Azure Vault from the command line, use the following command:

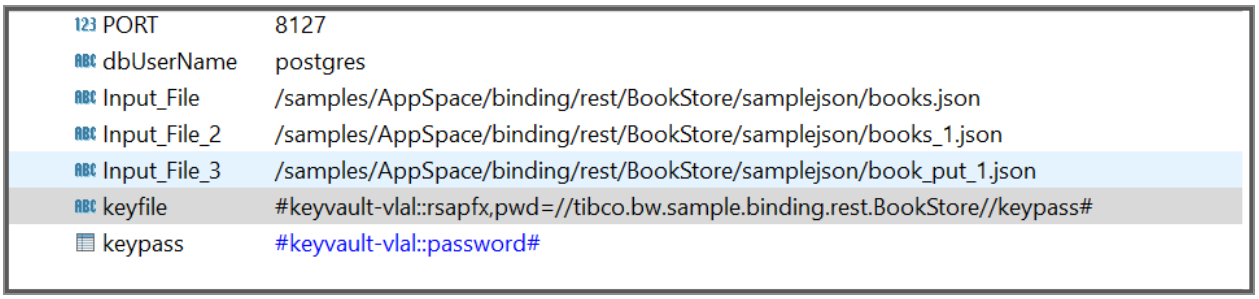
```
az keyvault certificate import --file <CERTIFICATE_TO_BE_IMPORTED> --name
<CERTIFICATE_NAME> --password <PASSWORD> --vault-name <KEY_VAULT_NAME>
```

CERTIFICATE_TO_BE_IMPORTED - Path of the certificate to be imported in the Azure Vault.

CERTIFICATE_NAME - Name of the certificate

KEY_VAULT_NAME - Name of the key vault used to store the certificate.

On TIBCO Business Studio for BusinessWorks, the format is stored as `#<KEY_VAULT_NAME>::<CERTIFICATE_NAME>::<PASSWORD>#`.



TIBCO Business Studio for BusinessWorks supports two authorization methods to connect to Azure Vault for certificate management:

- Service Principle and secrets
- Managed identities for Azure Resources

To enable the Azure Vault credential management system, pass the following environment variables at runtime:

For Service Principle and Secrets

- *AZURE_VAULT*
- *APP_CONFIG_PROFILE*
- *AZURE_CLIENT_ID*
- *AZURE_CLIENT_SECRET*
- *AZURE_TENANT_ID*

For Managed identities for Azure Resources

- *AZURE_VAULT*
- *APP_CONFIG_PROFILE*

Application Development for TIBCO Business Studio for BusinessWorks

The following section provides information about system module properties and environment variables as they apply to TIBCO Business Studio for BusinessWorks.

Environment Variables for TIBCO Business Studio for BusinessWorks

This section lists the environment variables that can be used for application deployment inside TIBCO Business Studio for BusinessWorks for configuration management.

Environment Variable	Default Values	Description
APPLICATION_NAME	n/a	This environment variable is used to provide custom application name to fetch application properties from config management.
APP_CONFIG_PROFILE	default	Name of the application profile that is to be used from a configuration management system.
BW_LOG_FORMAT	n/a	• Use value as JSON to change logging format of BWCE to JSON format. • Use value as 'Logstash JSON' to change logging format of BWCE to Logstash JSON format.
BW_LOGGER_ OVERRIDES	n/a	This environment variable contains the list of loggers and its values. The loggers are separated by whitespaces. For example,

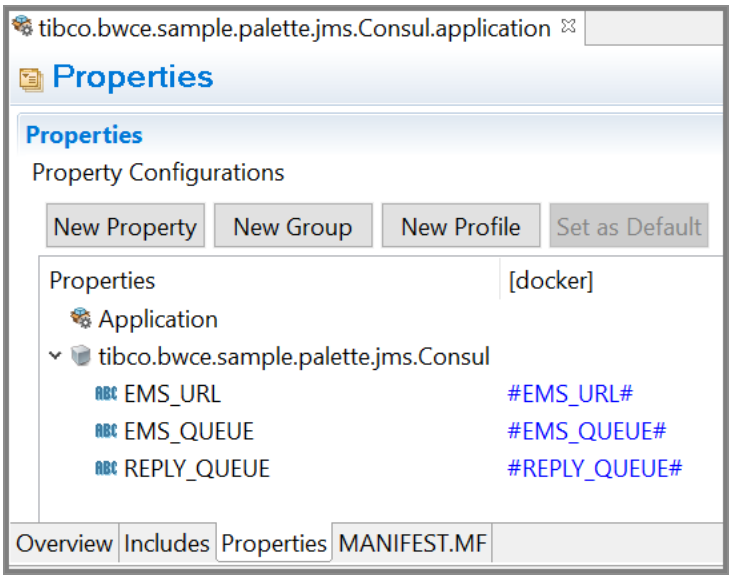
Environment Variable	Default Values	Description
		`BW_LOGGER_OVERRIDES="com.tibco.bw.binding.rest=DEBUG com.tibco.thor.frwk=DEBUG"`
BW_PROFILE_ENCRYPTION_KEYSTORE	n/a	The name of the KeyStore file.
BW_PROFILE_ENCRYPTION_KEYSTORETYPEn	n/a	The type of the KeyStore file. The common types are; jks, jceks, pkcs12.
BW_PROFILE_ENCRYPTION_KEYSTOREPASSWORD	n/a	Password of the private KeyStore.
BW_PROFILE_ENCRYPTION_KEYALIASPASSWORD	n/a	The alias password of the KeyStore.
BW_PROFILE_ENCRYPTION_KEYALIAS	n/a	The alias name used for the KeyStore.
CONSUL_SERVER_URL	n/a	Used to set Consul server configuration. For example, `CONSUL_SERVER_URL=http://127.0.0.1:8085` This must be set if you intend to use Consul for application configuration or for service registration and discovery.
CONSUL_CUSTOM_FOLDER	n/a	This must be set if you only intend to use properties from Consul Server stored in custom folders. If properties are stored or exported to `<CONSUL_`

Environment Variable	Default Values	Description
		`SERVER_URL>/ui/dc1/kv/FOLDER1/FOLDER2` location then the value for CONSUL_CUSTOM_FOLDER = FOLDER1/FOLDER2
CONSUL_FORMAT	n/a	Set the value of then environment variable to YAML to use the yaml file for configuration management service using Consul. Set the value of then environment variable to PROP to use the properties file for configuration management service using Consul.
CONSUL_DATA_KEY=<KEY_NAME>	n/a	The value of this property is the custom location of the yaml file or the properties files stored on Consul.
X_CONSUL_TOKEN	n/a	Used to authenticate the request made to the Consul server. This environment variable should be used when the consul server is enabled for authentication. **Note:** This should be set if authentication is enabled on the consul server and you want to authenticate the requests made to the consul server.

Using Consul as a Configuration Management Service from TIBCO Business Studio for BusinessWorks

You can use configurations from the configuration management services such as Consul, by defining the token as `#`*<property name>*`#` in the application properties, where *<property name>* is the name of the configuration parameter.

For example, `#EMS_URL#`.



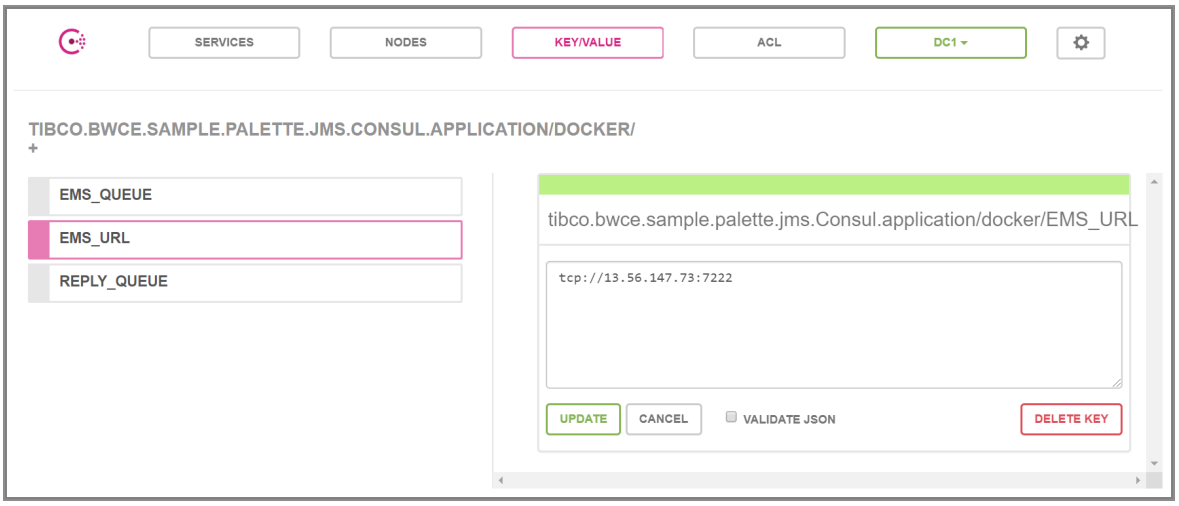
Follow these steps to use configurations from the Consul:

1. In TIBCO Business Studio for BusinessWorks, on the **Menu** bar select **Run > Run Configurations > Environment** and set the environment variables to `CONSUL_SERVER_URL` and `APP_CONFIG_PROFILE`. For more information, see Environment Variables for TIBCO Business Studio for BusinessWorks.

   > **Note:**
   > The X_CONSUL_TOKEN environment variable should be used when authentication is enabled on the Consul Server.

2. Select the desired profile as the default profile inside TIBCO Business Studio for BusinessWorks in order to fetch the values from the Consul Server.
3. In your Consul Service, Service, define the keys using the format `<BWCE_APP_NAME>/<PROFILE_NAME>/<KEY_NAME>`.

   For example, `tibco.bwce.sample.palette.jms.Consul.application/docker/EMS_URL`



Adding the YAML or Properties file for Configuration Management using Consul

Follow the steps to add the properties or YAML file for configuration management using Consul:

For YAML file:

1. By default, the YAML file is stored at `<APPLICATION_NAME>/<APP_PROFILE_NAME>` on Consul. Set the environment variable `CONSUL_SERVER_URL`, `APP_CONFIG_PROFILE`, and `CONSUL_FORMAT=YAML`. For more information, see Environment Variables for TIBCO Business Studio for BusinessWorks.
2. This YAML file can also be stored at another key location on Consul by passing an additional environment variable `CONSUL_DATA_KEY=<KEY_NAME>`, where *<KEY_NAME>* is the custom location of the YAML file stored on Consul.
3. In TIBCO Business Studio for BusinessWorks, configure the property by defining the token as `<#KEY_IN_YAML_FILE#>`. If a nested key is used, tokenize the key with the entire nested path and separate the key name by "//".

For properties file:

1. By default, the properties file is stored at `<APPLICATION_NAME>/<APP_PROFILE_NAME>` on the Consul. Set the environment variable `CONSUL_SERVER_URL`, `APP_CONFIG_PROFILE`, and `CONSUL_FORMAT=PROP`. For more information, see Environment Variables

for TIBCO Business Studio for BusinessWorks.

2. This properties file can also be stored at another key location on the Consul by passing an additional environment variable `CONSUL_DATA_KEY=<KEY_NAME>`, where *<KEY_NAME>* is the custom location of the properties file stored on the Consul.
3. In TIBCO Business Studio for BusinessWorks, configure the property by defining the token as `<#KEY_TIBCO Business Studio™ for BusinessWorks™ N_PROP_FILE#>`.

Support for using HTTPS enabled Consul Server as a configuration management Service.

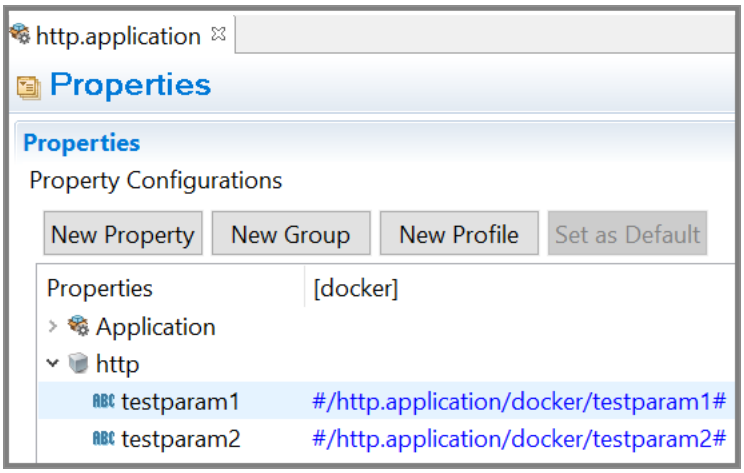
To connect Consul via HTTPS (SSL) for TIBCO Business Studio for BusinessWorks, the SSL certificate needs to be added to the Java CA keystore at `BW_HOME/tibcojre64/1.8.0/lib/security/cacerts`.

To connect Consul via HTTPS (SSL) for Docker, copy the SSL certificate to the `resources/addons/certs/` folder. Also copy the `libsunec.so` library found in `jre-8u<no>-linux-x64.rpm/tar.gz`, which is available in the Oracle resource library to the `/resources/addons/lib` folder before you create the base image.

Using AWS as a Configuration Management Service from TIBCO Business Studio for BusinessWorks

You can use the AWS Systems Manager parameter store for configuration management services in TIBCO Business Studio for BusinessWorks by defining a token such as `#/<BWCE_APP_NAME>/<PROFILE_NAME>/<property name>#` in the application properties, where *<property name>* is the name of the configuration parameter.

For example, `#/http.application/docker/testparam1#`.

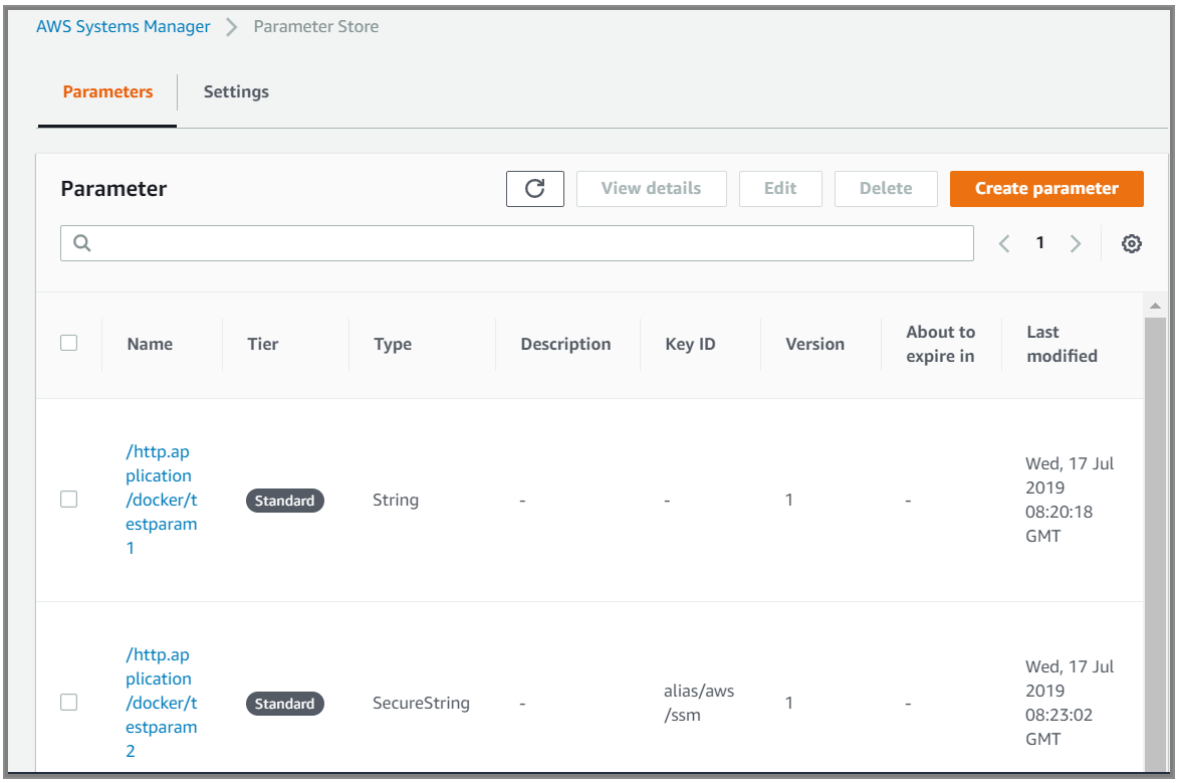


Procedure

Follow these steps to use configurations from AWS systems manager parameter store in TIBCO Business Studio for BusinessWorks:

1. Set the environment variables `AWS_ACCESS_KEY`, `AWS_SECRET_KEY`, `AWS_REGION`, `AWS_PARAMETER_STORE`, and `APP_CONFIG_PROFILE`. The `AWS_ACCESS_KEY` and `AWS_SECRET_KEY` environment variables are used as credentials for authentication. In order to enable assume role, the following additional environment variables should be passed, `AWS_ROLE_ARN`, `AWS_ROLE_SESSION_NAME`, `AWS_EXTERNAL_ID` (optional), and `AWS_EXPIRATION_DURATION` (optional). For more information on the environment variables, see Environment Variables for Docker.
2. Select the desired profile as the default profile inside TIBCO Business Studio for BusinessWorks in order to fetch the values from the AWS Server.
3. In your AWS parameter store, define the keys using the format `/<BWCE_APP_NAME>/<PROFILE NAME>/<KEY Name>`.

   For example, `/http.application/docker/testparam1`



Integrating ECS and EKS services with AWS parameter store

Support for configuration management service for the AWS Parameter Store is provided while deploying the application on ECS and EKS services.

Set the environment variables `APP_CONFIG_PROFILE` and `AWS_PARAMETER_STORE` to configure ECS and EKS services with the AWS Parameter Store.

To use the AWS Parameter Store with EKS service for password-less solution, deploy the application on EKS cluster configured to a service account, OIDC provider, and an IAM role associated with the cluster, pass the environment variables `APP_CONFIG_PROFILE` and `AWS_PARAMETER_STORE`. The AWS credentials do not need to be passed. For more information, see https://aws.amazon.com/it/blogs/opensource/introducing-fine-grained-iam-roles-service-accounts/

Note: The AWS credentials need to be provided while running the application on TIBCO Business Studio for BusinessWorks or Docker platform.

Application Development for Cloud Foundry

The following section provides information about system module properties and environment variables as they apply to TIBCO BusinessWorks Container Edition.

Environment Variables for Cloud Foundry

This section lists the environment variables that can used for TIBCO BusinessWorks Container Edition application deployment on Cloud Foundry platform.

Environment Variable	Default Values	Description
APP_CONFIG_ PROFILE	default	Name of the application profile that is to be used from a configuration management system such as ZUUL or Spring Cloud Config.
BW_LOG_FORMAT	n/a	• Use value as JSON to change logging format of BWCE to JSON format. • Use value as 'Logstash JSON' to change logging format of BWCE to Logstash JSON format.
BW_LOGLEVEL	ERROR	Used to set a log level for the TIBCO BusinessWorks Container Edition application. The default value is ERROR. Supported values are: • INFO • DEBUG • WARN • ERROR
BW_ENGINE_	8	Used to set engine thread count for the TIBCO

Environment Variable	Default Values	Description
THREADCOUNT		BusinessWorks Container Edition application.
BW_ENGINE_ STEPCOUNT	-1	Used to set engine step count for the TIBCO BusinessWorks Container Edition application.
BW_ APPLICATION_ JOB_FLOWLIMIT	n/a	Used to set flow limit for TIBCO BusinessWorks Container Edition application.
BW_PROFILE	n/a	Used to set the name of the BusinessWorks profile from the application.
BW_JAVA_OPTS	n/a	Used to set Java properties that are used at run time. The properties are specified using name-value pairs and are separated by spaces. For example, `BW_JAVA_OPTS="-Dname=value -Dname=value"`
BW_ COMPONENT_ JOB_FLOWLIMIT	n/a	Used to set flow limit at component level for TIBCO BusinessWorks Container Edition application. For example, `BW_COMPONENT_JOB_FLOWLIMIT=<applicationn version>.componentName`
MASHERY_ SERVICE_CONFIG		Applications can pass TIBCO Mashery configuration through the `MASHERY_SERVICE_CONFIG` environment variable. The value of the environment variable is a JSON string with the required TIBCO Mashery configuration. For more information, see Integrating with TIBCO Mashery.
ENABLE_ SERVICE_DIRECT_	false	If set to `true`, an application is registered using its host IP and port instead of the application route.

Environment Variable	Default Values	Description
REGISTRATION		
CONSUL_ SERVER_URL		Used to set Consul server configuration. For example, `CONSUL_SERVER_URL=http://127.0.0.1:8085`. This must be set if you intend to use Consul for application configuration or for service registration and discovery.
EUREKA_SERVER_ URL		Used to set Eureka server configuration. For example, `EUREKA_SERVER_URL=http://127.1.0.1:8080`/eureka This must be set if you intend to use Eureka for service registration and discovery.
CF_TARGET	n/a	To add an Elastic Runtime self-signed certificate automatically at runtime, set `CF_TARGET` to the API endpoint of your Elastic Runtime instance. For example, `CF_TARGET=https://api.cf.demo.com.` This is useful when using Spring Cloud Services in your application.
BW_JMX_CONFIG	n/a	Used to set JMX configuration (JMX port) for monitoring TIBCO BusinessWorks Container Edition application. For example, `BW_JMX_CONFIG: 8050`
BW_JAVA_GC_ OPTS	-XX:+UseG1GC	Used to set JAVA GC configuration. The value should be one of the standard Java GC VM Options.

Environment Variable	Default Values	Description
		For example, `BW_JAVA_GC_OPTS: -XX:+UseParallelGC`
DISABLE_BWCE_EAR_VALIDATION	False	Used to deploy the ActiveMatrix BusinessWorks 6.x application `EAR` file on TIBCO BusinessWorks Container Edition without converting project to Container Edition and rebuilding EAR file from TIBCO Business Studio for BusinessWorks **Note:** Ensure that the ActiveMatrix BusinessWorks 6.x EAR file is exported. ActiveMatrix BusinessWorks 6.x EAR file should only have TIBCO BusinessWorks Container Edition supported activities and features.
BREAK_CIRCUIT_ON_404_ERROR	True	Used to change the behavior of the Circuit Breaker functionality. By default, circuit breaks for 404 (Not Found) error code. It does not break the circuit for 404 error code on specifying the value of the variable as `false`.
SPRING_CLOUD_CONFIG_VAULT_TOKEN	n/a	Used to pass the Vault token for VMware Tanzu application, while using Vault as backend for Spring Cloud Config.
CUSTOM_LOGBACK	False	Used for customizing logs.

The TIBCO BusinessWorks Container Edition Buildpack

The TIBCO BusinessWorks Container Edition Buildpack must be uploaded in your Cloud Foundry environment before deploying any applications.

Customize the Buildpack

This Buildpack can be customized for the supported third-party drivers, OSGI bundles, integration with application configuration management systems, and application certificate management.

> **Note:** Customization of the Buildpack is not supported on the Windows platform.

Procedure

1. Download the TIBCO BusinessWorks Container Edition Buildpack zip file from http://edelivery.tibco.com.

   To download this file:

   a. Select **Container** from the `Operating Systems` dropdown list.

   b. Read and accept the **TIBCO End User License Agreement**.

   c. Select the radio button for **Individual file Download**.

   d. Click **+** sign to view the individual components and select `bwce_buildpack_cf-vx.x.x.zip`.

2. Extract the contents of the zip file to a temporary location.
3. Customize the Buildpack for the database drivers.

   a. Follow the steps outlined in **JBDC Connection** in the *TIBCO BusinessWorks™ Container Edition Bindings and Palettes Reference*.

   b. Copy the appropriate driver bundle from *TIBCO_HOME*/bwce/*version*/config/drivers/shells/<*driverspecific runtime*>/runtime/plugins/ to the <*your local buildpack repo*>/resources/addons/jars folder in your temporary location.

4. Provision the OSGi bundle jars.

   Copy the OSGi bundle jars into the <*your buildpack repo*>/resources/addons/jars folder in your temporary location.

5. Application Configuration Management

   For more information, see Using Configurations from Configuration Management Services for Cloud Foundry.

6. Certificate Management

   Certificates are used by applications to connect to different systems. For example, a

certificate to connect to Spring Cloud Config service or a certificate to connect to TIBCO Enterprise Message Service.

Bundling certificates with your application is not a good idea as you would need to rebuild your application when the certificates expire. To avoid that, copy your certificates into the `<your local buildpack repo>/resources/addons/certs` folder in your temporary location.

Once the certificates expire, you can copy the new certificates into the Buildpack without rebuilding your application. Deploy your application with the new Buildpack. To access the certificates folder from your application, use the environment variable `BW_KEYSTORE_PATH`. For example, `#BW_KEYSTORE_PATH#/mycert.jks` in your application property.

7. Provision the BusinessWorks Container Edition Plug-in Runtime

   To add TIBCO certified plug-ins:

   a. Download the appropriate Plug-in packaging. For example, TIBCO ActiveMatrix BusinessWorks™ Plug-in for WebSphere MQ, from https://edelivery.tibco.com.

   b. Locate the plug-in zip file, `<product id>_ePaas.zip` or `TIB_<version>_<build number>_bwce-runtime.zip` from the downloaded artifacts and copy into `<your local buildpack repo>/resources/addons/plugins`.

   To add a plug-in created using the TIBCO ActiveMatrix BusinessWorks™ Plug-in Development Kit to the runtime into your Buildpack:

   a. Install the plug-in if it is not installed.

   b. Navigate to the `TIBCO_HOME/bwce/palettes/<plug in name>/<plugin version>` directory and zip the `lib` and `runtime` folders into `<plugin name>.zip`.

   c. Copy `<plugin name>.zip` to the `<your buildpack repo>/resources/addons/plugins` folder.

   Copy any required OSGi bundles. For example, driver bundles into `<your buildpack repo>/resources/addons/jars`

8. Provision to add third-party client installation at runtime

   a. Package third-party client installation into zip.

   b. Copy zip file to `<YOUR-BUILDPACK-REPO>/resources/addons/thirdparty-`

`installs` folder.

9. Provision to add custom JDBC driver
    a. For more information, see the "Enabling Custom Drivers" section of the **JDBC Connection** topic in the *TIBCO BusinessWorks™ Container Edition Bindings and Palette Reference*.
    b. After the project has been exported as a plug-in to the location you specified, locate the JAR file in the plug-in folder, and copy paste the JAR to the *<YOUR-BUILDPACK-REPO>*`/resources/addons/jars` folder.
10. Provision to add custom JMS driver
    a. For more information, see the "Enabling Custom Drivers" section of the **JNDI Connection** topic in the *TIBCO BusinessWorks™ Container Edition Bindings and Palette Reference*.
    b. After the project has been exported as a plug-in to the location you specified, locate the JAR file in the plug-in folder, and copy paste the JAR to the *<YOUR-BUILDPACK-REPO>*`/resources/addons/jars` folder.
11. Provision to use custom logs
    a. Create a new folder `custom-logback` in the *<YOUR-BUILDPACK-REPO>*`/resources/addons` folder.
    b. Add the customized logback file in the folder. The name of the logback file should be `logback.xml`.

    > **Note:** While running the application, set the environment variable CUSTOM_LOGBACK="true".

12. Provision to add multiple Buildpack

    To push an application with multiple Buildpack, specify each Buildpack with a `-b` flag and run the following command: `cf push YOUR-APP -b BUILDPACK-NAME-1 -b BUILDPACK-NAME-2 ... -b BUILDPACK-NAME-3`

    Where:

    - `YOUR-APP` is the name of the application.
    - `BUILDPACK-NAME-1 -b BUILDPACK-NAME-2 ... -b BUILDPACK-NAME-3` are the

names of the Buildpack that you want to push with the application.

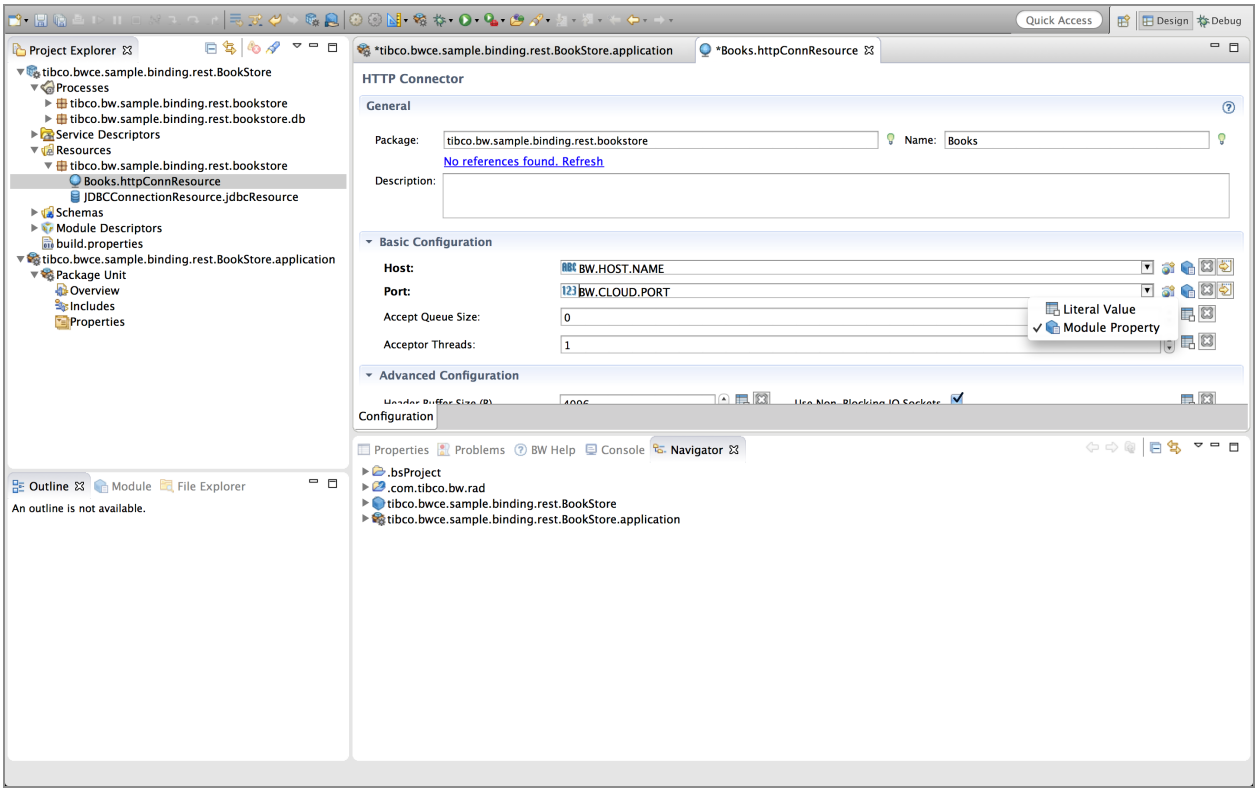
The GitHub branch location from where multiple Buildpack are created is `https://github.com/TIBCOSoftware/bwce-buildpack/tree/multi-buildpack-support`

The last Buildpack specified is the final TIBCO BusinessWorks Container Edition Buildpack, which modifies the open environment and sets the start command.

13. Zip the contents of the temporary location to create the TIBCO BusinessWorks Container Edition Buildpack zip file.
14. Push the Buildpack to the Cloud Foundry environment.

System Module Properties

The web applications deployed in the Cloud Foundry are expected to bind themselves to the port given by the Cloud Foundry. This can be achieved for TIBCO BusinessWorks Container Edition applications by configuring the `BW.CLOUD.PORT` system property for the HTTP Connector Shared Resource.



Using Cloud Foundry Services

To integrate with the Cloud Foundry database or messaging services (CUPS or Managed) follow the steps outlined in **Modifying Application Properties** in the *TIBCO BusinessWorks Container Edition Application Development*.

Modifying Application Properties

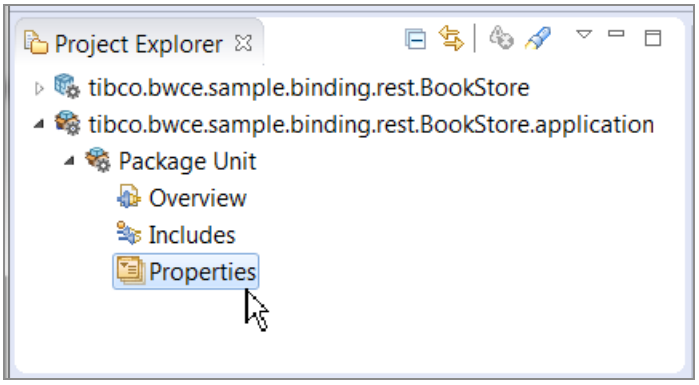
In order to make use of configuration from the Cloud Foundry runtime you need to modify the application properties.
Follow these steps to modify application properties.

The property names are specified in the format `#<serviceName>.credentials.<parameter>#`, where <*serviceName*> is the name of the service and <*parameter*> is the name of the configuration parameter.

For example, `#postgres.credentials.username#`.

Procedure

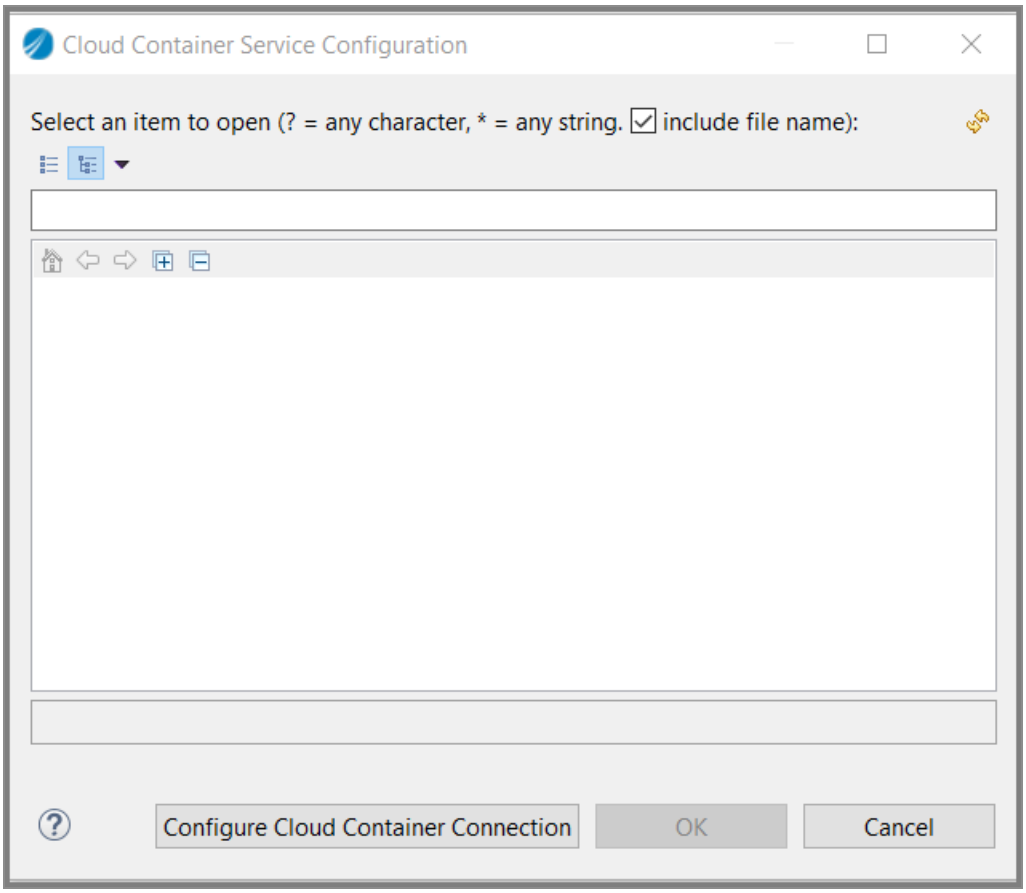
1. In the **Project Explorer** view, expand the project application and double click **Properties**.



   The properties are now displayed in the Editor view.

2. To modify the value of a property, first click the property name and the click the icon.

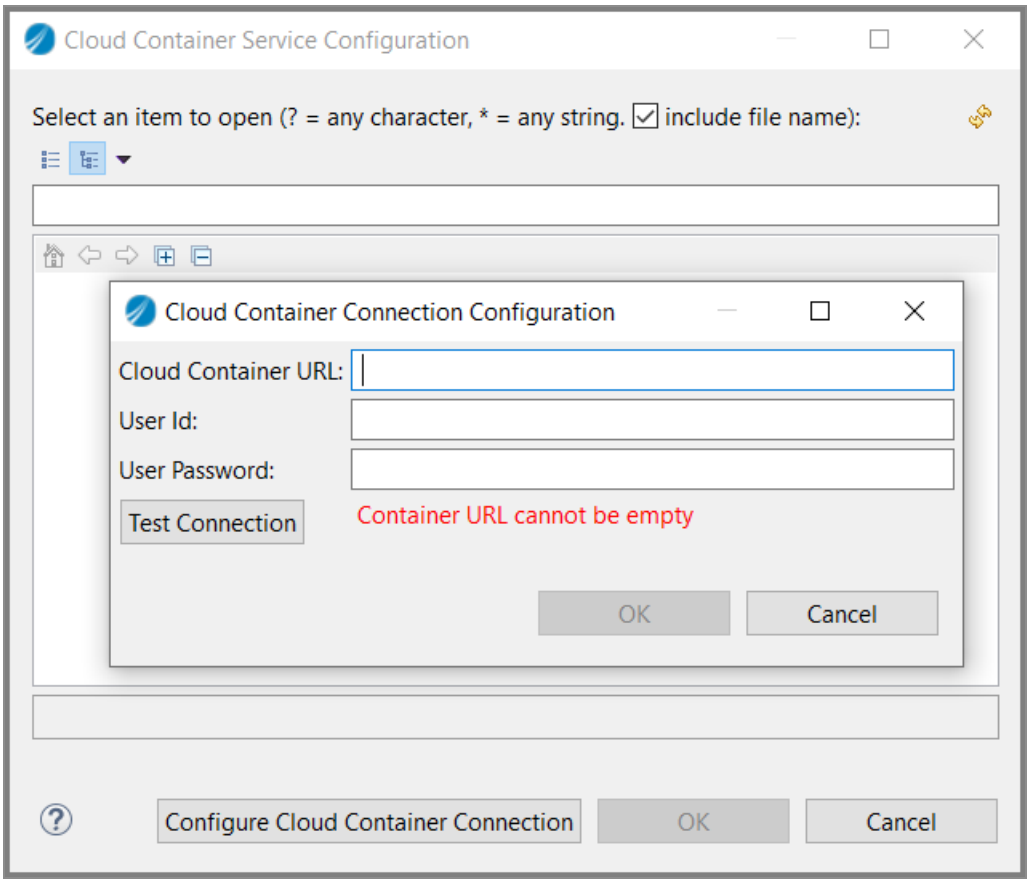
   The **Cloud Container Service Configuration** window is displayed.



This window initially contains no values.

3. Click **Cloud Container Connection**.

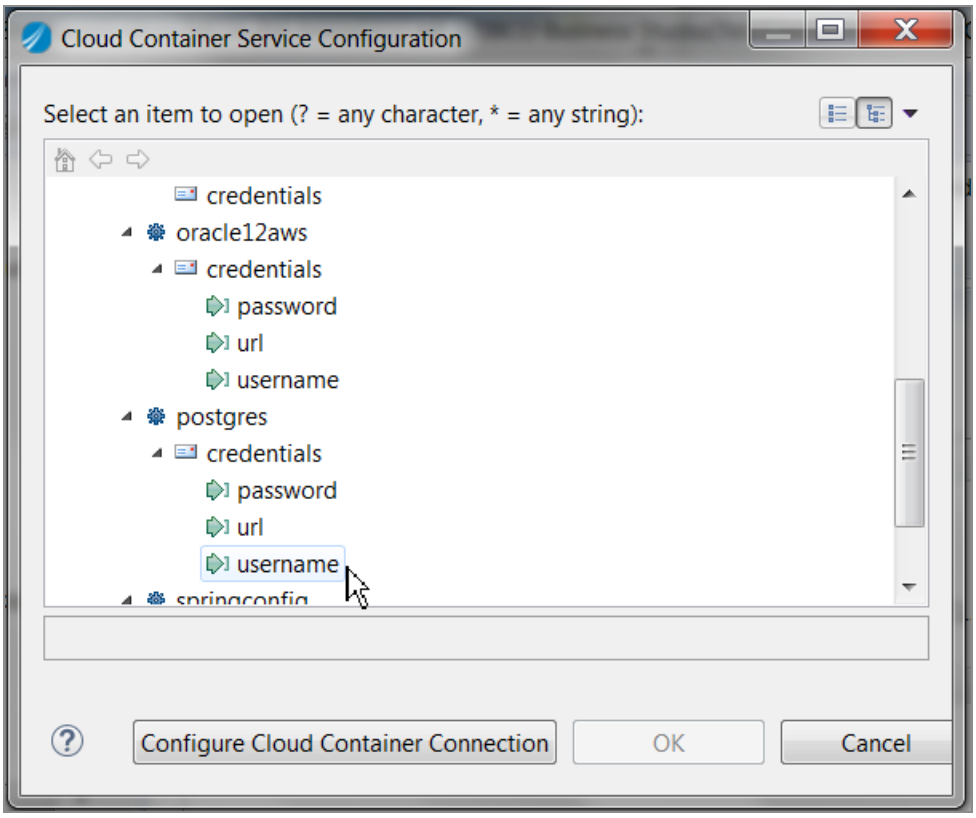
   The **Cloud Container Connection Configuration** window is displayed.



Note: The **Cloud Container Connection Configuration** window can be resized according to individual requirements.

4. Specify the configuration properties and click **Test Connection**.

   If the connection is successful, variables are populated in the **Cloud Container Service Configuration** window as shown in the next figure.



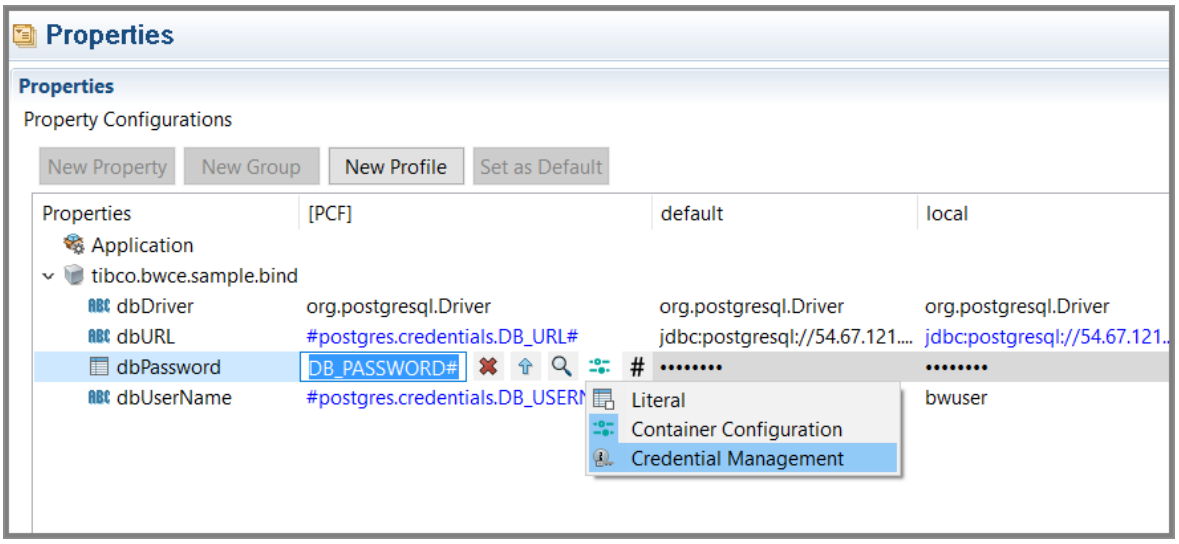
5. Choose the appropriate value and click **OK**.
6. Click **Save**.

Modifying Properties of Type Password or Integer

Follow these steps to modify properties of type Password or Integer.

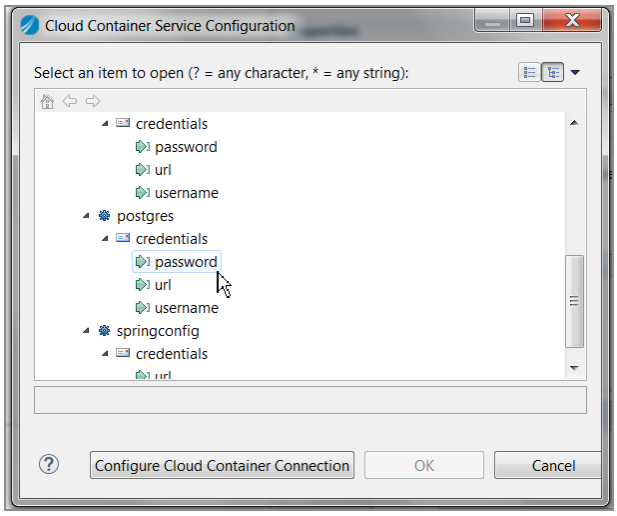
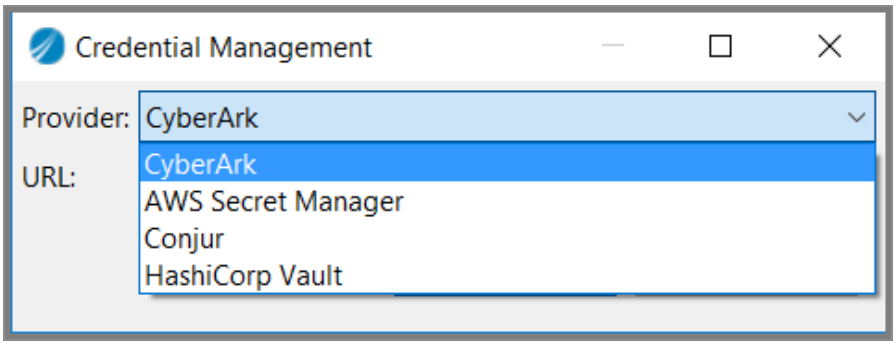
Procedure

1. Click on the property of type Password or Integer and then click .
2. Choose either of the following options: Literal, Container Configuration or Credential Management.



3.

Option	Description
Literal	Type the value for the property.
Container Configuration	Choose the value from the cloud container configuration. a. Click the icon that is now visible. b. Click **Configure Cloud Container Connection** in the **Cloud Container Service Configuration** window. c. Specify the configuration properties and click **Test Connection**. d. If the connection is successful, variables are populated in the **Cloud Container Service Configuration** window.

Option	Description
	
Credential Management	Credential management is used to fetch the password from various credential management systems. This option is available only for the application properties of type password in the Container and Cloud edition projects. The supported credential management systems are CyberArk, AWS Secret Manager, Conjur and HashiCorp Vault.  For more information of credential management services, see Credential Management Service for Properties of Type Password.

4. Click **Save**.

Integrating with TIBCO Mashery®

The following options are available to integrate with TIBCO Mashery:

- Using a custom user provided service (CUPS)

  In your cf CLI execute

  ```
  cf cups mashery-service -p
  "areaUuid,clientId,clientSecret,password,trafficManagerDomain,usern
  ame,masheryApiServerUri"
  ```

  All the service configuration parameters listed above are mandatory and the service name has to contain the word `mashery`.

- Using an environment variable The TIBCO Mashery configuration can be passed using the `MASHERY_SERVICE_CONFIG` environment variable. The value of this environment variable is a JSON string.

  ```
  MASHERY_SERVICE_CONFIG: '{"username":"xxx@tibco.com","password":
  "xxxx",
  "clientId":"xxxxxxxx","clientSecret":"xxxxxx","masheryApiServerUr
  i": "https://api.mashery.com","areaUuid": "xxxxxxxxx-xxxx-xxxx-
  xxxx-xxxxxxxxx","trafficManagerDomain":"xxxxx.api.mashery.com"}'
  ```

  The TIBCO BusinessWorks Container Edition runtime looks for an attached service or an environment variable for the TIBCO Mashery configuration. If found, that configuration is used to register HTTP based endpoints (HTTP, REST, or SOAP/HTTP).

Application Development for Docker

The following section provides information about system module properties and environment variables as they apply to TIBCO BusinessWorks™ Container Edition.

Environment Variables for Docker

This section lists the environment variables that can be used for TIBCO BusinessWorks Container Edition application deployment on Docker and Docker-based platforms.

Environment Variable	Default Values	Description
APPLICATION_NAME	n/a	This environment variable is used to provide custom application name to fetch application properties from config management.
APP_CONFIG_PROFILE	n/a	The name of the application profile that is to be used from a configuration management system such as ZUUL or Spring Cloud Config.
AWS_ACCESS_KEY	n/a	It is the access key used to access the AWS account from where the parameters are fetched. **Note:** This must be set if you want to use AWS Systems Manager Parameter Store or AWS secret manager for application configuration and secrets management.
AWS_EXTERNAL_ID	n/a	Optional. A unique identifier required when you assume a role in another account. It is used to address the confused deputy problem.

Environment Variable	Default Values	Description
AWS_ EXPIRATION_ DURATION (ms)	n/a	Optional. Parameter to specify the duration in minutes for which the temporary security credentials remain valid using AssumeRole.
AWS_ PARAMETER_ STORE	True	This environment variable is optional while running applications on TIBCO Business Studio for BusinessWorks or Docker platforms. It is mandatory while deploying applications on ECS services.
AWS_REGION	n/a	It is the region of your AWS account. **Note:** This must be set if you want to use AWS Systems Manager Parameter Store or AWS secret manager for application configuration and secrets management.
AWS_ROLE_ARN	n/a	This is required to use assumeRole for authentication. The Amazon Resource Name (ARN) of the role to assume.
AWS_ROLE_ SESSION_NAME	n/a	This environment variable is required to use assumeRole for authentication. An identifier for the assumed role session is used to identify uniquely a session when the same role is assumed by different principals or for different reasons. Please refer to the AWS documentation for further details on RoleSessionName.
AWS_SECRET_KEY	n/a	It is the secret key used to access the AWS account from where the parameters are fetched. **Note:** This must be set if you want to use AWS Systems Manager Parameter Store or AWS secret manager for application configuration and secrets management.

Environment Variable	Default Values	Description
AWS_SECRET_MANAGER	True	Set to use the AWS secret manager as the credential management system.
AZURE_CLIENT_ID	n/a	It is the Client ID of the service principle.
AZURE_CLIENT_SECRET	n/a	It is the Secret of the service principle.
AZURE_TENANT_ID	n/a	It is the Tenant ID of the Azure account.
AZURE_VAULT	True	This environment variable is set to true to enable credential management using the Azure Vault.
AWS_SESSION_TOKEN	n/a	AWS uses the session token to validate the temporary security credentials. When a call is made using temporary credentials, the call includes a session token, which returns along with the temporary credentials
AWS_ACCESS_KEY_ID	n/a	It is the access key used to access the AWS account. This is in case of temporary security credentials.
AWS_SECRET_ACCESS_KEY	n/a	It is the secret key used to access the AWS account. This is in case of temporary security credentials.
BW_LOG_FORMAT	n/a	• Use value a as JSON to change the logging format of TIBCO BusinessWorks Container Edition to JSON format. • Use value a as 'Logstash JSON' to change the logging format of TIBCO BusinessWorks Container Edition to Logstash JSON format.
BW_LOGLEVEL	ERROR	Used to set a log level for the TIBCO BusinessWorks Container Edition application. The default value is ERROR. Supported values are:

Environment Variable	Default Values	Description
		• INFO • DEBUG • WARN • ERROR
BW_ENGINE_ THREADCOUNT	8	Used to set engine thread count for the TIBCO BusinessWorks Container Edition application.
BW_ENGINE_ STEPCOUNT	-1	Used to set engine step count for the TIBCO BusinessWorks Container Edition application.
BW_ APPLICATION_ JOB_FLOWLIMIT	n/a	Used to set Flow limit for TIBCO BusinessWorks Container Edition application.
BW_PROFILE	n/a	Used to set the name of the BusinessWorks profile from the application.
BW_JAVA_OPTS	n/a	Used to set Java properties that are used at run time. The properties are specified using name-value pairs and are separated by spaces. For example, `BW_JAVA_OPTS="-Dname=value -Dname=value"` **Note:** BW_JAVA_OPTS environment variable can also be used to hardcore the heap parameters.
BW_JMX_CONFIG	n/a	Used to set JMX configuration (RMI host and JMX port) for monitoring TIBCO BusinessWorks Container Edition application. The value should be provided in RMI_ HOST:JMX_PORT format. For example, `BW_JMX_CONFIG=192.168.99.100:8050`

Environment Variable	Default Values	Description
BW_JAVA_GC_OPTS	-XX:+UseG1GC	Used to set JAVA GC configuration. The value should be one of the standard Java GC VM Options. For example: `BW_JAVA_GC_OPTS=-XX:+UseParallelGC`
BREAK_CIRCUIT_ON_404_ERROR	True	Used to change the behavior of the Circuit Breaker functionality. By default, the circuit breaks for the 404 (Not Found) error code. It does not break the circuit for 404 error code on specifying the value of the variable as `false`.
BW_COMPONENT_JOB_FLOWLIMIT	n/a	Used to set Flow limit at component level for TIBCO BusinessWorks Container Edition application. For example, `BW_COMPONENT_JOB_FLOWLIMIT=<applicationn version>.componentName`
BW_OSGI_SSH_PORT	n/a	This is the BWAppNode OSGi Console SSH port. It is used for OSGI remote access. Without passing this environment variable, the OSGI Port the default value configured is 1122.
BW_REST_DOCAPI_PORT	n/a	This is the Swagger framework port. This environment variable configures the port on which the Swagger framework serves the API documentation endpoint. The default value configured for this port i 7777.
BW_OSGI_SERVICE_PORT	n/a	This environment variable configures the OSGI Service Port. Without passing this environment variable, the default value configured for this port is 8090.
CONJUR_ACCOUNT	n/a	The account specified during setting up the Conjur.

Environment Variable	Default Values	Description
CONJUR_ APPLIANCE_URL	n/a	It is the Conjur HTTPS endpoint (OSS/DAP).
CONJUR_AUTHN_ LOGIN	n/a	It is the user or host identity.
CONJUR_AUTHN_ API_KEY	n/a	It is the user or host API Key.
CONSUL_ SERVER_URL	n/a	Used to set Consul server configuration. For example, `CONSUL_SERVER_URL=http://127.0.0.1:8085` This must be set if you intend to use Consul for application configuration or for service registration and discovery.
CONSUL_ CUSTOM_FOLDER	n/a	This must be set if you only intend to use properties from the Consul Server stored in custom folders. If properties are stored or exported to <CONSUL_ SERVER_URL>/ui/dc1/kv/FOLDER1/FOLDER2 location, then the value for CONSUL_CUSTOM_FOLDER = FOLDER1/FOLDER2
CUSTOM_ ENCRYPTION_KEY	n/a	The custom encryption key used in BWObfuscator.
CUSTOM_LOG_ BACK	False	Used for customizing logs.
CYBERARK	True	Set to use Cyberark as the credential management system.
CYBERARK_ KEYSTORE_PATH	n/a	The location of the JKS file.

Environment Variable	Default Values	Description
CYBERARK_ KEYSTORE_ PASSWORD	n/a	The password of the JKS file.
DISABLE_BWCE_ EAR_VALIDATION	False	Used to deploy the ActiveMatrix BusinessWorks 6.x application EAR file on TIBCO BusinessWorks Container Edition without converting the project to TIBCO BusinessWorks Container Edition and rebuilding the EAR file from TIBCO Business Studio for BusinessWorks. **Note:** Ensure that the ActiveMatrix BusinessWorks 6.x EAR file is exported. ActiveMatrix BusinessWorks 6.x EAR file should only have TIBCO BusinessWorks Container Edition supported activities and features.
EUREKA_SERVER_ URL		Used to set Eureka server configuration. For example, `EUREKA_SERVER_URL=http://127.1.0.1:8080/eureka` This must be set if you intend to use Eureka for service registration and discovery.
HASHICORP_ VAULT_ADDR	n/a	URL of the vault server
HASHICORP_ VAULT_AUTH	n/a	Provide the authentication value as Token or Userpass or Approle depending on the authentication method used.
HASHICORP_ VAULT_ NAMESPACE	n/a	The name of the namespace to be used in the application to retrieve secrets from the vault. Optional. This environment variable must be used only if namespaces is used.
HASHICORP_	The default	Optional. Path where the authentication option is

Environment Variable	Default Values	Description
AUTH_PATH	path for Userpass is userpass/ The default path for Approle is approle/	enabled in the vault.
HASHICORP_ VAULT_KV_ VERSION	2	A version of the key value engine to be used. The default value is 2 if this environment variable is not used. The only other valid value for this field is 1.
HASHICORP_ VAULT_TOKEN	n/a	Provide the token value when the authentication method is token.
HASHICORP_ VAULT_ROLE_ID	n/a	Provide the role-id if the authentication method is Approle.
HASHICORP_ VAULT_SECRET_ ID	n/a	Provide the secret-id if the authentication method is Approle.
HASHICORP_ VAULT_ USERNAME	n/a	Provide the username if the authentication method is Userpass.
HASHICORP_ VAULT_ PASSWORD	n/a	Provide the password if the authentication method is Userpass.
HASHICORP_ CACERT	n/a	Provide the path of the CA certificate used.
MASHERY_ SERVICE_CONFIG	n/a	Applications can pass TIBCO Mashery configuration information using the `MASHERY_SERVICE_CONFIG` environment variable.

Environment Variable	Default Values	Description
		The value of the environment variable is a JSON string with the required TIBCO Mashery configuration. For more information, see Integrating with TIBCO Mashery.
SPRING_CLOUD_CONFIG_SERVER_URL	n/a	Used to set URL for Spring Cloud Config Server. For example: `SPRING_CLOUD_CONFIG_SERVER_URL =http://127.1.0.1:8888/` **Note:** This must be set, if you want to use the Spring Cloud Config Server for application configuration management.
SPRING_CLOUD_CONFIG_ACCESS_TOKEN_URI	n/a	Used to set OAuth access token URL for Spring Cloud Config Server, if OAuth is enabled.
SPRING_CLOUD_CONFIG_CLIENT_ID	n/a	Used to set OAuth client ID for Spring Cloud Config Server, if OAuth is enabled.
SPRING_CLOUD_CONFIG_CLIENT_SECRET	n/a	Used to set OAuth client secret for Spring Cloud Config Server, if OAuth is enabled.
X_CONSUL_TOKEN	n/a	Used to authenticate the request made to the Consul server. This environment variable should be used when the Consul server is enabled for authentication.

Environment Variable	Default Values	Description
		Note: This should be set if authentication is enabled on the Consul server and you want to authenticate the requests made to the Consul server.
VAULT_AUTH_PATH	n/a	Path of the Authentication mounted.

Creating the TIBCO BusinessWorks Container Edition Base Docker Image for Linux Containers

Follow these steps to create a base docker image for Linux containers.

Before you begin

- Download the TIBCO BusinessWorks Container Edition runtime zip file, `bwce-runtime-<version>.zip`, from http://edelivery.tibco.com.

  To download this file,

  1. Select **Container** from the **Operating Systems** dropdown list.
  2. Read and accept the **TIBCO End User License Agreement**.
  3. Select the radio button for **Individual file Download**.
  4. Click **+** sign to view the individual components and select `bwce-runtime-<version>.zip`.
- An installation of Docker.

Note: TIBCO ActiveMatrix BusinessWorks does not ship docker folder or scripts with it, because the scripts can be pulled directly from GitHub without installing TIBCO BusinessWorks Container Edition separately.

Procedure

1. Navigate to the *TIBCO_HOME*/bwce/<*version*>/docker directory.
2. Copy the bwce-runtime-<*version*>.zip file to the *TIBCO_HOME*/bwce/<*version*>/docker/resources/bwce-runtime folder.
3. Open a command terminal and run the following command from the *TIBCO_HOME*/bwce/<*version*>/docker folder:

   ```
   docker build -t TAG-NAME .
   ```

   For example,

   ```
   docker build -t tibco/bwce:latest .
   ```

   By default, the debian:bookworm-slim image is used to create base docker image for TIBCO BusinessWorks Container Edition. The following is the Dockerfile content.

   - Dockerfile content for Debian:bookworm-slim:

     ```
     FROM debian:bookworm-slim
     LABEL maintainer="TIBCO Software Inc."
     ADD . /
     RUN chmod 755 /scripts/*.sh && apt-get update && apt-get --no-
     install-recommends -y install unzip ssh net-tools && apt-get -
     y install xsltproc && apt-get clean && rm -rf
     /var/lib/apt/lists/*
     RUN groupadd -g 2001 bwce \
     && useradd -m -d /home/bwce -r -u 2001 -g bwce bwce
     USER bwce
     ENV LANG C.UTF-8
     ENV LC_ALL C.UTF-8
     ENTRYPOINT ["/scripts/start.sh"]
     ```

   You can also change the image with the following Dockerfile content.

   - Dockerfile content for openSUSE:

     ```
     FROM opensuse/leap
     LABEL maintainer="TIBCO Software Inc."
     ADD . /
     RUN chmod 755 /scripts/*.sh && zypper -n update && zypper -n
     ```

```
refresh && \
zypper -n in unzip openssh net-tools
RUN groupadd -g 2001 bwce \
&& useradd -m -d /home/bwce -r -u 2001 -g bwce bwce
USER bwce
ENTRYPOINT ["/scripts/start.sh"]
```

- Dockerfile content for CentOS 7:

```
FROM centos:7
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 755 /scripts/*.sh && yum -y update && yum -y install
unzip ssh net-tools
ENTRYPOINT ["/scripts/start.sh"]
```

- Dockerfile content for CentOS 9:

```
FROM quay.io/centos/centos:stream9
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 755 /scripts/*.sh && yum -y update && yum -y install
unzip ssh net-tools
ENTRYPOINT ["/scripts/start.sh"]
```

- Dockerfile content for eclipse-temurin: 11-jre-alpine

```
FROM eclipse-temurin:11-jre-alpine
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 755 /scripts/*.sh && apk update && apk add unzip
openssh net-tools
RUN apk add --no-cache bash
RUN addgroup -S bwcegroup && adduser -S bwce -G bwcegroup
USER bwce
ENTRYPOINT ["/scripts/start.sh"]
```

> **Note:** To use OpenJDK11 and remove TIBCO JRE from the eclipse-temurin: 11-jre-alpine based container image, navigate to https://github.com/TIBCOSoftware/bwce-docker/tree/openjdk-alpine and clone the 'eclipse-alpine' repository to your local machine. For more information to create the base Docker image for the Linux container, see Creating the TIBCO BusinessWorks Container Edition Base Docker Image for Linux Containers.

- Dockerfile content for ubi8:

```
FROM registry.access.redhat.com/ubi8/ubi:latest
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 777 scripts/*.sh && yum install -y unzip net-tools
&& \
yum update -y; yum clean all
ENTRYPOINT ["/scripts/start.sh"]
```

- Dockerfile content for rhel7-minimal

```
FROM registry.access.redhat.com/rhel7-minimal
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 777 scripts/*.sh && microdnf install unzip net-tools
-enablerepo=rhel-7-server-rpms && \
microdnf update; microdnf clean all
ENTRYPOINT ["/scripts/start.sh"]
```

> **Note:** Ensure that you have a valid Red Hat subscription for using **rhel7-minimal**.

- Dockerfile content for RedHat Standard OS:

```
FROM registry.access.redhat.com/rhel7/rhel
LABEL maintainer="TIBCO Software Inc."
ADD . /
```

```
RUN chmod 777 scripts/*.sh && yum install -y unzip ssh net-
tools && \
yum update -y; yum clean all
ENTRYPOINT ["/scripts/start.sh"]
```

- Dockerfile content for AmazonLinux 2 OS:

```
FROM amazonlinux:2
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 777 scripts/*.sh && yum install -y unzip ssh net-
tools && \
yum update -y; yum clean all
RUN groupadd -g 2001 bwce \
&& useradd -m -d /home/bwce -r -u 2001 -g bwce bwce
USER bwce
ENV LANG C.UTF-8
ENV LC_ALL C.UTF-8
ENTRYPOINT ["/scripts/start.sh"]
```

- Dockerfile content for AmazonLinux 2023 OS:

```
FROM amazonlinux:2023
LABEL maintainer="TIBCO Software Inc."
ADD . /
RUN chmod 777 scripts/*.sh && yum update -y && yum install -y
unzip openssh-clients.x86_64 net-tools.x86_64
findutils && yum clean all
ENV LANG C.UTF-8
ENV LC_ALL C.UTF-8
ENTRYPOINT ["/scripts/start.sh"]
```

Result

The TIBCO BusinessWorks Container Edition base Docker image is now created. This base Docker image can now be used to create Docker application images.

Support for stopping the container gracefully.

To stop container gracefully in TIBCO Business Studio for BusinessWorks, run the following commands:

For Docker:

```
docker stop -t <timeout period in sec> <conainer name/ID>
```

By default, the timeout is 10 seconds. It waits for seconds to stop before killing the container forcefully.

For Kubernetes:

You can customize the grace period by setting the terminationGracePeriodSeconds at the pod spec level, which defaults to 30 seconds.

```
spec:
containers:
       name: test
       image: ...
       terminationGracePeriodSeconds: 60
```

Creating the TIBCO BusinessWorks Container Edition Base Docker Image for Windows Containers

Before you begin

- Download the TIBCO BusinessWorks Container Edition runtime zip file `bwce-runtime-windows-<version>.zip`, from http://edelivery.tibco.com.

  To download this file,

  1. Select **Container** from the **Operating Systems** dropdown list.
  2. Read and accept the **TIBCO End User License Agreement**.
  3. Select the radio button for **Individual file Download**.
  4. Click the **+** sign to view the individual components and select `bwce-runtime-windows-<version>.zip`.
- An installation of Docker.

Procedure

1. Navigate to https://github.com/TIBCOSoftware/bwce-docker/tree/windows-270-onwards and clone the `bwce-docker` repository to your local machine using the following command:

   ```
   git clone https://github.com/TIBCOSoftware/bwce-
   docker/tree/windows-270-onwards
   ```

2. Copy the `bwce-runtime-windows-<version>.zip` file to the `resources/bwce-runtime` folder.
3. Navigate to the cloned directory and open a command terminal and run the following command:

   ```
   docker build -t TAG-NAME .
   ```

   For example,

   ```
   docker build -t tibco/bwce:latest .
   ```

   By default, the `microsoft:nanoserver` image is used to create base Docker image for TIBCO BusinessWorks Container Edition. You can also change the image with the following Dockerfile content.

   - Dockerfile content for Windows:

     ```
     FROM microsoft/nanoserver
     MAINTAINER TIBCO Software Inc.
     ADD . /
     CMD ["powershell", "c:/scripts/start.ps1"]
     ```

   Note: Currently, TIBCO BusinessWorks Container Edition Windows Container supports `microsoft/nanoserver` as the base OS.

Result

The TIBCO BusinessWorks Container Edition base Docker image is now created. This base Docker image can now be used to create Docker application images.

Creating the TIBCO BusinessWorks Container Edition Docker Image for Rendezvous Palette

Follow these steps to create a docker image for TIBCO BusinessWorks Container Edition applications designed using the Rendezvous palette.

Procedure

1. Navigate to the *TIBCO_HOME*/bwce/<*version*>/docker directory.
2. Open a command terminal and execute the following command from the *TIBCO_HOME*/bwce/<*version*>/docker folder:

   ```
   docker build -t TAG-NAME .
   ```

3. Create an EAR file for the TIBCO BusinessWorks Container Edition application and keep the EAR file in a folder.
4. Download the TIB_rv_<version>_linux_x86_64.zip file from the http://reldist.na.tibco.com/package/rv/ reldist location. Unzip the TIB_rv_<version>_linux_x86_64.zip file and add the unzipped file in the same folder where the EAR file is kept.
5. Use the Dockerfile to create an application image. For more information, see Building an Application Image.

Result

The application image is ready to run with on-premises Rendezvous.

Extending the Base Docker Image

The TIBCO BusinessWorks Container Edition base Docker image must be uploaded to your Docker environment before deploying any applications.

The base Docker image extends supports to the following:

- Third-party drivers
- OSGI bundles

- Integration with application configuration management systems
- Application certificate management

Before you begin

Create the base Docker image for the Linux or Windows container following steps in Creating the BWCE Base Docker Image for Linux Containers or Creating the BWCE Base Docker Image for Windows Container.

The base Docker image can be extended in the following ways:

- **Provision supported JDBC drivers**
  1. Run `bwinstall[.exe]` help from `<BWCE_HOME>/bin` and follow instructions to add the driver to your TIBCO BusinessWorks Container Edition installation.
  2. Copy all the contents from `<TIBCO_HOME>/bwce/version/config/drivers/shells/<driverspecific runtime>/runtime/plugins/` to a temporary folder.
  3. From the temporary folder, use the Dockerfile given below to copy these jars into the base Docker image:

     ```
     FROM tibco/bwce:latest
     COPY . /resources/addons/jars
     ```

- **Provision the OSGi bundle jars**
  1. Copy the OSGified bundle jars into a temporary folder.
  2. From the temporary folder, use the Dockerfile given below to copy these jars into the base Docker image:

     ```
     FROM tibco/bwce:latest
     COPY . /resources/addons/jars
     ```

- **Application Configuration Management**

  For more information, see Using Configurations from Configuration Management Services.

- **Certificate Management**

  Certificates are used by applications to connect to different systems. For example, a certificate to connect to the TIBCO Enterprise Message Service.

Bundling certificates with your application is not a good idea as you would need to rebuild your application when the certificates expire. To avoid that, copy your certificates into a temporary folder. From the temporary folder, use the Dockerfile given below to copy these certificates into the base Docker image:

```
FROM tibco/bwce:latest
COPY . /resources/addons/certs
```

Once the certificates expire, you can copy the new certificates into the base Docker image without rebuilding your application. Deploy your application with the base Docker image. To access the certificates folder from your application, use the environment variable `BW_KEYSTORE_PATH`. For example, `#BW_KEYSTORE_PATH#/mycert.jks` in your application property.

- **Provision the TIBCO BusinessWorks™ Container Edition Plug-in Runtime**

  To add TIBCO certified plug-ins:

  1. Download the appropriate Plug-in packaging. For example, the TIBCO ActiveMatrix BusinessWorks™ Plug-in for WebSphere MQ, from https://edelivery.tibco.com.
  2. Locate the plug-in zip file, `<product id>_ePaas.zip` or `TIB_<version>_<build number>_bwce-runtime.zip` from the downloaded artifacts and copy into a temporary folder. From the temporary folder, use the Dockerfile given below to copy these plug-in runtime zip files into the base Docker image:

     ```
     FROM tibco/bwce:latest
     COPY *.zip /resources/addons/plugins
     ```

  To add plug-ins created using the TIBCO ActiveMatrix BusinessWorks™ Plug-in Development Kit to runtime into your base Docker image:

  1. Install the plug-in if it is not installed.
  2. Navigate to the *TIBCO_HOME*/bwce/palettes/`<plug in name>`/`<plugin version>` directory and zip the `lib` and `runtime` folders into `<plugin name>.zip`.
  3. Copy `<plugin name>.zip` to a temporary folder.
  4. From the temporary folder, use the Dockerfile given below to copy these plug-in runtime zip files into the base Docker image:

```
FROM tibco/bwce:latest
COPY *.zip /resources/addons/plugins
```

- **Provision the third-party installations to runtime**

  To add the third-party installation to the Docker image:

  1. Package third-party client installation into zip.
  2. Copy the zip to a temporary folder.
  3. From the temporary folder, use the Dockerfile given below to copy the archive files into the base Docker image:

     ```
     FROM tibco/bwce:latest
     COPY *.zip /resources/addons/thirdparty-installs
     ```

  4. Build the Docker image:

     ```
     docker build -t YOUR-TAG.
     ```

- **Provision to add custom JDBC driver**

  1. Follow the steps outlined in the **Enabling Custom Drivers** section of the **JDBC Connection** topic in the *Bindings and Palette Reference* guide. After the project has been exported as a plug-in, locate the JAR in the plug-ins folder and copy it to a temporary folder.
  2. From the temporary folder, use the Dockerfile given below to copy these jars into the base Docker image:

     ```
     FROM tibco/bwce:latest
     COPY . /resources/addons/jars
     ```

- **Provision to add custom JMS driver**

  1. Follow the steps outlined in the **Enabling Custom Drivers** section of the **JNDI Connection** topic in the *Bindings and Palette Reference* guide. After the project has been exported as a plug-in, locate the JAR in the plug-ins folder and copy it to a temporary folder.
  2. From the temporary folder, use the Dockerfile given below to copy these jars

into the base Docker image:

```
FROM tibco/bwce:latest
COPY . /resources/addons/jars
```

- **Provision to use custom logs**
  1. Create a folder `custom-logback` in the `resources/addons` folder.
  2. Add the customized Logback file in the folder. The name of the Logback file should be `logback.xml`.

     Note: While running the application, set the environment variable CUSTOM_LOGBACK="true".

What to do next

Open a command terminal from the temporary folder where you created the Dockerfile and run:

```
docker build -t TAG-NAME .
```

For example,

```
docker build -t tibco/bwce-oracledriver:latest .
```

Provisioning Custom Profile

1. Create and export an application profile as a substvar file. For more information on how to export an application profile see, Exporting an Application Profile. While exporting the application profile, select the **Include System Properties** checkbox.
2. Copy the substvar file to the base Docker image:

   ```
   FROM tibco/bwce:latest
   COPY *.substvar /
   ```

3. Build the Docker image:

```
docker build -t YOUR-TAG.
```

4. Set the *BW_PROFILE* environment variable and run the application providing the substvar profile name.

Building an Application Image

Follow these steps to build an application image in TIBCO BusinessWorks Container Edition.

Procedure

1. Copy the Dockerfile from the samples directory to the location where you placed the EAR file.
2. From the terminal, navigate to the folder where the EAR and Dockerfile are stored.
3. In the Dockerfile, make sure the base image points to the TIBCO BusinessWorks Container Edition runtime base image.

   Also make sure the ear file path and name is correct.

   ```
   FROM baseimage
   MAINTAINER Tibco
   ADD appname.ear /
   EXPOSE 8080
   ```

4. On the terminal run the following command to generate the application image.

   ```
   docker build -t appname .
   ```

5. If the TIBCO BusinessWorks Container Edition applications are designed using Rendezvous, use the following dockerfile:

   ```
   FROM baseimage
   COPY <<EAR NAME>> /
   COPY tibrv /tibrv
   RUN echo $PATH
   ENV PATH=$PATH:/tibrv/<rv_version>/bin
   ENV PATH=$PATH:/tibrv/<rv_version>/lib
   ENV PATH=$PATH:/tibrv/<rv_version>/lib/tibrvnative.jar
   ```

```
ENV LD_LIBRARY_PATH=/tibrv/<rv_version>/lib
```

Integrating with TIBCO Mashery

You integrate TIBCO BusinessWorks Container Edition applications with TIBCO Mashery using an environment variable.

- TIBCO Mashery configuration can be passed using the `MASHERY_SERVICE_CONFIG` environment variable. The value of this environment variable is a JSON string.

  ```
  MASHERY_SERVICE_CONFIG: '{"username":"xxx@tibco.com","password":
  "xxxx",
  "clientId":"xxxxxxxx","clientSecret":"xxxxxx","masheryApiServerUr
  i": "https://api.mashery.com","areaUuid": "xxxxxxxxx-xxxx-xxxx-
  xxxx-
  xxxxxxxxx","trafficManagerDomain":"xxxxxxxx.api.mashery.com","host"
  :"10.98.200.235","ports":[{"port":"8080","mappedPort":"32764"},
  {"port":"8090","mappedPort":"32763"}]}'
  ```

  For the JSON string:

  - Host field contains the IP (10.98.200.235) of the machine where the container is deployed.
  - Ports field is a key-value pair array, where port signifies the exposed container port (8080) and mappedPort (32764) signifies the mapped machine port.

  The TIBCO BusinessWorks Container Edition runtime looks for an environment variable for the TIBCO Mashery configuration. If found, that configuration is used to register HTTP based endpoints (HTTP, REST, or SOAP/HTTP).

Running container on Docker based platform as non-root user

You can built and deploy the TIBCO BusinessWorks Container Edition application on Docker based platform as a non-root user.

Note: The BWCE applications use non-root user for running BWCE containers. The non-root users are now the default users.

Before you begin

Ensure that you have set up Docker based platform to build and deploy an application.

Procedure

1. Relax the security at OpenShift level by granting all the authenticated users access to the **anyuid** SCC (Security Context Constraints).

   ```
   oc adm policy add-scc-to-group anyuid system:authenticated
   ```

2. Create a user in the Dockerfile with a known user ids (UID) and group ids (GID), and run the container by using the same user. When adding UID or GID to Dockerfile, ensure that the UID or GID is not reserved and used in the base image.

   Note: While granting **anyuid** SCC, ensure that the USER is defined in the Dockerfile. Otherwise, Openshift allows images to run as root user.

Improving container startup time for Docker

When creating the base docker image, the runtime `.zip` file is extracted before the BWEngine starts, causing a delay in the container startup time.

To avoid the delay, use the modified Dockerfile `setup.sh`, and the `start.sh` scripts to extract the TIBCO BusinessWorks Container Edition runtime `.zip` while creating the base image.

The `reducedStartupTime` folder contains the modified Dockerfile, the `setup.sh` script, and the `start.sh` script.

Replace the Dockerfile from `TIBCO_HOME/bwce/<version>/docker` and both the `setup.sh` and `start.sh` scripts from `TIBCO_HOME/bwce/<version>/docker/scripts` with the Dockerfile and the `setup.sh` and `start.sh` scripts placed inside the `reducedStartupTime` folder.

Remote Debugging

You can debug an application running on a remote container through TIBCO Business Studio for BusinessWorks.

Procedure

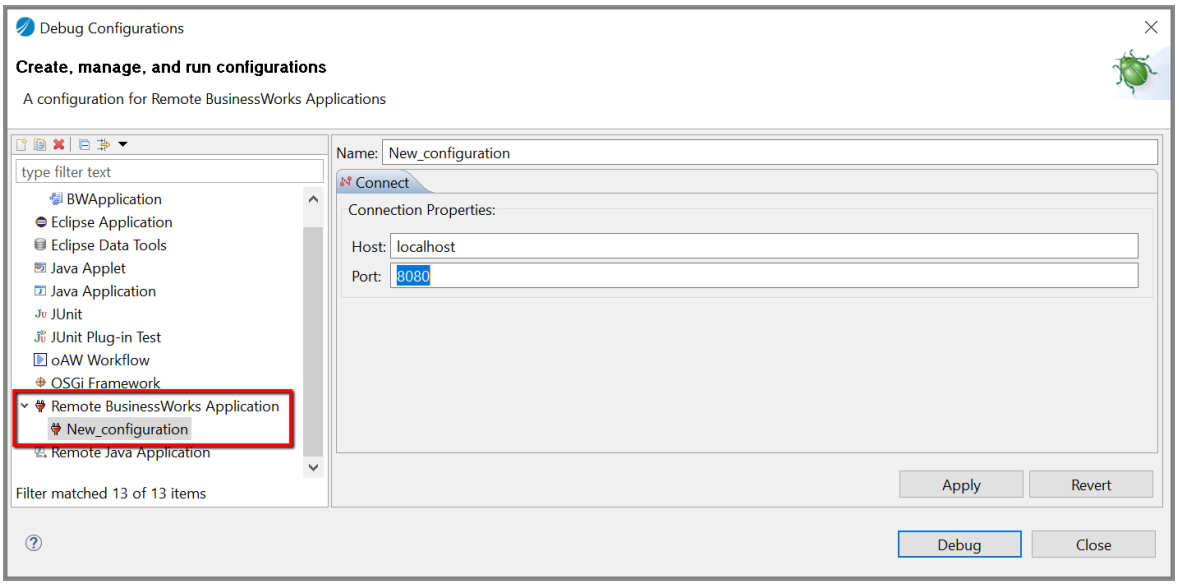
1. Expose ports `8090` and `5005` in the Docker file, while building an application image.
2. Map ports `8090` and `5005` using `-P` command while deploying the application.
3. Once the application starts successfully, enable remote debugging by invoking the following engine REST API on port `8090`:

   ```
   http://localhost:<mappedPortNo>/bw/bwengine.json/debug/?interface=0
   .0.0.0&port=5005&engineName=Main
   ```

   The HTTP Verb is POST and the URL can be accessed by API Clients such as Postman, Swagger UI and so on.

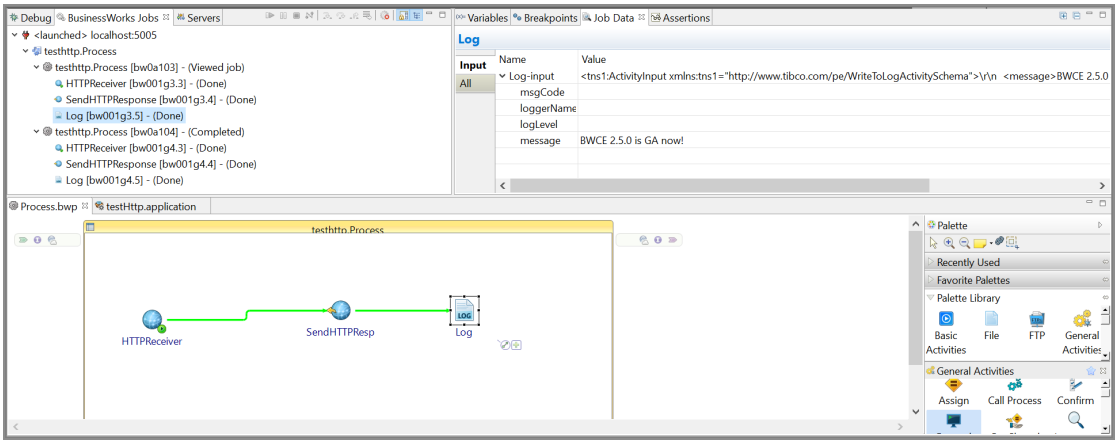
   > **Note:** The `<mappedPortNo>` is the port which is mapped to `8090` while deploying the application.

4. Import the application running in the container to TIBCO Business Studio for BusinessWorks that you want to remote debug.
5. In TIBCO Business Studio for BusinessWorks , create a Remote Debug launch configuration.
   a. Choose **Run > Debug Configurations**.
   b. In the Debug Configuration dialog, choose **Remote BusinessWorks Application > New_configuration**. Enter the following information:
      - **Name**: The name of the configuration.
      - **Host**: The name of the docker host.
      - **Port**: The remote debug port which is mapped to port `5005` while deploying the application.



6. Launch the application using the Remote Debug launch configuration.

   The application is launched in the debugger. Job data is displayed in the Debug window with the process diagram.



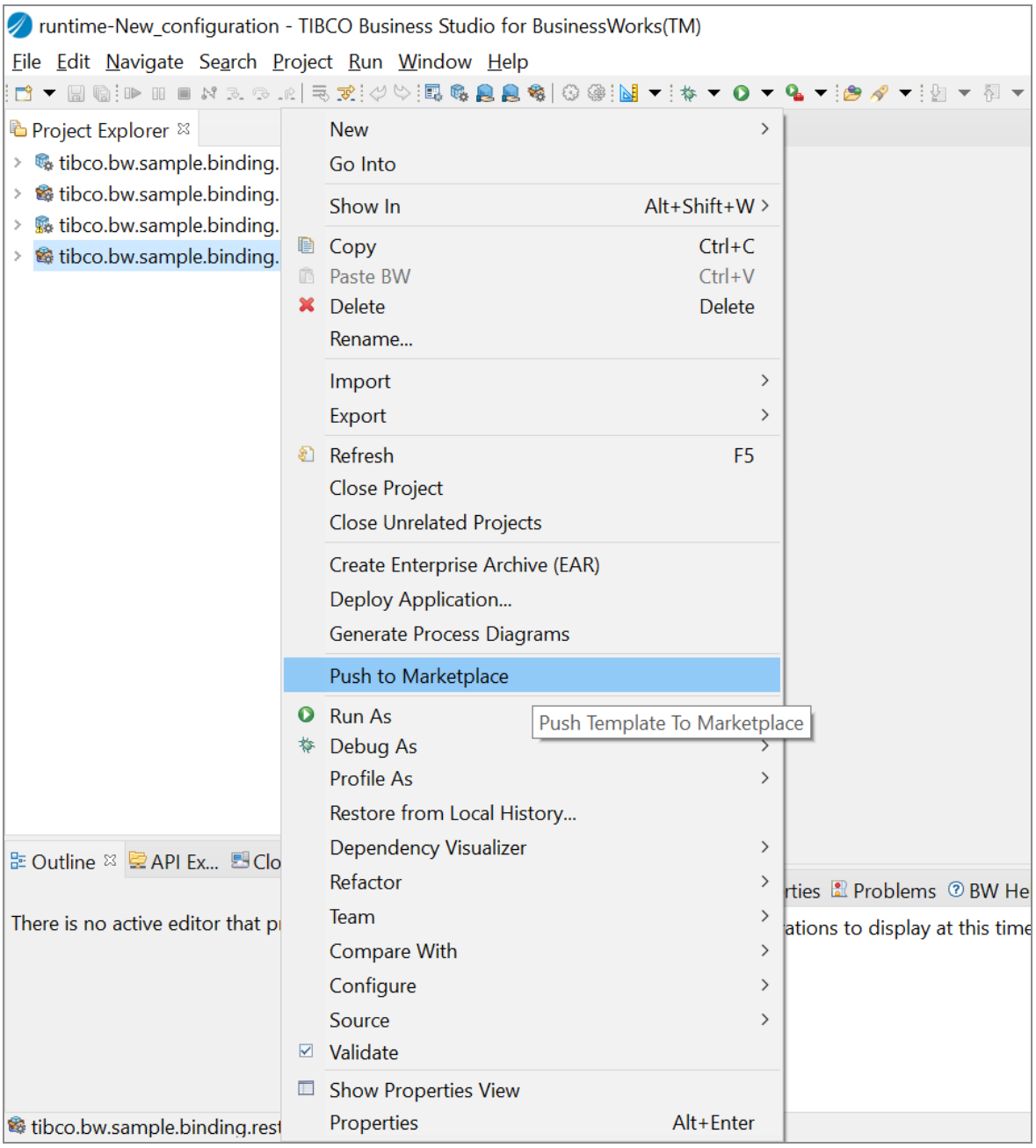
Pushing Application Templates to TIBCO Cloud™ Integration Marketplace

The Pushing Application Templates to Marketplace feature is an integration of the Application Template with the TIBCO Cloud Integration Marketplace via TIBCO Business Studio for BusinessWorks. You can use the feature to publish the app templates as marketplace listings on the TIBCO Cloud Integration Marketplace. You can push your integration projects as a marketplace listing by using the **Push to Marketplace** functionality, which other users from the organization can download and create their own integration projects using the **Import Marketplace** listing functionality. For more information about other marketplace listing-related actions, see TIBCO Cloud™ Integration Marketplace.

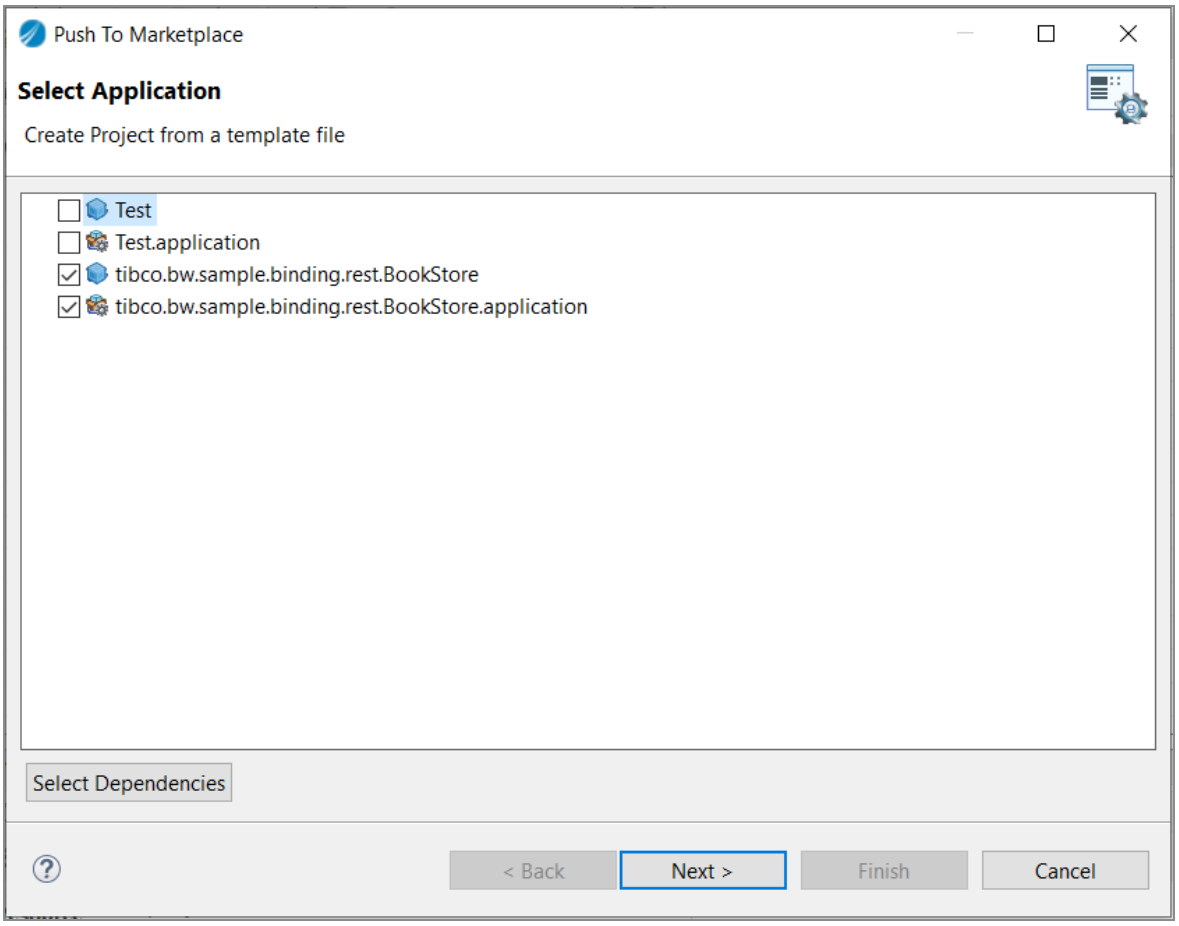
Perform the following steps to push an app template to the Marketplace:

Procedure

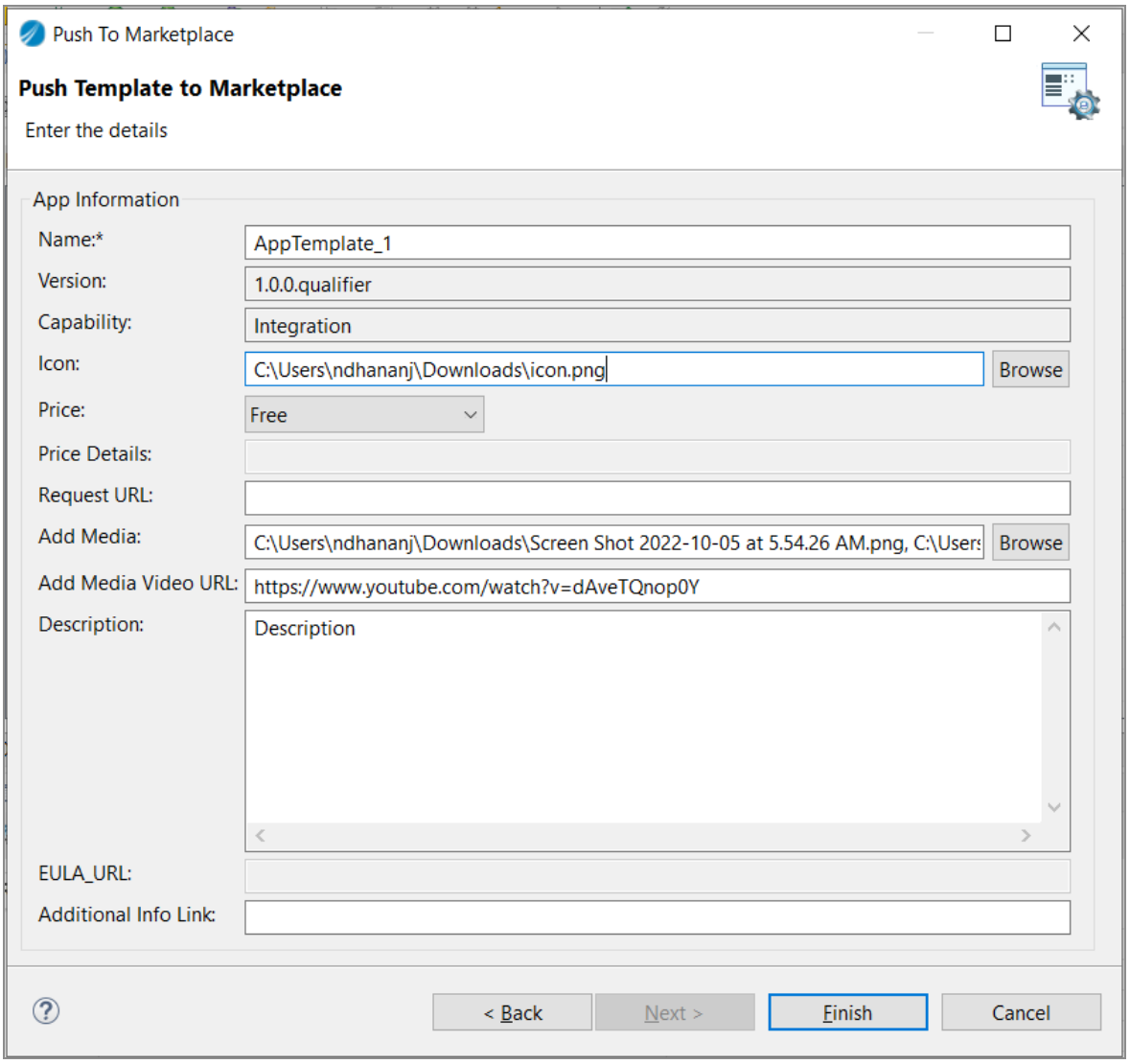
1. Right-click a Project in the Project Explorer window and select **Push to Marketplace**.



2. The **Push to Marketplace** dialog opens. Select the application module and its dependencies to be pushed onto the Marketplace and click **Next**.



3. Enter the appropriate details and click **Finish**.



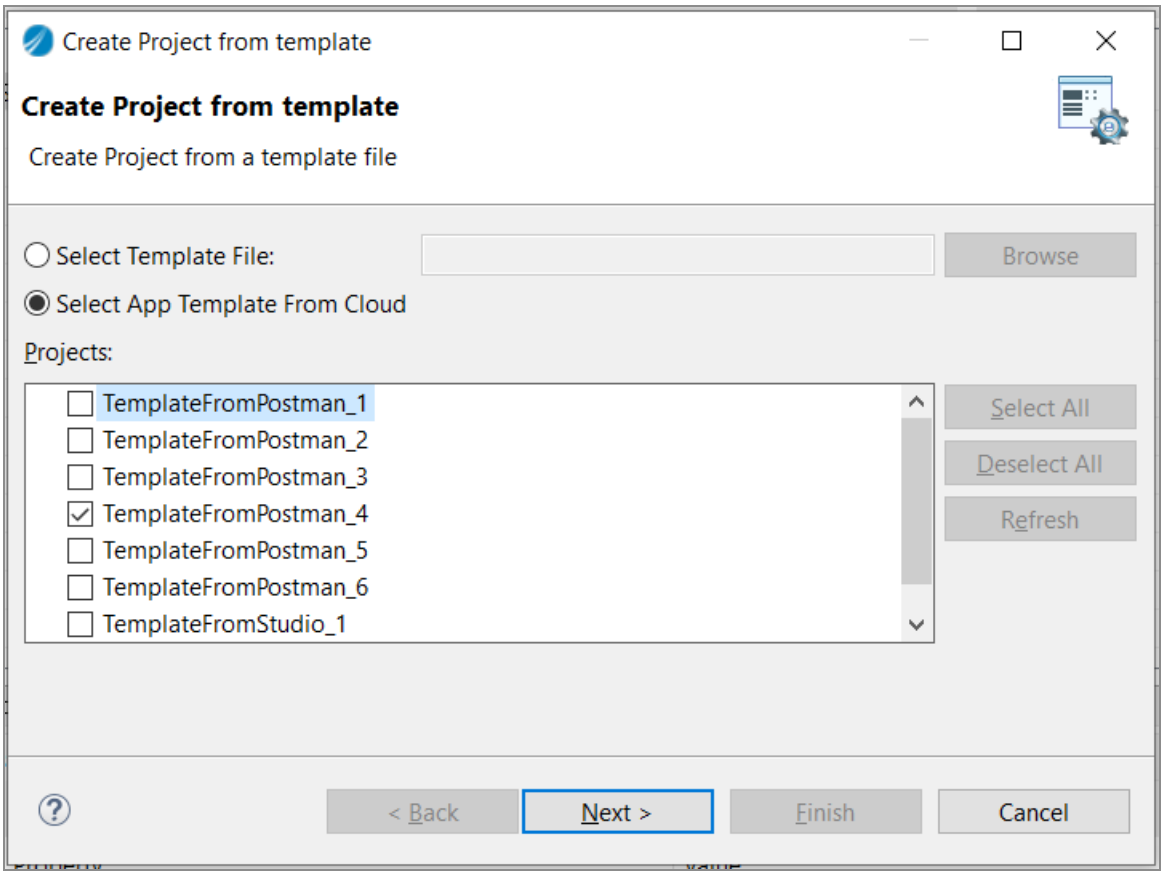
The app templates are pushed to the TIBCO Cloud Integration Marketplace.

For more information, see TIBCO Cloud™ Integration Marketplace.

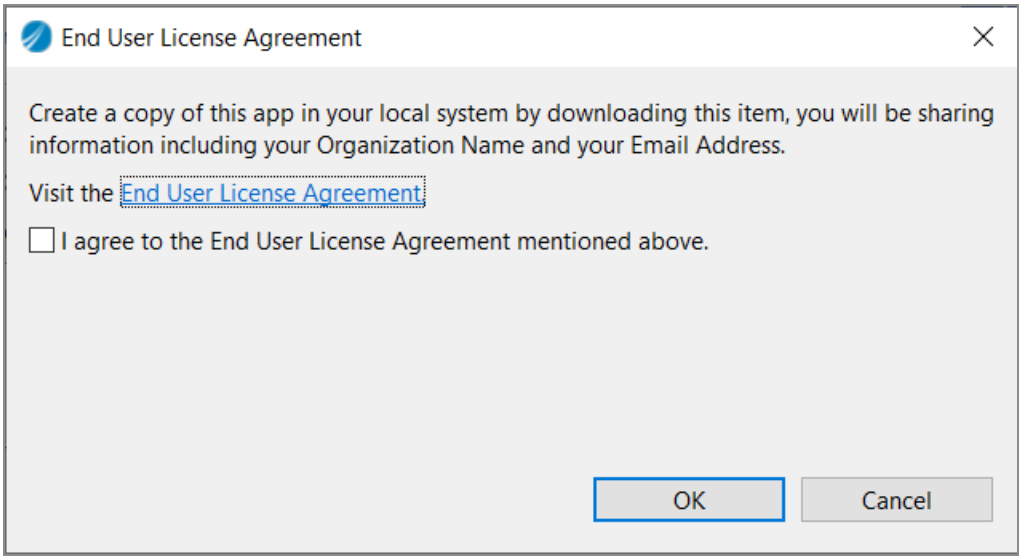
Perform the following steps to import an app template from the Marketplace:

Procedure

1. Right-click the Project Explorer window and go to **New > BusinessWorks Project from Template**.
2. The **Create Project from Template** dialog opens. Select **Select App Template from Cloud** and click **Next**.



3. Accept the End User License Agreement by selecting the **I agree to the End User License Agreement mentioned above** checkbox.



4. Click **OK**.

   The template gets downloaded. Then follow the process similar to Creating a Project from a Template.

Note: You can also download the app template from TIBCO Cloud Integration UI and import it into TIBCO Business Studio for BusinessWorks.

Connecting to TIBCO Cloud

You can use this feature to connect to TIBCO Cloud in the TIBCO Cloud Integration Environment using TIBCO Business Studio client. You can use the SSO (Single Sign-on) method to connect to the cloud.

SSO (Single Sign-on)

The single sign-on feature is provided from TIBCO Business Studio for BusinessWorks 2.7.2 onwards. The security folder under `BW_HOME` contains two configuration files, *software_statement.properties* and *ssoConfig.properties*. The *software_statement.properties* file contains the software statement for SSO client registration, whereas the *ssoConfig.properties* file contains the configuration details for SSO authentication.

> **Note:** If you want to use the SSO feature, you must first log in with the SSO option each time on restarting and launching TIBCO Business Studio for BusinessWorks.

To log in to the cloud, click the **SSO Login** icon or click the SSO login hyperlink (Login with SSO...) on the API Explorer tab.

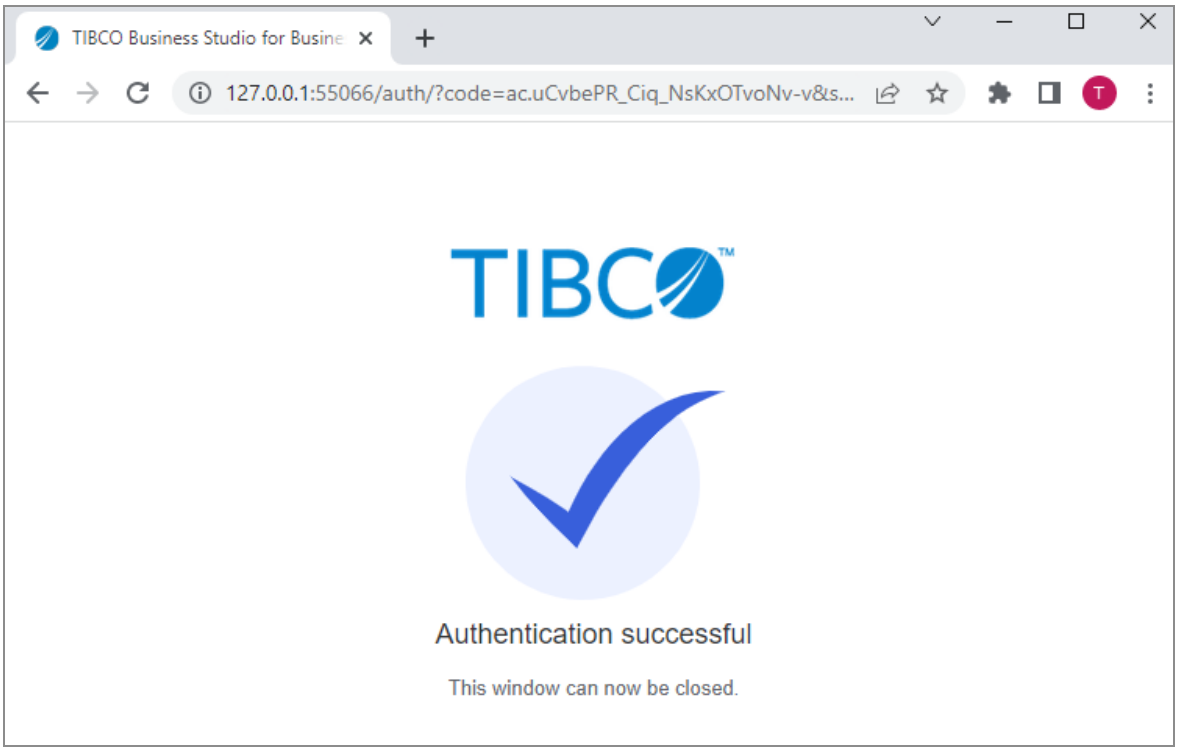
Important: You must update the configuration details (`auth_url`, `tci_hostname`, `tci_portnumber`, and `redirected_uris`) in the `ssoConfig.properties` file as required before starting TIBCO Business Studio for BusinessWorks. When you start TIBCO Business Studio for BusinessWorks, the `clientDetails.properties` file is automatically generated and it contains the clientID. This is a one-time process and later the existing file is used.

- auth_url is the URL, where you want to register your client. The default value is *auth_url=https://account.cloud.tibco.com:default*.
- tci_hostname is used to get the list of available APIs and cloud applications in the API Explorer and Cloud Applications tabs respectively. The default value is *tci_hostname=https://integration.cloud.tibco.com:default*.
- tci_portnumber is the port number. The default value is *tci_portnumber=443*.

Before you begin

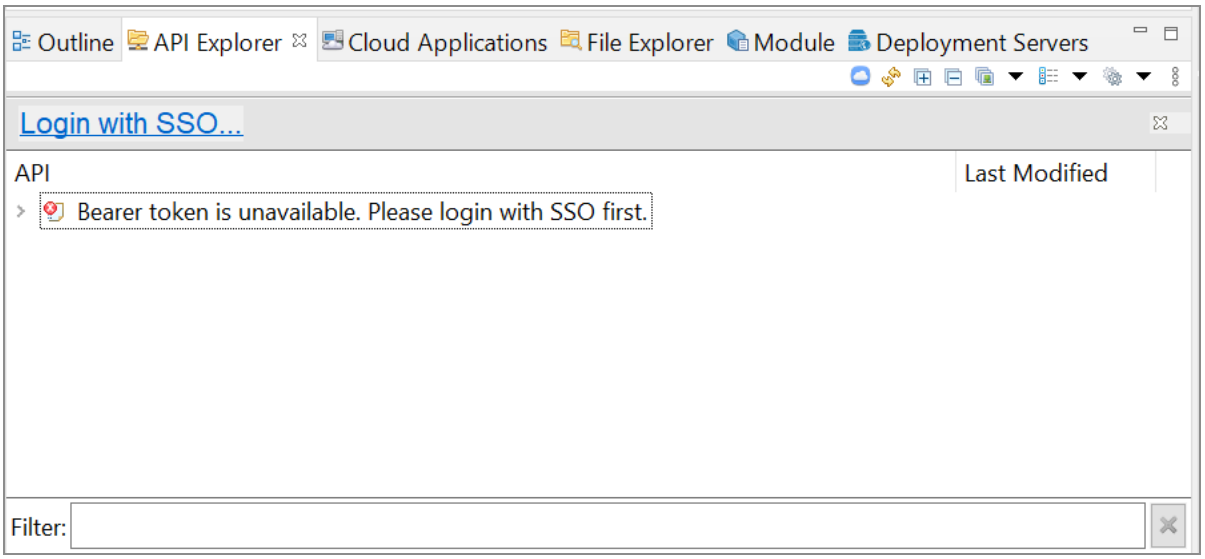
Update the configuration details in the *ssoConfig.properties* file as required before starting TIBCO Business Studio for BusinessWorks.

1. Start TIBCO Business Studio for BusinessWorks. The *clientDetails.properties* file is generated and the client is registered.
2. Click the **SSO Login** icon or hyperlink (Login with SSO...) on the API Explorer.
3. Enter the user name and password for cloud authentication in the Username and Password fields respectively. After entering the credentials, you are redirected to the TIBCO BusinessWorks Container Edition login page in the browser. On successful login, you can see the **Authentication successful** page.

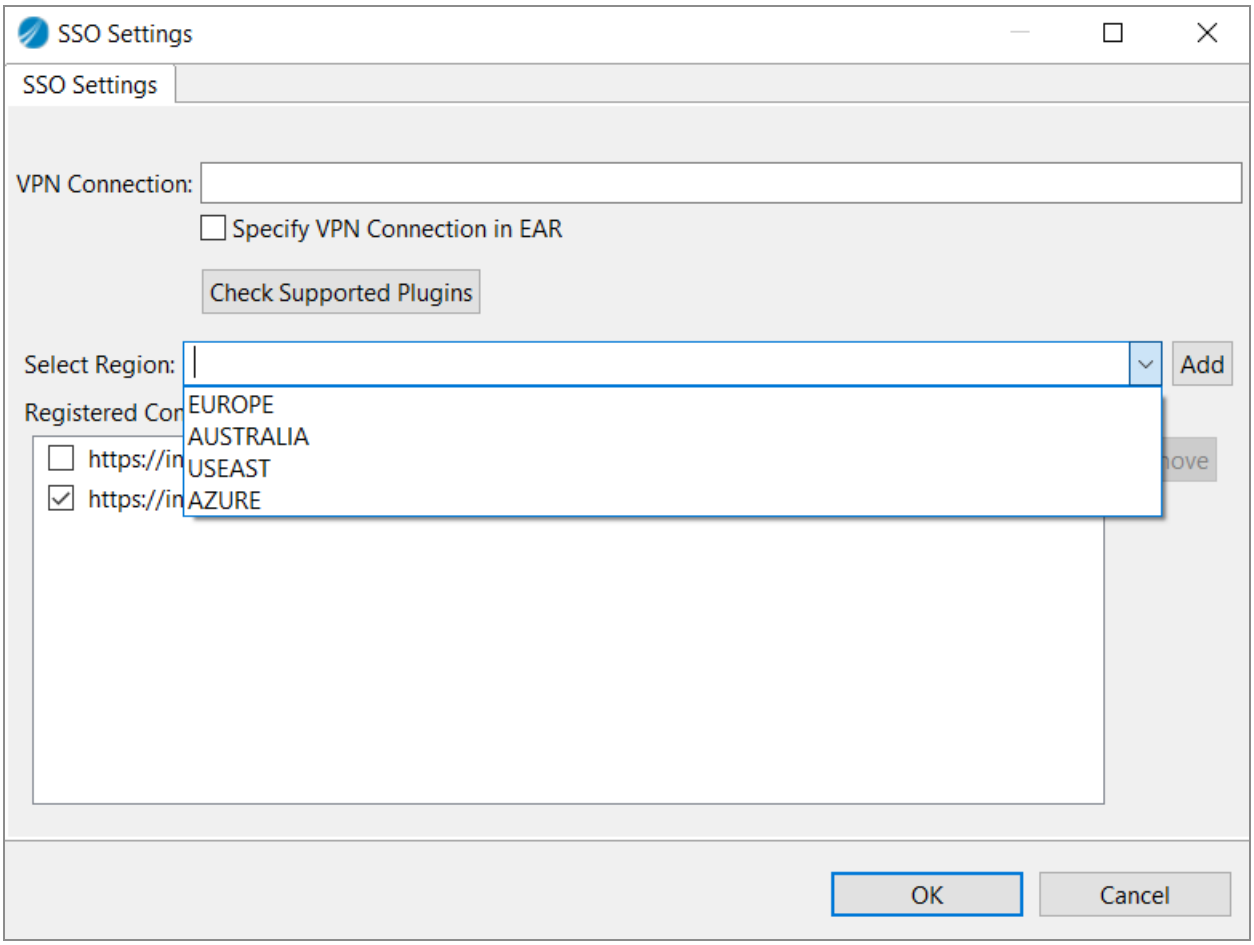


The final bearer token is generated and the API Explorer and Cloud Applications tabs in the TIBCO Business Studio for BusinessWorks get refreshed to show the updated list of APIs and cloud applications.

You can specify the VPN connection in EAR by configuring the VPN details and also select and register the region you want to connect to in the SSO Settings dialog.



SSO Settings Dialog

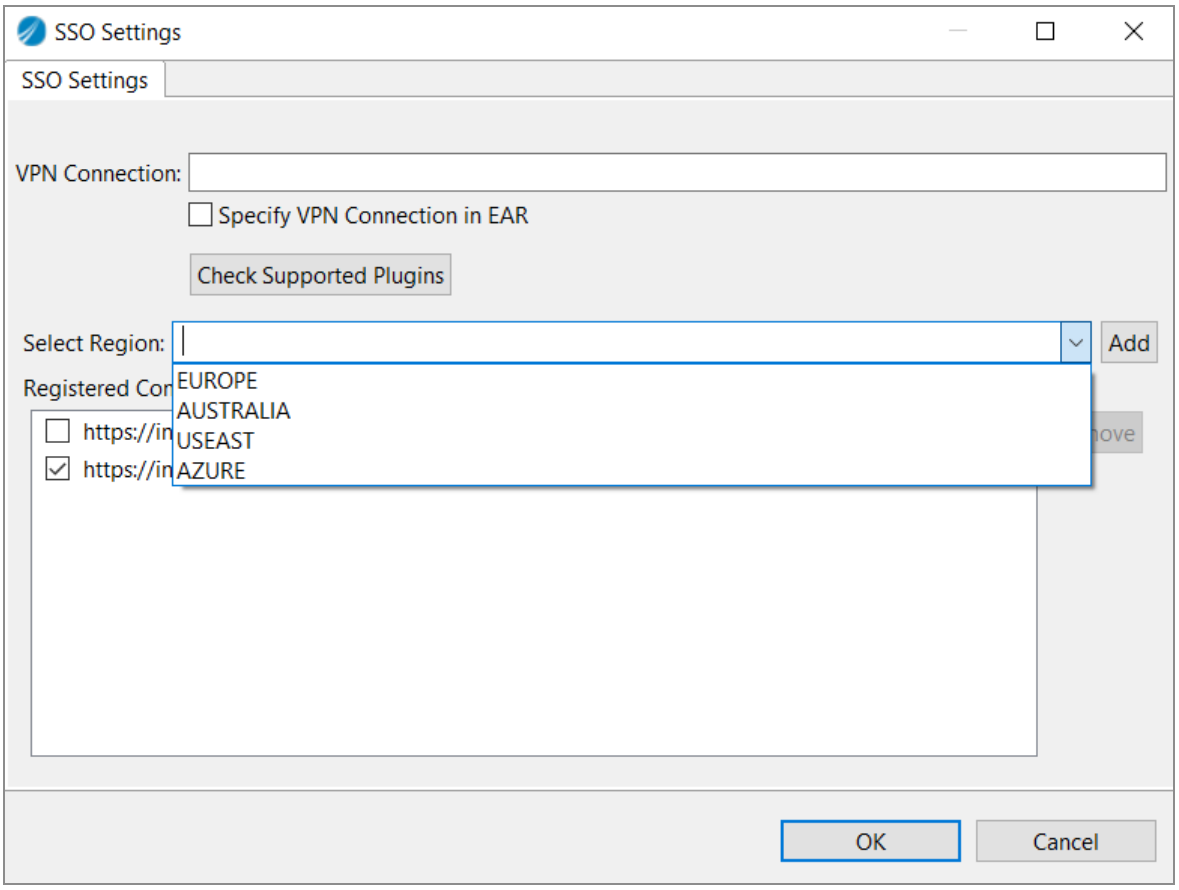


The **Check Supported Plugins** button is used to check and synchronize all the supported plugins on the cloud.

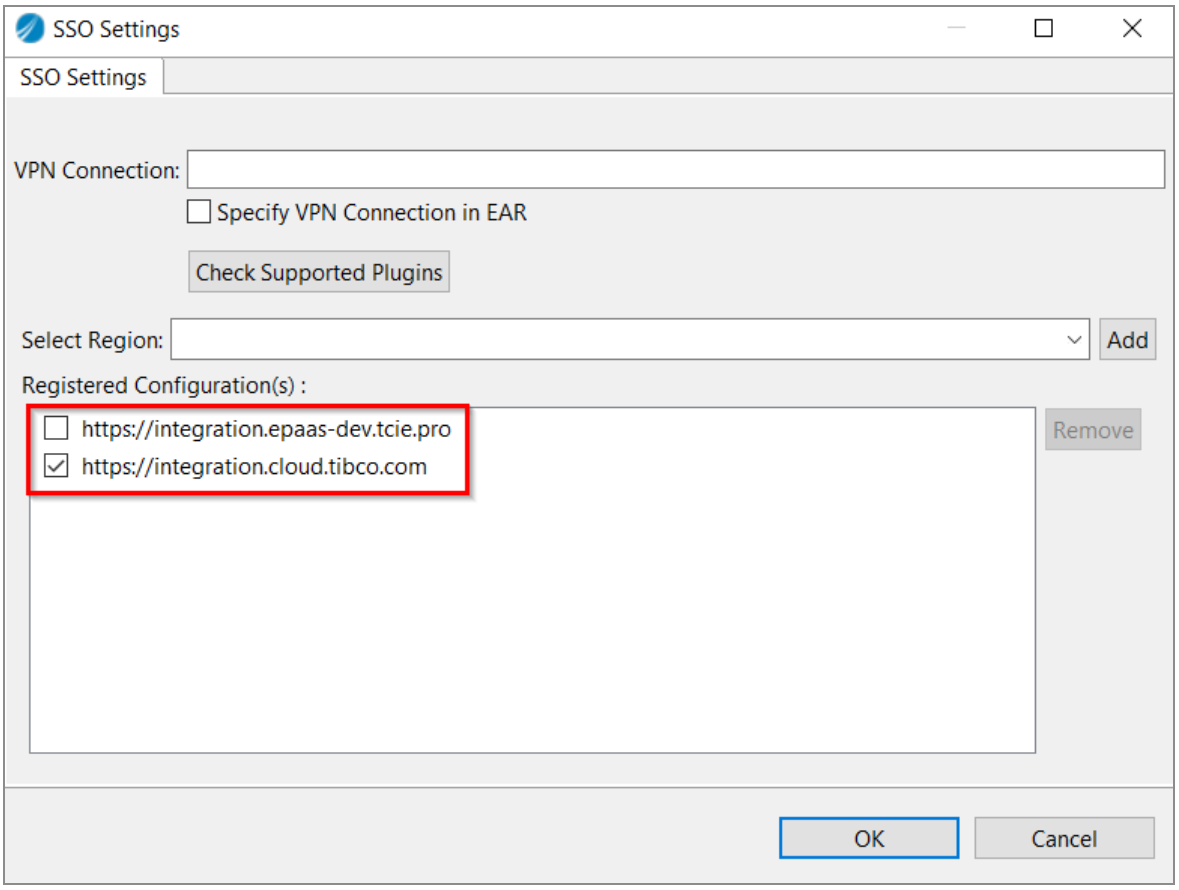
Perform these steps to register a region for SSO login:

Procedure

1. Open the **SSO Settings** dialog. The default US production (https://integration.cloud.tibco.com) is selected.
2. Select the TIBCO Cloud Integration region you want to register from the dropdown list and click **Add**.



3. Select the **Registered Configuration(s)** checkbox and click **OK**. The selected region is now registered and your current cloud connection is reset.



4. Click **Login to SSO...** to connect to the registered region.

Consuming a Service Deployed in TIBCO Cloud Mesh

This feature describes how to use **TIBCO Cloud Mesh** in an application in the **TIBCO Cloud Integration Environment** and **TIBCO Business Studio** client.
TIBCO Cloud Mesh allows you to discover any private REST endpoint exposed within TIBCO Cloud™ domains, in your organization or related organizations.

Authentication and authorization for these private endpoints is provided automatically. You browse available services and select one, rather than copying and pasting a URL. For more information on **TIBCO Cloud Mesh**, see the TIBCO Cloud Mesh topic in the TIBCO Subscriber Cloud documentation.

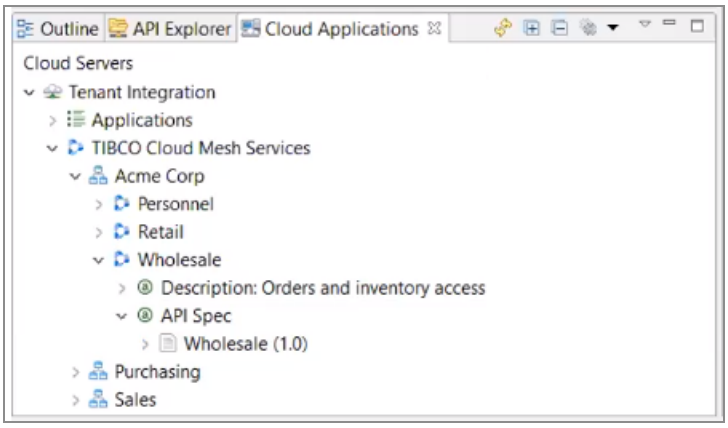
Before you begin

Note: Before you can use the TIBCO Cloud™ Mesh service, you must connect to TIBCO Cloud™. For information, see Connecting to TIBCO Cloud.

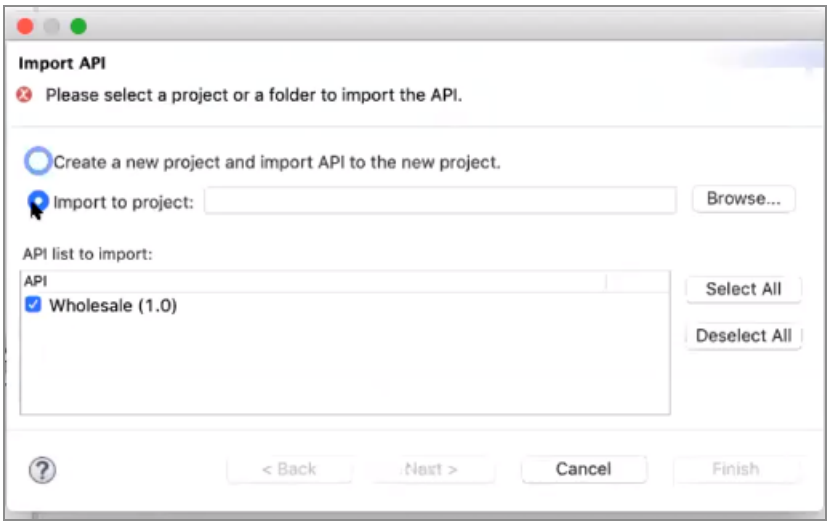
To use this feature, perform the following steps:

Procedure

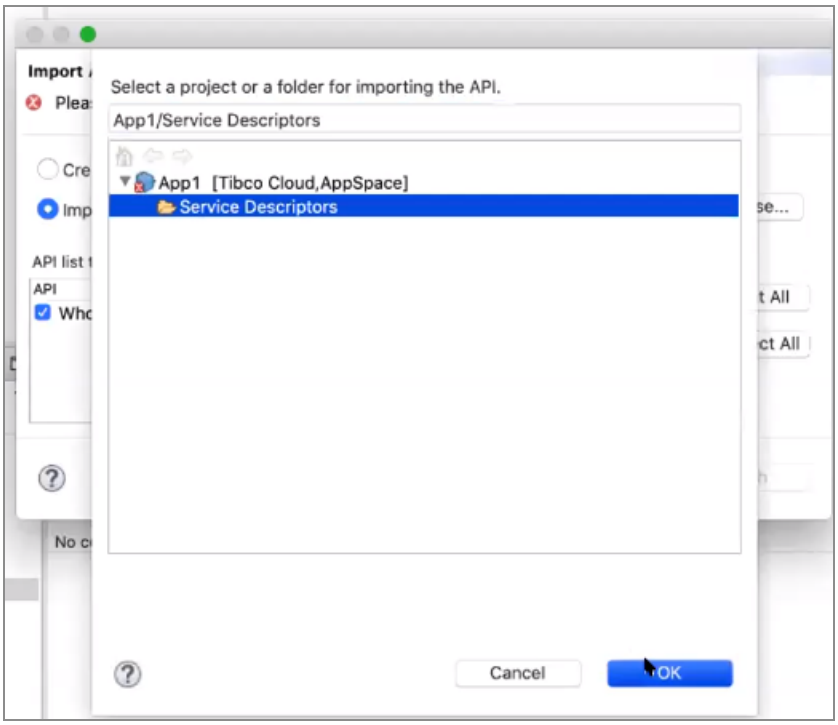
1. Open the **Cloud Applications** view. (See note above if you have not yet connected to TIBCO Cloud™.)
2. Under the **Applications** section, find the **TIBCO Cloud Mesh Services** section.
3. Expand the section to view the services available to you. If your organization has related (parent or child) organizations, the screen displays all the related organizations as folders containing the services. Applications with Private endpoints (required to display the applications) are listed in the **TIBCO Cloud Mesh Services** in the **Cloud Servers** view.



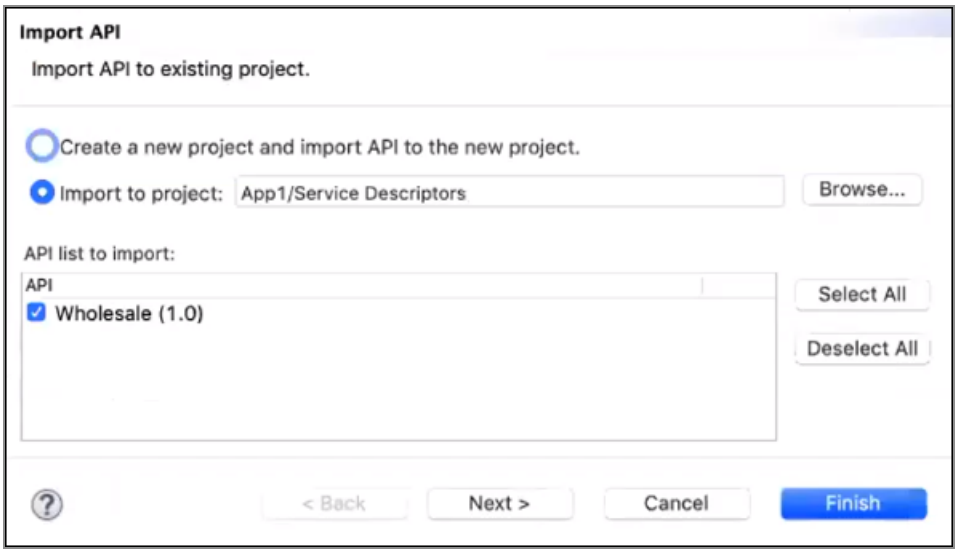
4. Right-click the mesh service and select **Import**.



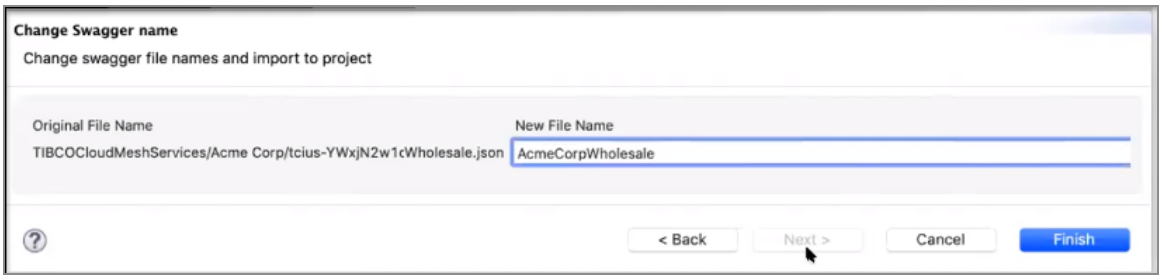
5. In the **Import API** dialog, select **Import to project**.
6. Click **Browse**.



7. Select a folder or project and click **OK**.

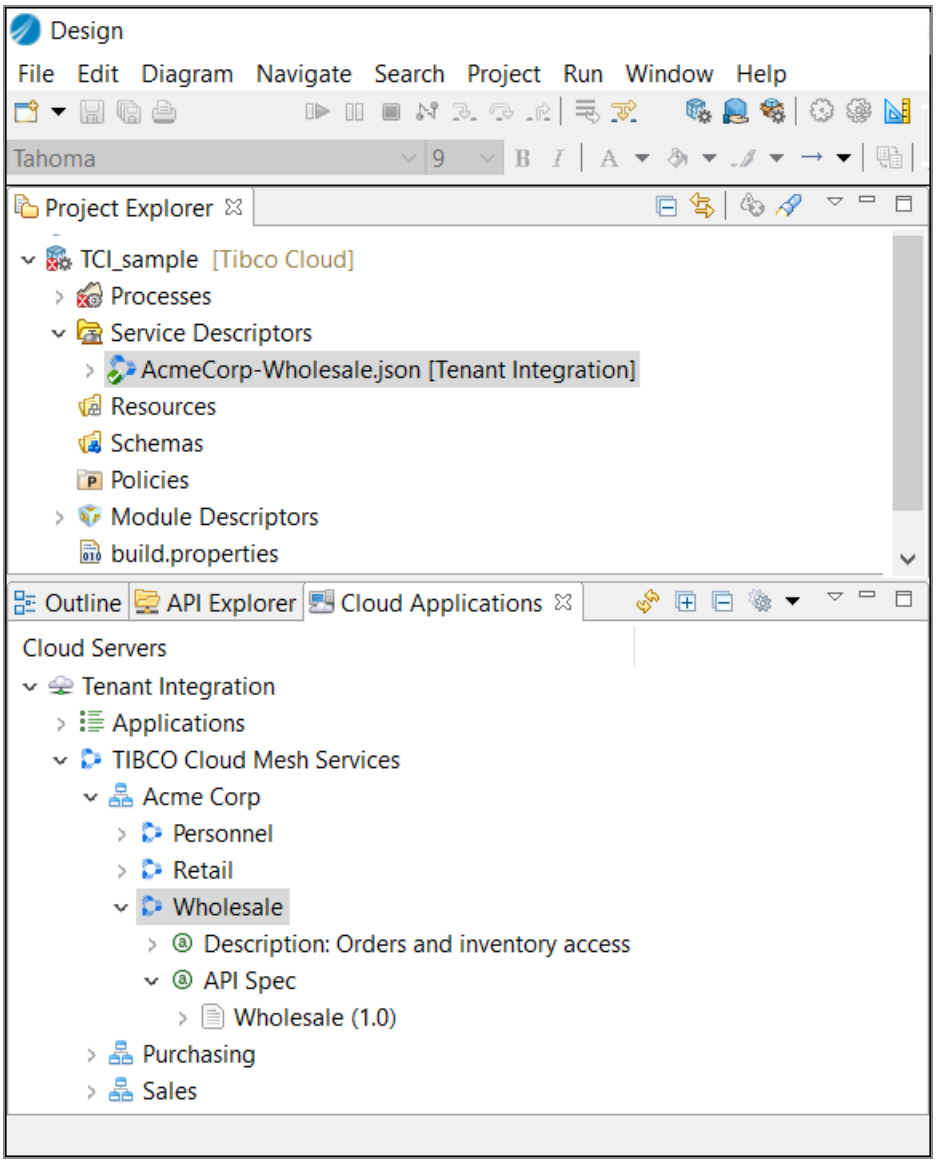


8. In the Import API screen, do **not** click **Finish**. Instead, click **Next**.



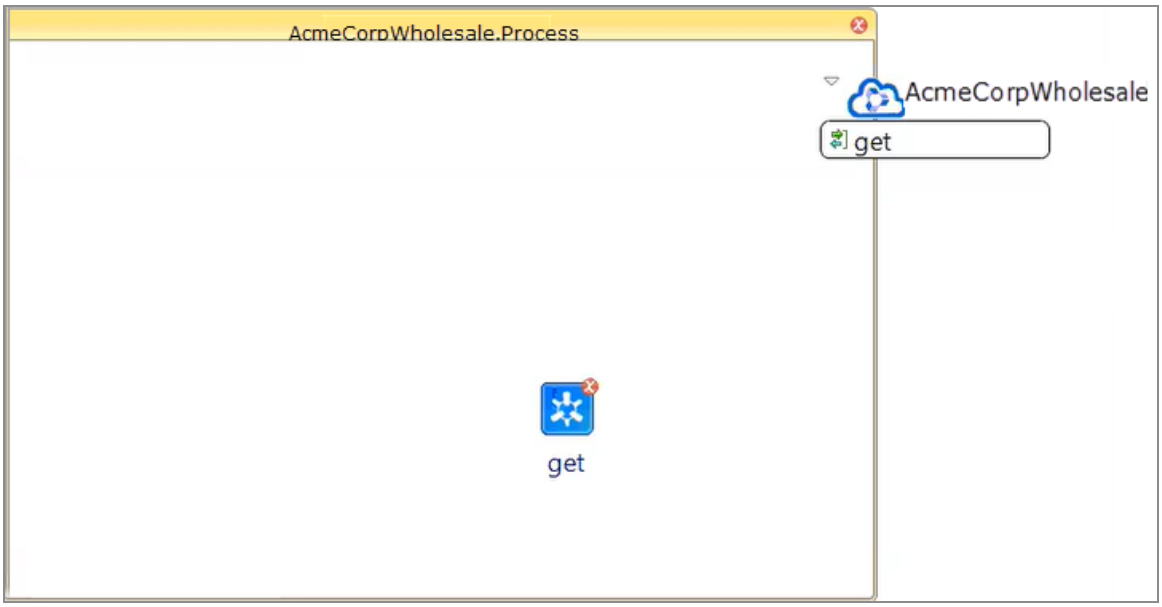
9. On the **Change Swagger name** screen, enter a logical, unique name for the Swagger file. The default pre-generated name tends to be too long and include random characters.
10. Click **Finish**.

    The **TIBCO Cloud Mesh** service descriptor files (including the API files) are displayed in the **Project Explorer** section.



11. Drag and drop the mesh service from the **Project Explorer** to the application's **Process** window and select **Create Reference** from the resulting dropdown menu.

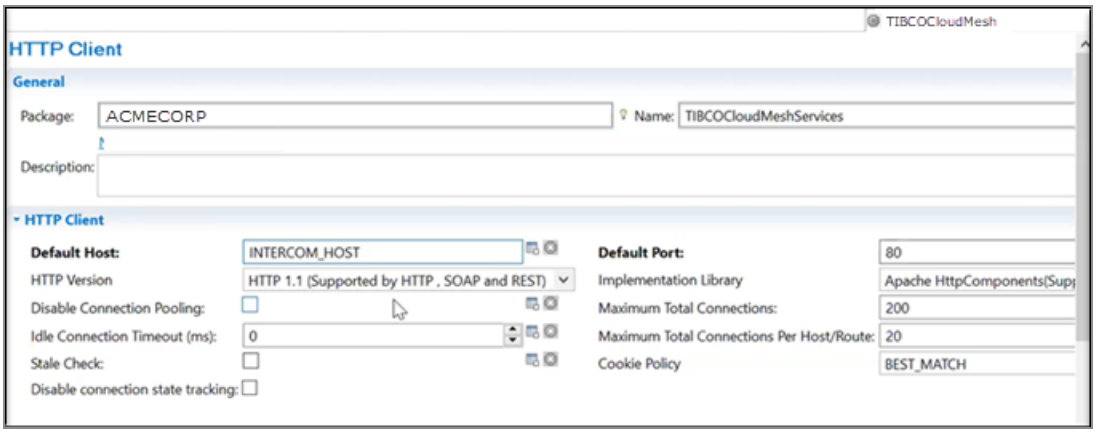
    The service icon is added to the application project window.



HTTP Client Resource

All reference bindings that use TIBCO Cloud™ Mesh are preconfigured to use a single, special HTTP client resource, called **TIBCOCloudMeshServices**. This resource cannot be changed in the reference binding.

Select the **TIBCOCloudMeshServices.httpClientResource** in the `Project Explorer` tree.

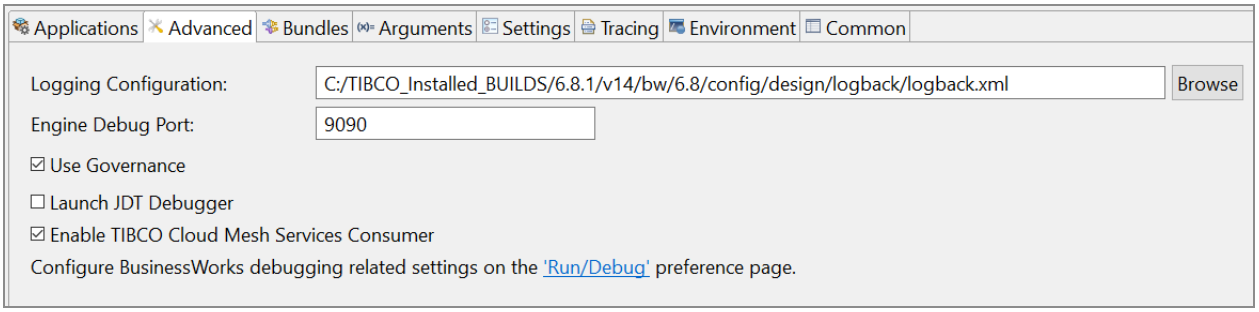


The **Default Host** field is automatically populated with the default value: `INTERCOM_HOST`. This value is a constant and cannot be changed.

To verify and run the newly created client, either use Studio Debugger to debug the client application (the debugger securely connects to the service running in the cloud and returns the output), or push the client application to the cloud and scale it up.

Note: To run or debug the client application in TIBCO Business Studio for BusinessWorks, ensure that in the **Advanced** tab, the **Enable TIBCO Cloud Mesh Services Consumer** checkbox is selected.

Single Sign On: After selecting the **Enable TIBCO Cloud Mesh Services Consumer** the checkbox, you can connect to the default cloud server mentioned in the *ssoConfig.properties* file.



REST Service Binding

When you create a REST service binding, **TIBCO Business Studio** generates a default name `Resource` and a default path `/resource` for the service. When deployed to **TIBCO Cloud Integration**, these names appear in the generated Swagger metadata for the endpoint. If left unchanged, all services deployed in this way appear indistinguishable in **TIBCO Cloud Mesh**. Therefore, it is a good idea to immediately give any new service binding a more specific name and path.

Note: If you are using **TIBCO Cloud™ Mesh**, you have an attribute Service App ID for reference binding in **TIBCO Business Studio™**. So in case the service application is changed and the new application also has the same mesh point. You can always provide the new application ID and the client application works fine.

HTTP Endpoints for Health Checks

When the TIBCO BusinessWorks Container Edition application is started, it creates an HTTP GET endpoint on port `7777` and the path `_ping` by default in TIBCO BusinessWorks Container Edition application. This endpoint can be used to configure health-checks on Docker or Kubernetes based platforms.

Configuring Probes in Kubernetes

To perform health checks in Kubernetes, configure the following probes:

- Liveness probe: Indicates when a container restarts.
- Readiness probe: Indicates when a container is ready to start accepting traffic.
- Startup probe: Checks if the application in a container is started.

The HTTP endpoint in the TIBCO BusinessWorks Container Edition application is used to configure readiness probes in Kubernetes. The endpoint must be accessed only after the application is started.

Setting Up Liveness Probe

To view the liveness of an application, you need to add the liveness probe in the `manifest.yml` file.

```
livenessProbe:
 httpGet:
   path: /health/liveness
   port: 7777
```

For more information, see the "Configure Liveness, Readiness, and Startup Probes" topic in the Kubernetes documentation.

Note: The liveness probe fails only if the container is unhealthy or non-responsive.

Setting Up Readiness Probe

To view the readiness of an application, you must add the readiness probe in the `manifest.yml` file.

```
readinessProbe:
 # an http probe
 httpGet:
   path: /health/readiness
   port: 7777
```

For more information, see the "Configure Liveness, Readiness, and Startup Probes" topic in the Kubernetes documentation.

Setting Up Startup Probe

To view the startup of an application, you need to add the startup probe in the `manifest.yml` file.

```
startupProbe:
 # an http probe
 httpGet:
   path: /health/startup
   port: 7777
```

For more information, see the "Configure Liveness, Readiness, and Startup Probes" topic in the Kubernetes documentation.

Checking Application Liveness in Docker

An HTTP application endpoint is exposed to check the application liveness or readiness in Docker.

While running Docker container, you need to bind the user defined host port with container's port `7777`. After the host port is bind and start the container, access the

endpoint URL:

```
<base_url>:<user_defined_host_port>/_ping
```

Service Registration and Discovery

The service registration and discovery mechanism lets TIBCO BusinessWorks Container Edition services announce their availability and lets clients dynamically find these services.

- **Cloud Foundry**

  Service registration and discovery on Cloud Foundry is done using one of the following options:

  - Support for service registration and discovery is provided through an integration with the Eureka Service Registry service.

    You first create an instance of the service using the VMware Tanzu Apps Manager and then bind this service instance to your application.

  - Support for Consul service registration and discovery is provided using the `CONSUL_SERVER_URL` environment variable. For more information, see Environment Variables for Cloud Foundry.

  Note:
  TIBCO BusinessWorks Container Edition communicates with Spring Cloud® Services using HTTPS. If your Cloud Foundry installation uses a self-signed SSL certificate, this certificate should be added to the buildpack before your application can be deployed. For more information, see Customize the Buildpack in *Certificate Management* to add self-signed SSL certificate to the buildpack.

- **Docker and Docker based platforms**

  - Support for Eureka service registration and discovery on Docker and Docker based platforms is provided using the `EUREKA_SERVER_URL` environment variable. For more information, see Environment Variables for Docker or Docker based Platforms.

  - Support for Consul service registration and discovery on Docker and Docker based platforms is provided using the `CONSUL_SERVER_URL` environment variable. For more information, see Environment Variables for Docker or Docker based Platforms.

- To discover services registered on Eureka Server using IP, the **bw.servicediscovery.eurekaclient.useIP** java property is used. The default value of the java property is False. This property needs to be a part of **BW_JAVA_OPTS** environment variable.

Service Registration Configuration

For information on configuration see *HTTP Connector* in the *Shared Resources* section of the *Palette Reference*.

> **Note:** On Cloud Foundry to use the direct registration mode set the ENABLE_SERVICE_DIRECT_REGISTRATION environment variable to `true`.
>
> For more information, see Environment Variables for Cloud Foundry.

Service Discovery Configuration

For information on configuration see *HTTP Client* in the *Shared Resources* section of the *Palette Reference*.

Support for Security Enabled Service Registry

Connection to Consul/Eureka via HTTPS (SSL) for Docker: For security enabled service registry for Docker, you need to copy the SSL certificate to the `resources/addons/certs` folder and create a base docker image.

Connection to Consul/Eureka via HTTP (SSL) for VMware Tanzu: For security enabled service registry for VMware Tanzu, you need to copy the SSL certificate to the `resources/addons/certs` folder in the buildpack.

Circuit Breaker Support

The Circuit Breaker feature is supported through the use of Resilience4j libraries.

By default, all the exceptions count as a failure. The list of the exceptions are not a part of this feature.

For more information, information, see *HTTP Client* in the *Shared Resources* section of the *Bindings and Palette Reference*.

- Grafana Dashboard Integration
  To monitor the events that are generated by the Resilience4j library, use Prometheus and Grafana for visualization.
  - VMware Tanzu - You can retrieve the Resilience4j metrics using `http://<routable url>/resilience4j_metrics`

    For example,

    ```
    http://myapp.demopcf.com:80/resilience4j_metrics
    ```

  - Docker or Docker-based platforms - You can retrieve the Resilience4j metrics using the command `http://<Container IP>:8090/resilience4j_metrics`

    For example,

    ```
    http://132.99.1.6:8090/resilience4j_metrics
    ```

Note: The applications running with Hystrix Circuit Breaker support with versions prior to 2.7.0, such applications base image does not need to be changed unless you want to link running applications with the TIBCO BusinessWorks Container Edition base image of 2.7.0.

For existing applications prior to TIBCO BusinessWorks Container Edition 2.7.0 with Circuit Breaker enabled through Hystrix, follow these steps to use Circuit Breaker using Resilience4j over Hystrix libraries.

Procedure

1. Import existing application in TIBCO Business Studio for BusinessWorks.
2. Navigate to **Problems** tab and select **EMF Validation Problem**. Resolve all the EMF validation problems related to Circuit Breaker configuration error.

> **Note:** Review the Circuit Breaker properties to ensure that the properties are configured as per the requirement after migrating to Resilience4j.

After all the issues related to Circuit Breaker configuration are resolved, you can use the application for Circuit Breaker configuration with Resilience4j libraries.

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. Subprocesses, 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

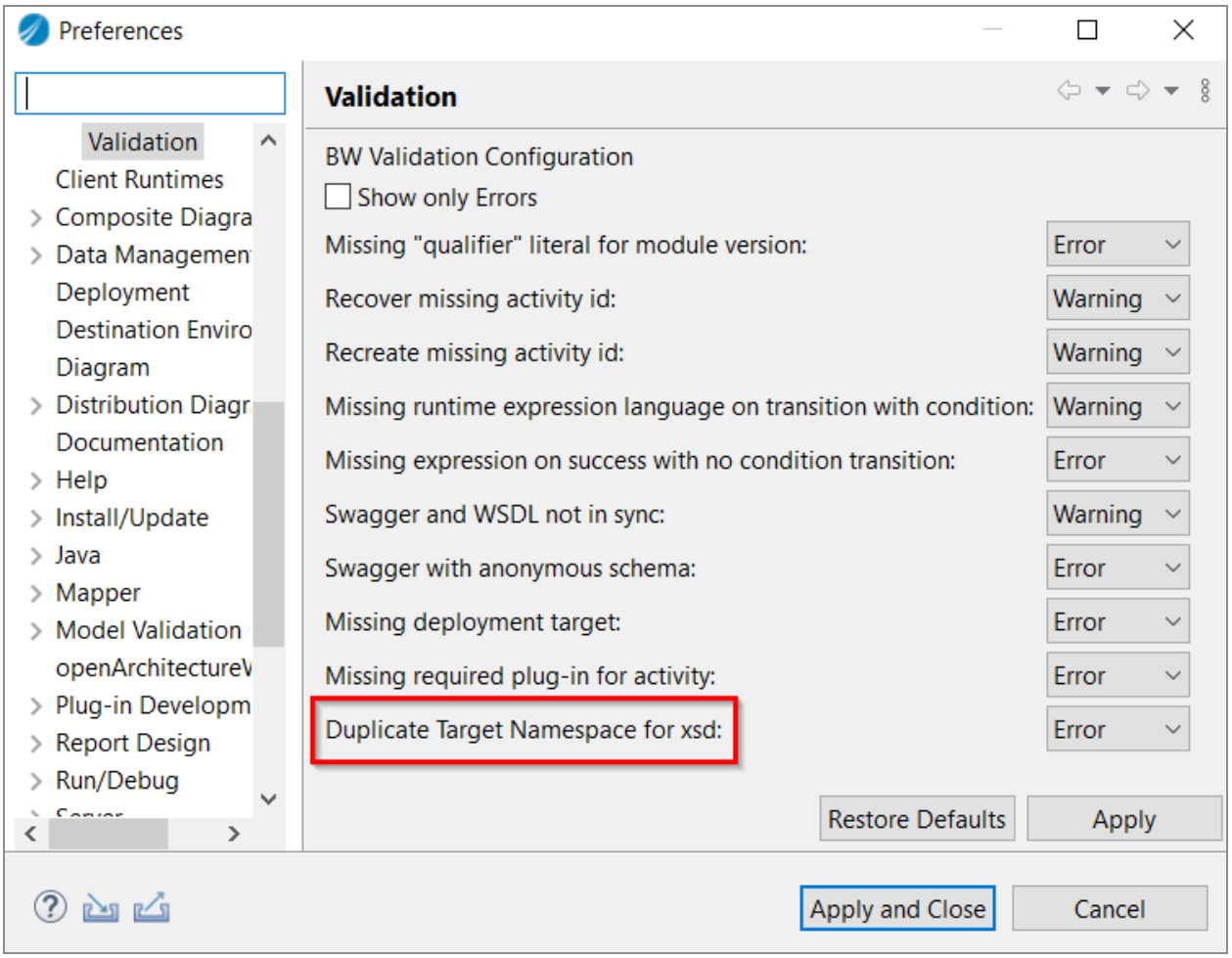
When designing complex business processes, ensure that the service contracts on the interfaces are well-defined.

Avoid XML Collisions

Avoid defining schema (XSD) or WSDL components with the same qualified names in the same module. Doing so can 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 a unique namespace by specifying the location of the schema document in the **Dependencies** section of the process.

A new preference, **Duplicate Target Namespace for XSD** is added under **BusinessWorks > Validation**. The preference is set to **Warning** by default. You can use this preference, in which **Error**, **Warning**, and **Ignore** options are provided, and as per the selection, you can see either an error or a warning on the **Problems** tab, or duplicate namespaces are ignored in TIBCO Business Studio for BusinessWorks.



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 TIBCO BusinessWorks™ Container Edition 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. TIBCO recommends you to 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 TIBCO BusinessWorks Container Edition 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

TIBCO recommends you to perform the **Project > Clean** operation in case of changes in the XSD or WSDL files.

The Support for Undo-Redo Operations is Limited

TIBCO suggests you 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 Actions on TIBCO BusinessWorks Container Edition Activities and Processes is Limited

To reuse the Copy functionality for TIBCO BusinessWorks Container Edition 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 TIBCO® Cloud Integration.

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.

- **Issue 1**:

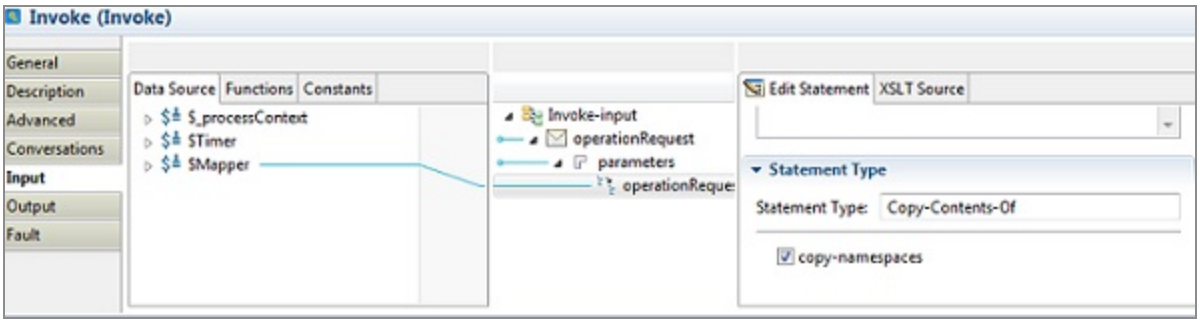
  ```
  "The expression refers to a variable name,variableName,that is not
  defined in the static context".
  ```

  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.GenXDMException: The prefix 'tns' is
  already bound to http://NamespaceTest.com/Example Caused by:
  org.genxdm.exceptions.GenXDMException: 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** checkbox. See the following image.


Invoke (Invoke)
General
Description
Advanced
Conversations
Input
Output
Fault
Data Source
Functions
Constants
$_processContext
$Timer
$Mapper
Invoke-input
operationRequest
parameters
Edit Statement
XSLT Source
Statement Type
Statement Type:
Copy-Contents-Of
copy-namespaces

TIBCO Documentation and Support Services

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

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the TIBCO BusinessWorks™ Container Edition page:

- *TIBCO BusinessWorks™ Container Edition Release Notes*
- *TIBCO BusinessWorks™ Container Edition Installation*
- *TIBCO BusinessWorks™ Container Edition Application Development*
- *TIBCO BusinessWorks™ Container Edition Application Monitoring and Troubleshooting*
- *TIBCO BusinessWorks™ Container Edition Bindings and Palettes Reference*
- *TIBCO BusinessWorks™ Container Edition Concepts*
- *TIBCO BusinessWorks™ Container Edition Error Codes*
- *TIBCO BusinessWorks™ Container Edition Getting Started*
- *TIBCO BusinessWorks™ Container Edition Migration*
- *TIBCO BusinessWorks™ Container Edition Performance Benchmarking and Tuning*
- *TIBCO BusinessWorks™ Container Edition REST Implementation*
- *TIBCO BusinessWorks™ Container Edition Refactoring Best Practices*
- *TIBCO BusinessWorks™ Container Edition Samples*

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

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

Legal and Third-Party Notices

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software 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. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

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 "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.