



TIBCO BusinessWorks™ Container Edition

Getting Started

Version 2.10.0 | December 2024



Contents

| | |
|---|-----------|
| Contents | 2 |
| About the Getting Started Guide | 4 |
| Orientation | 5 |
| TIBCO Business Studio™ for BusinessWorks™ | 6 |
| Application Development | 6 |
| Web Services | 8 |
| Shared Resources | 8 |
| Cheat Sheet | 9 |
| REST Support | 11 |
| REST Documenter and Tester | 12 |
| Discovering API Models from TIBCO Business Studio for BusinessWorks | 12 |
| Importing an API Model into your Workspace | 15 |
| Creating an XML Schema for a Swagger File | 17 |
| Consuming a REST Endpoint in TIBCO Business Studio for BusinessWorks | 18 |
| Synchronizing the Imported REST API Models in TIBCO Business Studio for BusinessWorks | 19 |
| Archive Files | 20 |
| Debugger | 22 |
| Runtime | 24 |
| Changing Help Preferences | 25 |
| REST Service Tutorial | 27 |
| Creating a Service Instance of Cloud Foundry managed PostgreSQL Database Service | 27 |
| Importing a Process Package | 28 |
| Building a REST Service | 31 |
| Testing the REST Service in Cloud Foundry | 37 |

| | |
|---|-----------|
| Testing the POST and GET Operations | 40 |
| Testing the REST Service in OpenShift | 41 |
| Troubleshooting | 46 |
| REST Reference Tutorial | 47 |
| TIBCO Documentation and Support Services | 52 |
| Legal and Third-Party Notices | 54 |

About the Getting Started Guide

This guide contains tutorials that are designed to familiarize you with a representational set of activities you might use to develop an application. By referring to these simple tutorials, you can understand how to use TIBCO BusinessWorks™ Container Edition within each phase of the project life cycle.

These tutorials illustrate the basic activities for creating an application, building and testing a simple REST service, and basic information on deploying the Administration sample applications using the provided scripts.

For more information, see one of the following topics:

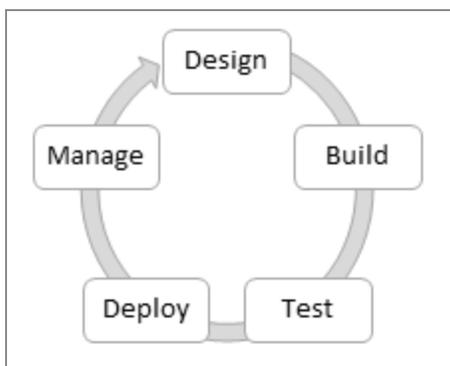
- [REST Service Tutorial](#): Walks you through the steps to build and test a simple REST Service using TIBCO BusinessWorks Container Edition and the Swagger UI.
- [REST Reference Tutorial](#): Shows you how to create a simple REST Invoke to an existing REST Service defined by a Swagger specification.

The [Orientation](#) section introduces you to the TIBCO Business Studio for BusinessWorks development environment. Before you continue, read the *TIBCO BusinessWorks Container Edition Concepts* guide to familiarize yourself with the TIBCO BusinessWorks Container Edition concepts and terminology.

Orientation

TIBCO BusinessWorks Container Edition is an integration product suite for enterprise, web, and mobile applications.

TIBCO Business Studio for BusinessWorks allows you to create services and integrate applications using a visual, model-driven development environment, and then deploy them in the TIBCO BusinessWorks Container Edition runtime environment.



This product uses Eclipse-based graphical user interface (GUI) provided by TIBCO Business Studio for BusinessWorks to define business processes and generate deployable artifacts in the form of archive files. These deployable artifacts can be:

- deployed and run in the product runtime, and
- managed using the administration command line console or `bwadmin`, or the web-based Admin UI.

For information about developing applications and TIBCO Business Studio for BusinessWorks, see the following guides in the documentation set:

- *Application Development*
- *Palette References*
- *Samples*

TIBCO Business Studio™ for BusinessWorks™

TIBCO Business Studio for BusinessWorks is the design-time IDE (based on Eclipse) where you create and test your processes.

You use TIBCO Business Studio for BusinessWorks for end-to-end application development. You can create new services, orchestrate business process, and integrate applications in a short time. A model-driven development approach is supported, with a rich set of palettes for process design. These palettes can be used to visually create and test business processes that connect to various technologies such as a database, messaging servers, and so on.

TIBCO Business Studio for BusinessWorks is installed as part of TIBCO BusinessWorks Container Edition.

To open TIBCO Business Studio for BusinessWorks:

- On Unix: Run the TIBCO Business Studio for BusinessWorks executable located in the `$TIBCO_HOME/studio/<version>/eclipse/` directory.
- On Windows: Click **Start > All Programs > TIBCO > TIBCO_HOME > TIBCO Business Studio <version_number> > Studio for Designers**

On the **Workspace Launcher** dialog, accept the default workspace or browse to create a new workspace, and click **OK**.

When TIBCO Business Studio for BusinessWorks starts, the default development environment, a *workbench*, is displayed.

For more information, see the *Application Development* guide.

Application Development

Applications solve integration problems of varying complexity. Using TIBCO Business Studio for BusinessWorks, applications can be developed using an application-oriented integration style or a service-oriented integration style. How you design your application's integration style depends on the following factors:

- Speed of integration
- Data abstraction
- Richness of operation primitives

- Typical endpoints

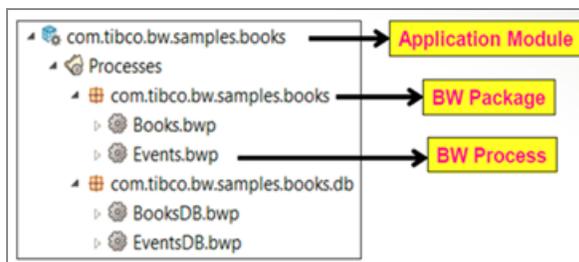
For more information about an application's integration style and other application design considerations, see "Application Design Considerations" in the *TIBCO BusinessWorks Container Edition Application Development* guide.

Processes allow you to implement business logic that can obtain and manage the flow of information in an enterprise between a source and different destinations. In process-driven design, the business processes or integration flows are first realized and captured. For more information about process design, see "Process Design Considerations" in the *TIBCO BusinessWorks Container Edition Application Development* guide.

Processes developed in TIBCO Business Studio for BusinessWorks are saved in an application module. Application modules are equivalent to projects and are saved to folders on the disk. The TIBCO Business Studio for BusinessWorks workspace contains one or more application modules.

- An application module contains one or more packages
- A package contains one or more processes, which in turn are main processes or subprocesses
- A process is stored as a single file with a .bwp extension

An application module contains a special folder called **Processes**. This folder contains the user created processes. In addition, an application module also contains folders to store WSDL files, schemas, and shared resources. The **Processes** folder is shown below.



i Note: A package should follow the Java naming convention.

Processes are designed in the **Process Editor**. Activities and shared resources help you rapidly design business processes. An activity is the individual unit of work in a process. There are multiple ways to add an activity to a process: from the right-click menu on the **Process Editor**, from the palettes, and from the **File Explorer** or **Project Explorer**. To add an activity from the palette, select it and drop it on the **Process Editor**.

Implemented services are shown as chevrons on the left side of the **Process Editor**. References that are invoked are shown on the right side of the **Process Editor**. For a simple process, services and references are optional.

Web Services

Web services are application components that communicate using open standard protocols. You can develop SOAP-based web services using the Generate Concrete WSDL Wizard. The wizard generates a WSDL file and the appropriate response activities. You can develop REST-based web services using the REST Service Wizard in TIBCO BusinessWorks Container Edition.

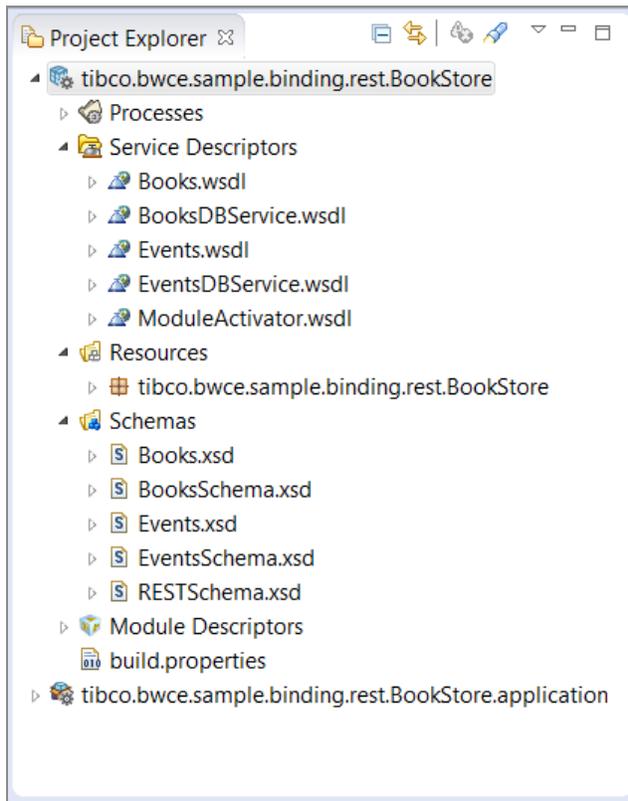
Select a WSDL file in the **Project Explorer** and drop it on the **Process Editor** to implement a web service. Dropping the WSDL file displays a menu for creating services or implementing operations. Response activities are automatically generated.

To create a REST service, select a path under the `.json` file in the **Service Descriptors** folder and drop it on the **Process Editor** to implement a web service. When you drop the path, it displays a menu with an option to create a service or a reference.

Shared Resources

Shared resources are configurations that are shared among activities. These are resources such as database, JMS and HTTP connections, and connections to other servers. Resources are added to special folders in the **Project Explorer**. The following image shows these folder in the **Project Explorer**.

Shared Resources Folders in Project Explorer



The following types of folders for shared resources can exist in a project.

- **Resources:** Contains shared resources used by activities in a process.
- **Schemas:** Stores XSD (schema) files.
- **Service Descriptors:** Stores WSDL and JSON files.

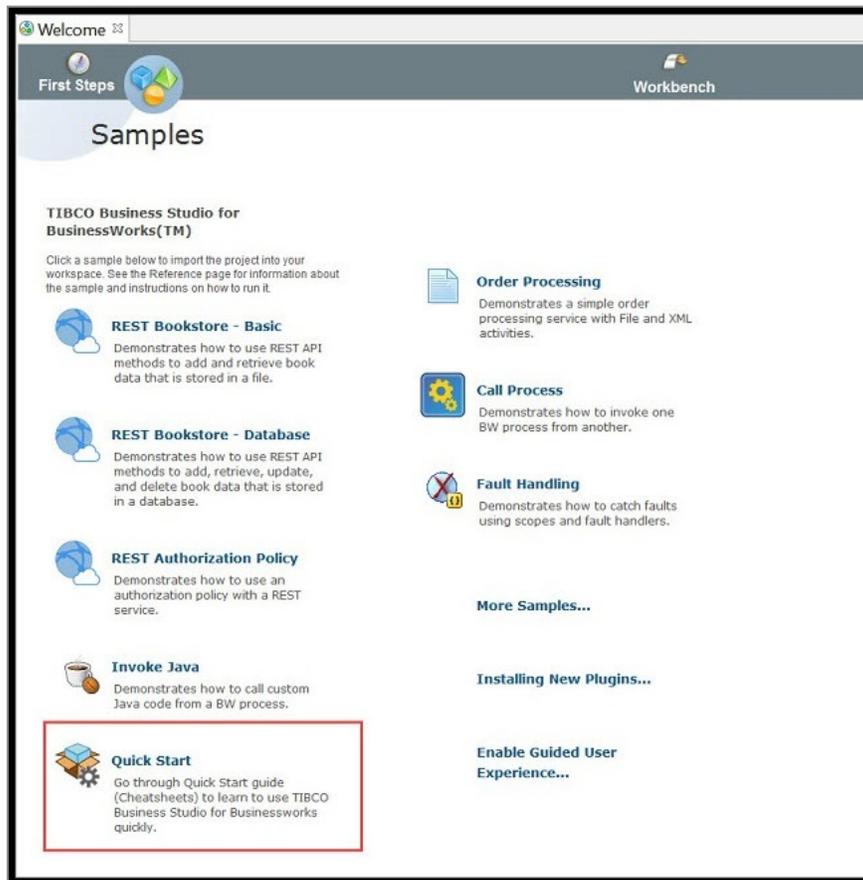
Cheat Sheet

A quick reference help is available in the form of cheat sheets for few common tasks in TIBCO Business Studio for BusinessWorks.

Open a cheat sheet using any one of the following ways:

- On the Welcome page, click the Quick Start link. If you open TIBCO Business Studio for BusinessWorks in a new workspace, the Welcome screen is displayed. If you open TIBCO BusinessWorks™ Container Edition in an existing workspace, open the

Welcome page by using **Help > Welcome**.



- Click **Help > Cheat Sheets**.

Cheat sheets are available for the following tasks:

- Adding unit testing to a project.
- Creating a basic SOAP application.
- Creating an application using Java Invoke.
- Configuring a JMS connection and creating a JMS send message application.
- Configuring a JDBC connection using custom driver.
- Developing a REST API to expose a SOAP service.
- Exposing a SOAP service via REST API.
- Using security policies in a project.

REST Support

The REST Service wizard is used to build RESTful services.

i Note: When you create a REST service, make sure to edit the **Default Host** field in the HTTP Connection Resource to reflect the actual host name. By default, the Default Host field is set to localhost.

REST Service Wizard

Developing a RESTful service is a simple three-step process:

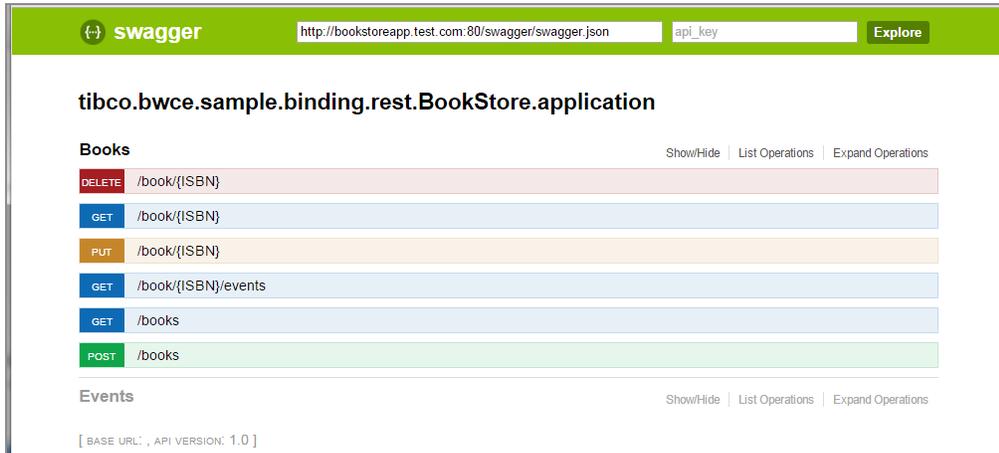
1. Name the REST resource.
2. Choose the resource definition (the XSD schema).
3. Choose the REST operations to implement.

The input and output messages for the operations are automatically generated along with a Response activity. An HTTP shared resource is also generated with the default configuration. You can then add activities and implement the business logic for each operation in the process.

REST Documenter and Tester

A REST documenter and tester is automatically generated for a REST resource. Refer to **OSGi commands to List REST URLs** in the *REST Implementation* guide. The documentation is based on the Swagger specification and is rendered using the Swagger UI.

Swagger UI



This Swagger based interface provides a convenient and quick way to:

- View REST endpoints and operations implemented by the REST resource service.
- Examine the inputs and outputs for each operation in JSON format.
- Enable **Input** fields to specify JSON or XML input for each operation.
- Invoke an operation and receive a live response for the input supplied.

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 dropdown 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

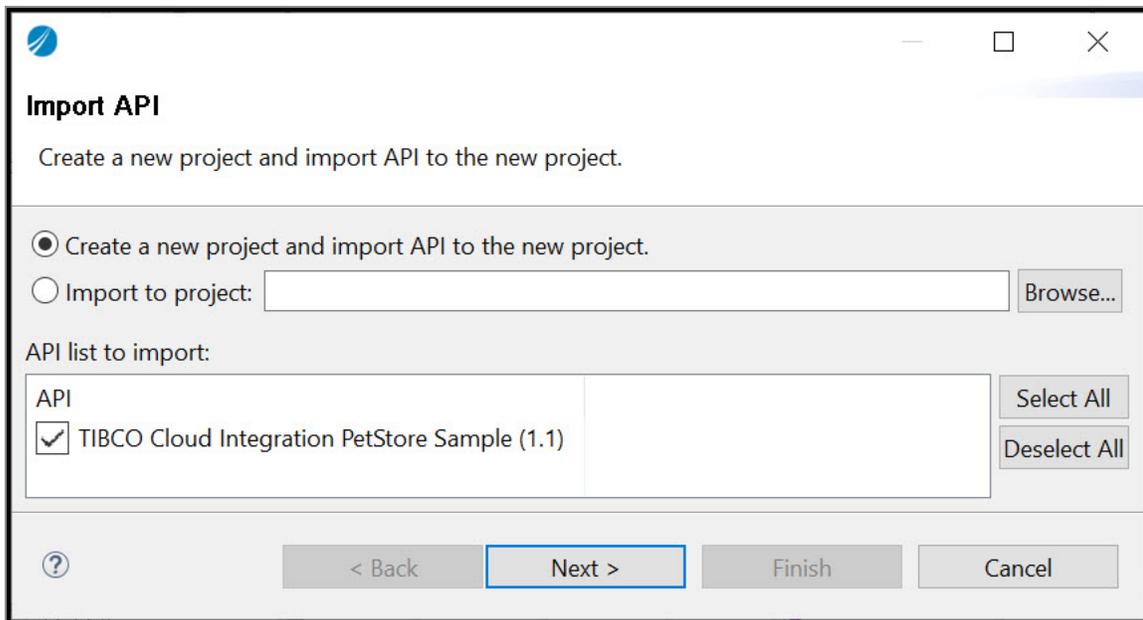
The APIs that are discovered from local and remote servers are displayed in the **API Explorer** tab of the TIBCO BusinessWorks Container Edition. You can use these APIs in your project by importing them into the **Service Descriptors** folder of the project. The .json file for the API gets copied into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



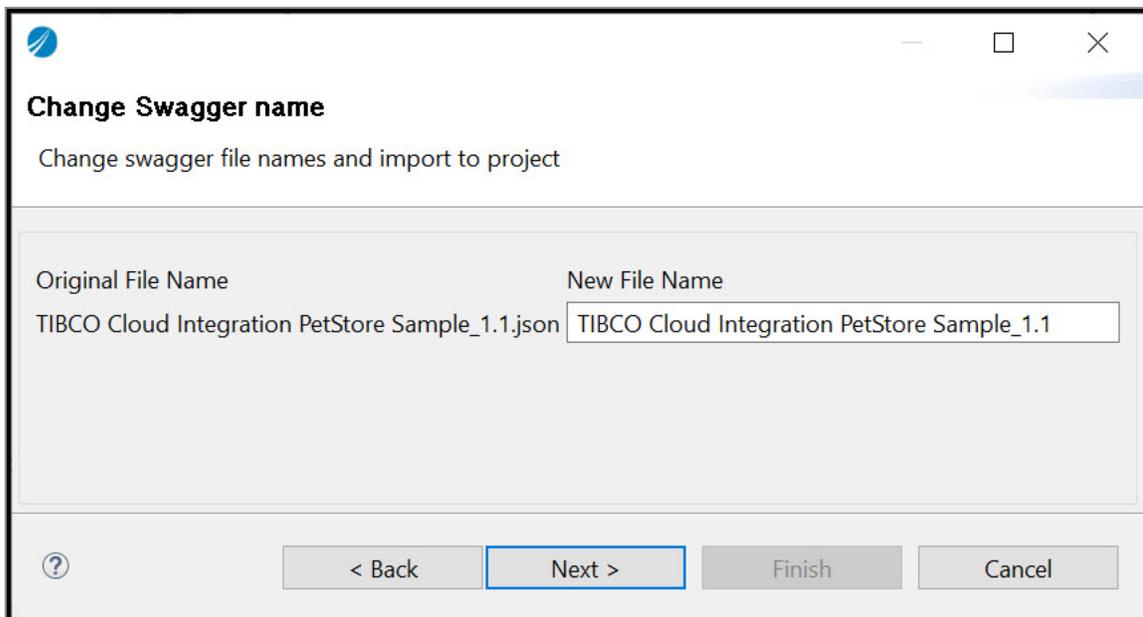
Every API you selected in the **API Explorer** is listed in this dialog. If an API has multiple versions, all versions are listed. By default, all APIs listed here are selected. You can deselect APIs that you do not want to import by clearing its checkbox.

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 |

| Option | Description |
|--|---|
| | 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.

i Note: APIs that were created using a Swagger file must be implemented exactly as defined by the Swagger file. TIBCO BusinessWorks Container Edition 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. Ensure 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.

Consuming a REST Endpoint in TIBCO Business Studio for BusinessWorks

Create a REST Reference binding to consume a REST endpoint.

Note: You cannot edit the REST reference binding configuration for APIs that are imported from a source external to TIBCO Business Studio for BusinessWorks.

Before you begin

Swagger API documents must be imported into the project's **Service Descriptors** folder. This gives you the ability to expand and collapse endpoints, operations, parameters, and response codes in the **Project Explorer**.

To consume a REST API that exists in the **Service Descriptors** of the project, do the following:

Procedure

1. Expand the Swagger file in the **Service Descriptors** special folder to view the endpoints, operations, parameters, and response codes.
2. Drag an endpoint on the right side of the canvas to create a REST reference binding. This creates a cloud shaped icon with a right facing arrow. The cloud is an indication that it is a REST reference whereas the arrow within the cloud indicates that it is a binding. Since the binding is within a cloud, it is an indication that it is a REST binding. You cannot convert a REST binding to a SOAP binding or the opposite way.

Note: When you create a REST reference for the service, make sure to edit the **Default Host** field in the HTTP Client Resource to reflect the actual host name. By default, the **Default Host** field is set to localhost.

3. Drag and drop an operation from the REST reference binding on to the canvas. This creates an **Invoke** activity, which is pre-configured to invoke the operation. It

also creates an HTTP Client Shared Resource with the host name and port number. The configuration for these entities is copied from the Swagger document from which you created the reference binding. The reference consists of the name of the API as well as the operations that it supports.

When invoking a POST or PUT method, you must provide the request string in the **Input** tab. To do so, click the column next to **item** under **postRequest** in **XPath Expression** and provide the request string in the dropdown box.

4. Test the configured process using the TIBCO Business Studio for BusinessWorks debugger.

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.

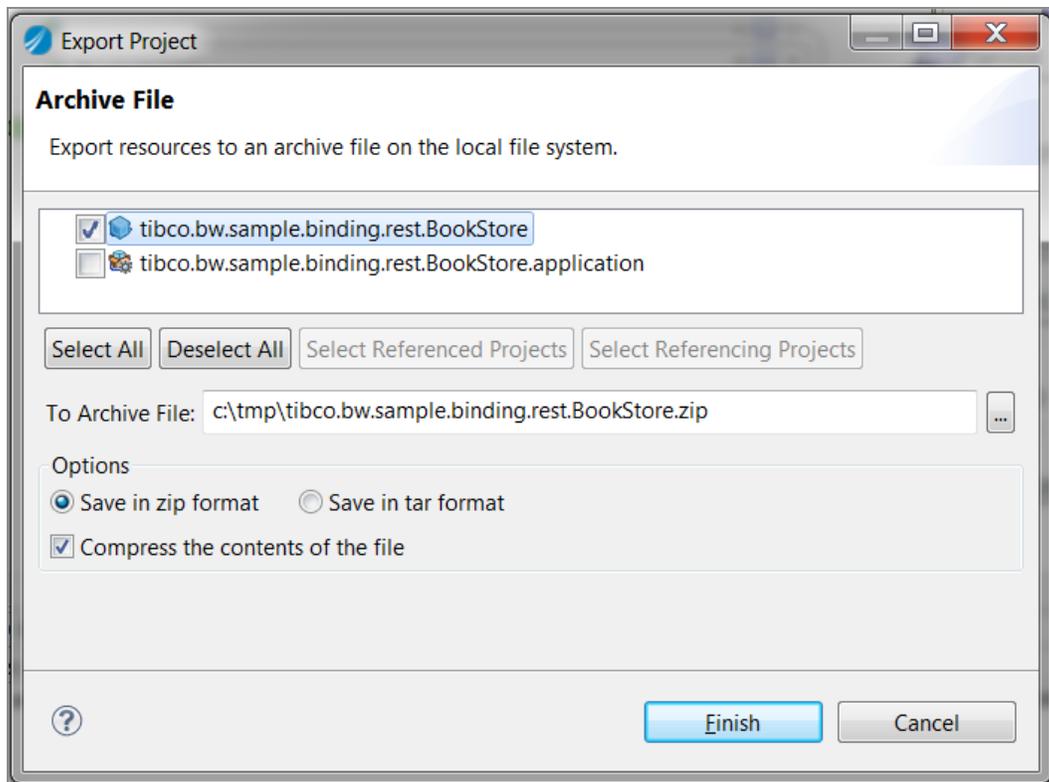
Archive Files

After completing an application module, you must define an application to build a deployment archive file. An application defines all the processes, properties, and resources that must be included in the archive file. By default, all processes are included.

To create an archive, choose one of the following:

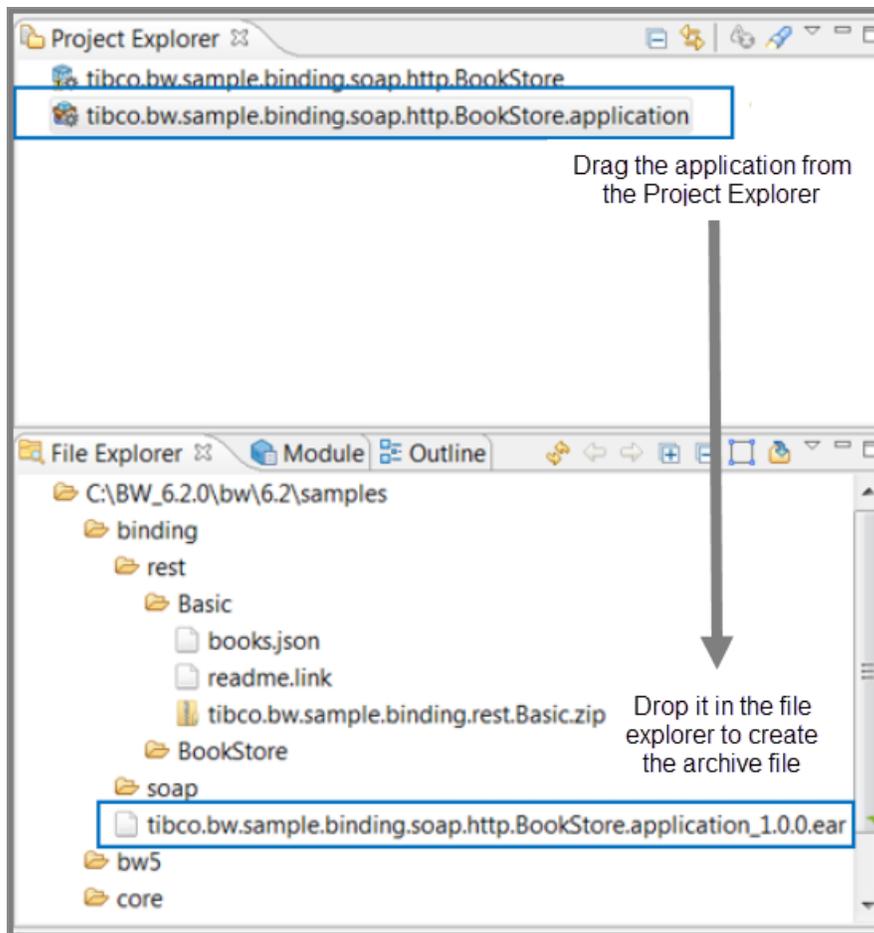
- Right-click the project in the **Project Explorer** and choose **Export > Studio Projects to Archive**. The **Export Project** dialog is displayed.

Export Project Dialog



- Drag the project from the **Project Explorer** and drop it on a folder in the **File Explorer**.

Drag and Drop Project to File explorer

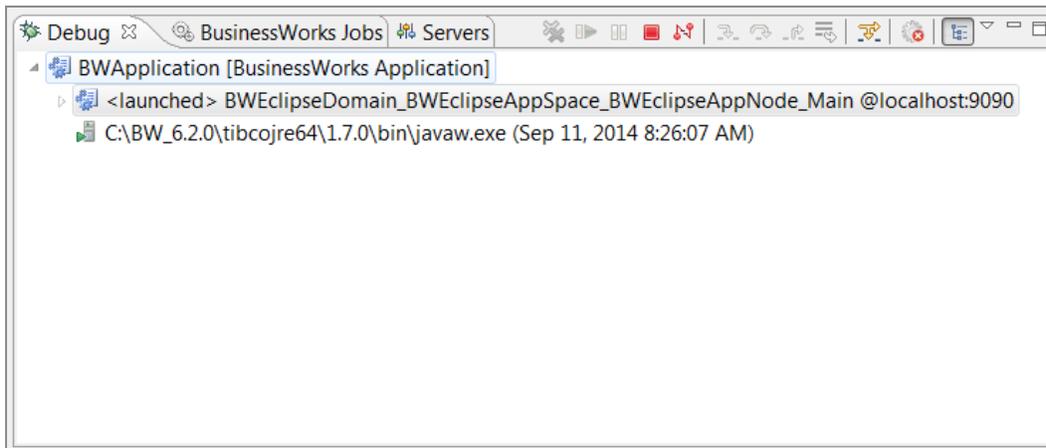


In both scenarios, the archive file is created with all required processes, properties, and resources. In the first scenario, you can name the archive file, select the format, and select the resources to include. In the second scenario, the archive is created for you in the format appropriate for your operating environment. All required elements are included.

Debugger

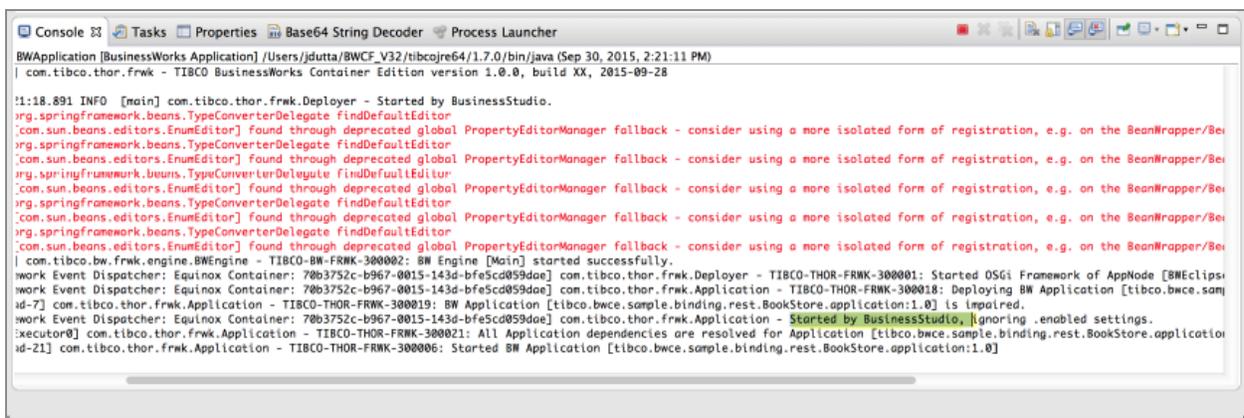
TIBCO Business Studio™ for BusinessWorks™ debugger is used to test processes during the process development stage. Starting the debugger brings up the **Debug** perspective. This perspective can be used to set breakpoints, steps through processes, examine job variables, and activity input/output at each step.

Debug Perspective



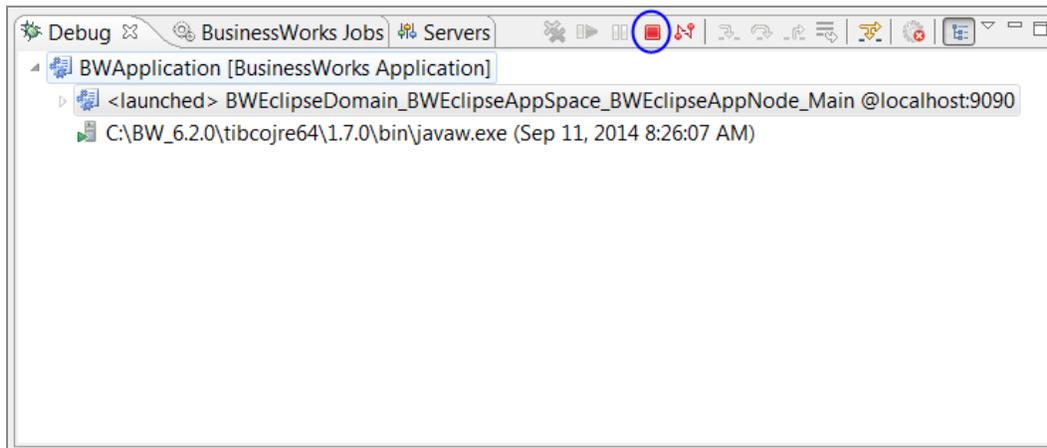
The **Console** view displays the messages and errors returned by the runtime.

Console View



Start the debugger with the **Run > Debug** command. To stop the debugger, press the **Stop** icon on the **Debug** perspective toolbar:

Stop Icon in Debug Perspective

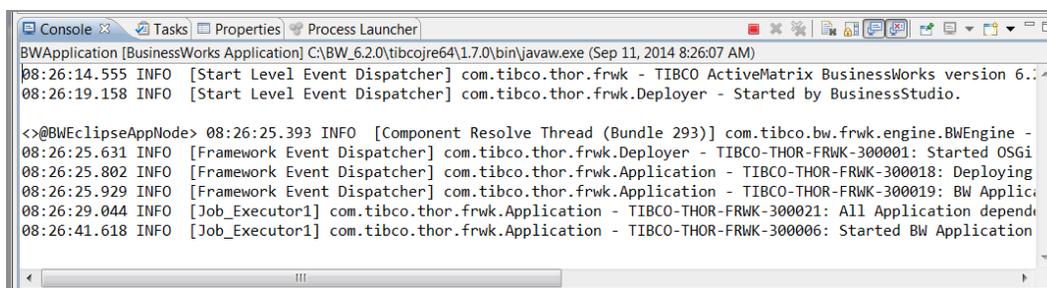


Runtime

You can run applications in TIBCO Business Studio for BusinessWorks and test them in a runtime environment, which consists of a domain, an AppSpace, and an AppNode on your local machine. These runtime entities were created when you installed TIBCO BusinessWorks Container Edition. For more information about runtime entities, see the *TIBCO BusinessWorks Container Edition Concepts* guide.

To run an application in TIBCO Business Studio for BusinessWorks, choose the **Run > Run** command. (Applications can also be run with the **Run > Run Configurations** command. This option enables you to manage and open the run configurations.) The Run command opens the **Console view** where progress messages and errors are displayed.

Console View



Click the **Businessworks Jobs** view in the top left to see the jobs created for the process. To stop the current job, click the **Stop** button  on the **Console view** toolbar.

From the **Console view**, you can use OSGi commands to monitor the running AppNode and gather metrics about your application. For information about OSGi commands, press Enter in the **Console view** to display the `<>@BWEclipseAppNode>` prompt. Type `help` to get a list of commands.

The scope is indicated along with the command. Commands with the scope `bw` return information about the running application. Type a command name followed by `-h` for information about the command. For example, the command `help bw:dsr` returns:

```
dsr - Diagnoses Shared Resource issues
  scope: bw
  parameters:
    String Partial or full name of a Shared Resource. Case is
    ignored.
```

Changing Help Preferences

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

Before you begin

Before changing the help preferences to access documentation locally or from an internal web server, download the documentation.

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

To point to a custom location:

Procedure

1. Perform one of the following steps in TIBCO Business Studio for BusinessWorks based on your operating system:
 - On Windows OS: Click **Window > Preferences**
 - On macOS: Click **TIBCO Business Studio > Preferences.**
2. In the Preferences dialog, click **BusinessWorks > Help.**
3. Click **Custom Location**, and then browse to the `html` directory in the folder where you extracted the documentation or provide the URL to the `html` directory on your internal web server.
4. Click **Apply**, and then click **OK.**

REST Service Tutorial

The REST Bookstore sample lets you explore the REST tooling in TIBCO Business Studio for BusinessWorks. You can import this sample into TIBCO Business Studio for BusinessWorks through **File Explorer** and examine the project and the solution implemented by it.

The processes in the sample implement different aspects of a bookstore, such as adding books, deleting a book, and getting a list of books or a single book by ISBN. For more information about the sample, see "Using REST to Manage Books for a Bookstore" in the *TIBCO BusinessWorks Container Edition Samples* guide. This tutorial walks you through the steps to build an additional REST service for the sample and test it in the debugger. You can use the Swagger UI to invoke the operations for the REST resource.

Prerequisites

- Access to a locally running PostgreSQL database.
- The latest version of Google Chrome.

Creating a Service Instance of Cloud Foundry managed PostgreSQL Database Service

To bind an application to a managed PostgreSQL service already running in your Cloud Foundry environment complete the following steps:

Procedure

1. In your Cloud Foundry environment, run:

```
cf services
```

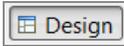
2. Run

```
cf create-service postgresql default postgres
```

Importing a Process Package

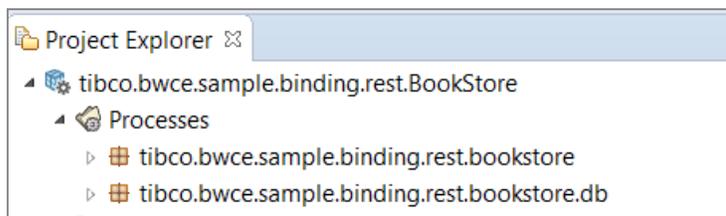
These steps show how to create a new process package.

Procedure

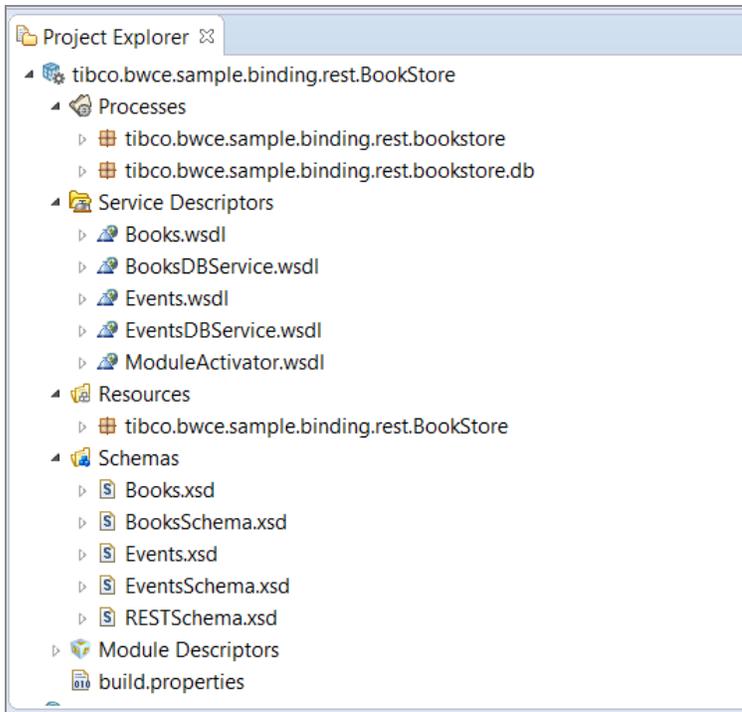
1. Open TIBCO Business Studio for BusinessWorks.
2. Open the **Design** perspective by clicking the  icon in the upper right.
3. Click the **File Explorer** tab. If the tab is not visible, click **Window > Show View > Other > FileSystem > File Explorer** and click **OK**.
4. Click **File > Switch Workspace** and select or open a clean new workspace.
5. In the samples directory, select **cloudfoundry > binding > rest > Bookstore** and double-click **tibco.bwce.sample.binding.rest.BookStore.zip**.

This opens the project in the **Project Explorer**.

6. In the **Project Explorer**, expand the **tibco.bwce.sample.binding.rest.BookStore** project.
7. You can also import the sample using the **File > Import > General > Existing Studio Projects into Workspace > Select Archive File > Browse** option.
8. The project is displayed in the **Project Explorer** panel on the left.

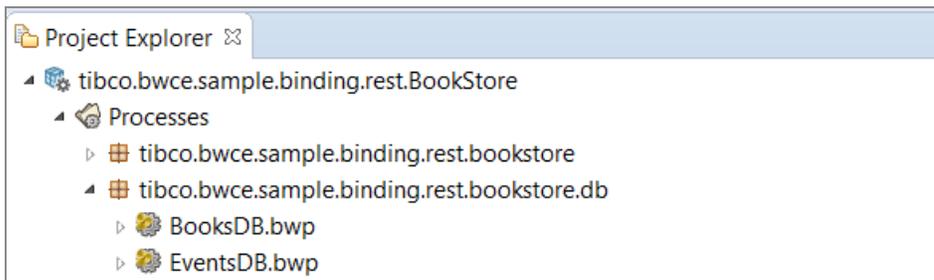


9. Expand the folders in the project to see all the project processes and resources. Refer to the *Application Development* guide for information about the folder structure.



10. Expand **Processes** and then expand **tibco.bwce.sample.binding.rest.bookstore.db**. The **BooksDB.bwp** process is displayed.

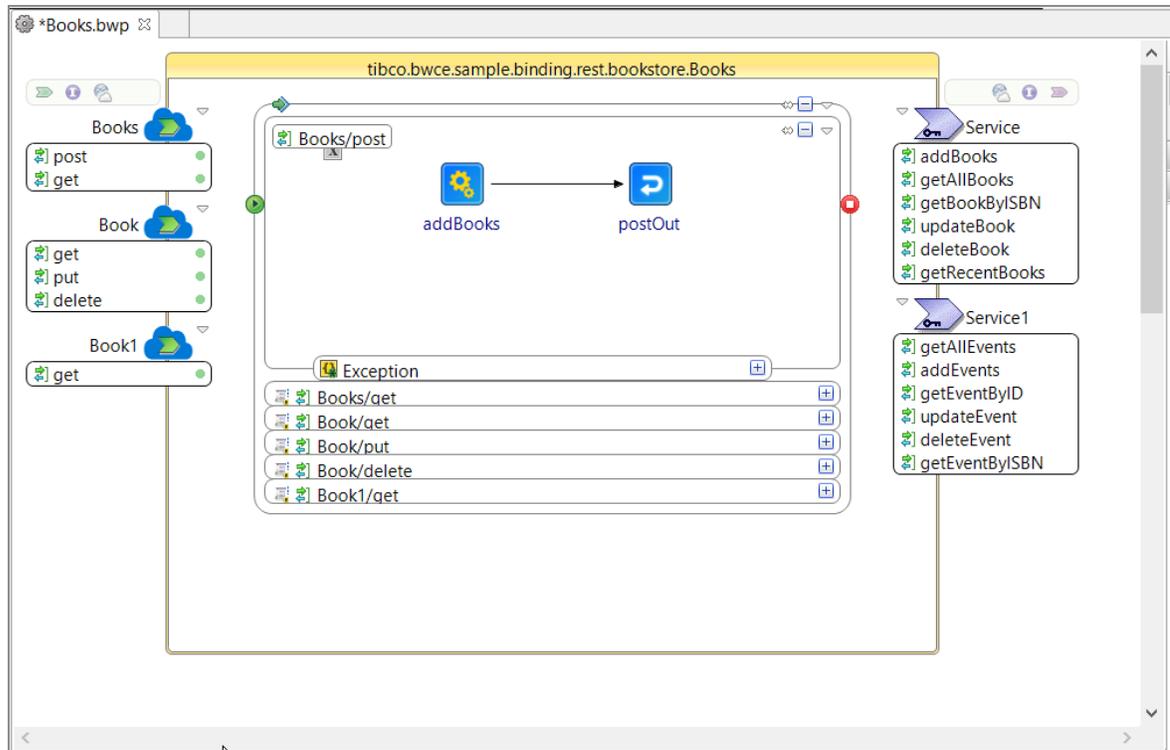
Note: bwp is a BusinessWorks process.



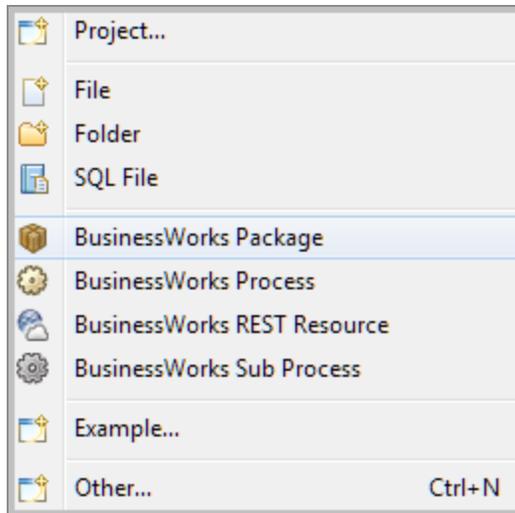
11. Double-click **BooksDB.bwp**.

The BusinessWorks process comprises:

- Green chevron on the left indicates the service details.
- addBooks, getAllBooks, and other operations implemented by this process.
- Each operation is implemented separately.



12. Double-click an operation to display the process for example, **BooksPersist > addBooks**.
 - a. In the addBooks operation, a JDBC activity is seen.
 - b. The activity is repeated using a ForEach group.
 - c. addBooksOut represents the **Response** to the web service request.
13. To add a new process package named tibco.bwce.sample.rest, right-click on **Processes** in the Project Explorer and select **New > BusinessWorks Package**.



14. In the BusinessWorks Package screen, specify `tibco.bwce.sample.rest` in the **Name** field.
15. Click **Finish** and verify that the new package `tibco.bwce.sample.rest` has been added in the Project Explorer.

Building a REST Service

This section details how to build a REST service.

Before you begin

The `tibco.bw.sample.binding.rest.BookStore` sample is loaded in the Project Explorer.

Procedure

1. To define a REST Resource named MyBooks, select `tibco.bw.sample.binding.rest.BookStore` > **New** > **BusinessWorks REST Resource**.
The REST Service Wizard window is displayed.

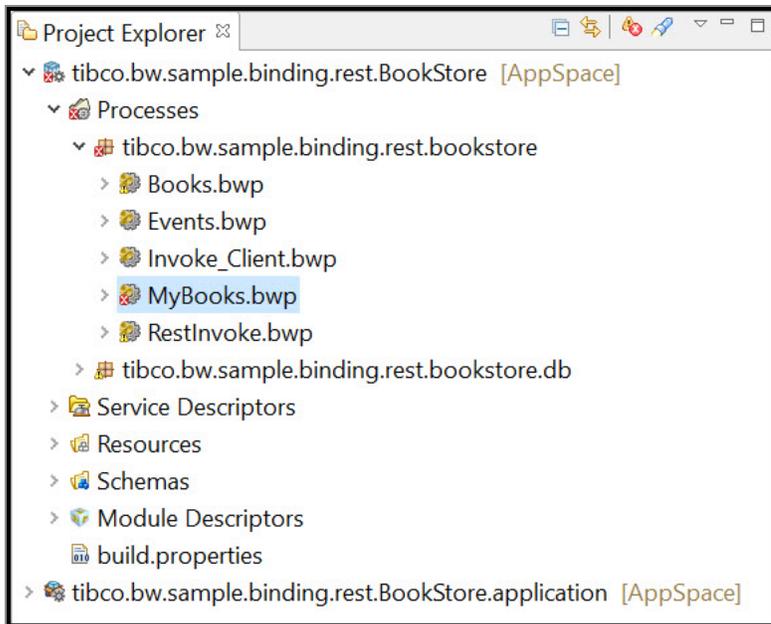
The screenshot shows the 'REST Service Wizard' window with the title 'Create a new REST Resource'. The subtitle is 'Configure the REST Service implementation.' The window contains the following fields and options:

- Resource Name:** A text box containing 'Resource'.
- Summary:** A text box containing 'Resource service'.
- Resource Service Path:** A text box containing '/resource'.
- Resource Definition:** An empty text box with a 'Browse...' button to its right.
- Operations:** A group of checkboxes:
 - POST
 - GET
 - PUT
 - DELETE
 - PATCH
 - OPTIONS
 - HEAD
 - An 'Add Custom Operation' button.
- Implementation Data:** Radio buttons for 'Structured' (selected) and 'Opaque'.

At the bottom of the window, there are four buttons: a help icon (?), '< Back', 'Next >', and 'Finish' (highlighted with a blue border). A 'Cancel' button is also present to the right of 'Finish'.

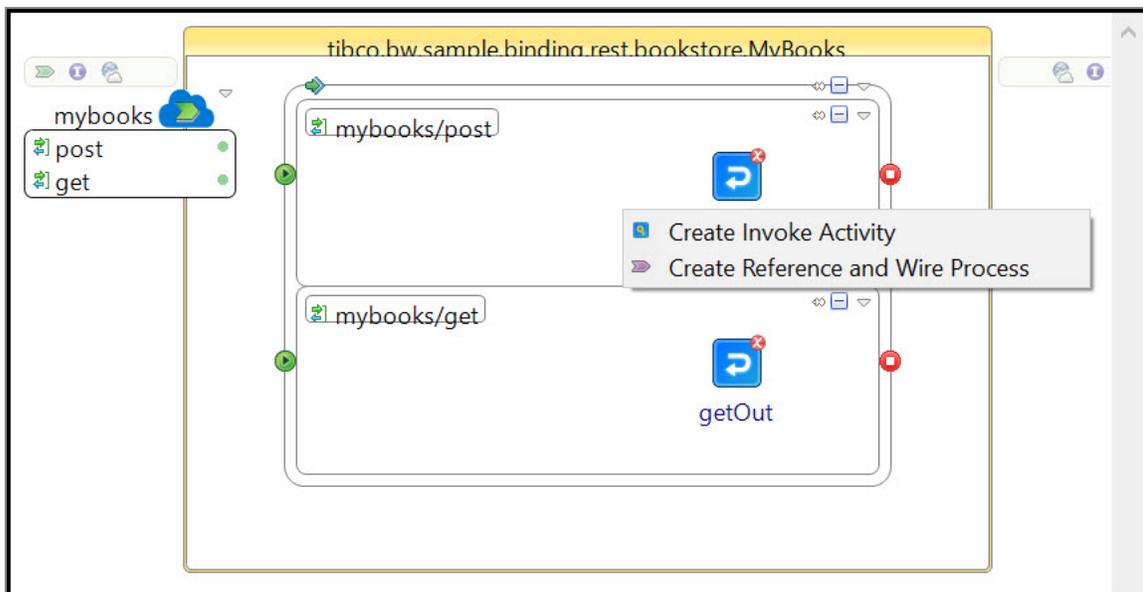
2. Specify the following values in the REST Service Wizard window.
 - a. **Resource Name:** MyBooks
 - b. **Summary:** Summary about the new REST service. (default)
 - c. **Resource Service Path:** Auto-filled
 - d. **Resource Definition:** Select **Browse > Schemas > Books.xsd > Books** in the Select Schema Element Declaration window.
 - e. **Operations:** Select POST and GET checkboxes.
 - f. **Implementation Data:** Accept the default value of **Structured**.
3. Click **Finish**.

This creates a new process **MyBooks.bwp** process is opened in the **Process Editor**.



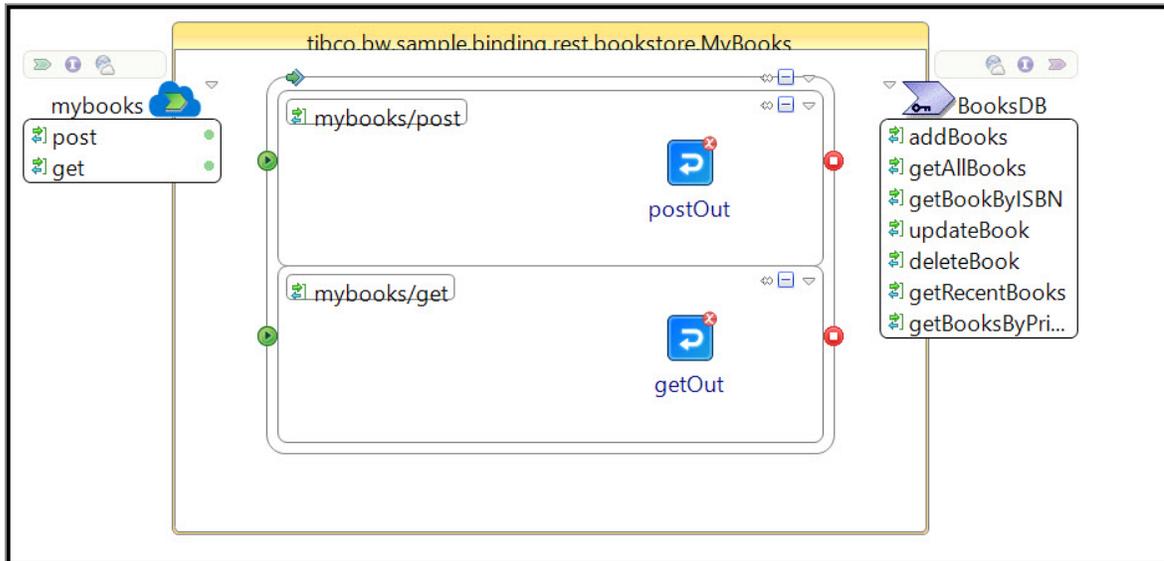
- Open the **tibco.bw.sample.binding.rest.bookstore.db** package in the **Project Explorer** and select the **BooksDB.bwp** process. Drag it to the **Process Editor** and drop it on the implemented POST operation.

A menu is displayed with two options: Create Invoke Activity and Create Reference and Wire Process.

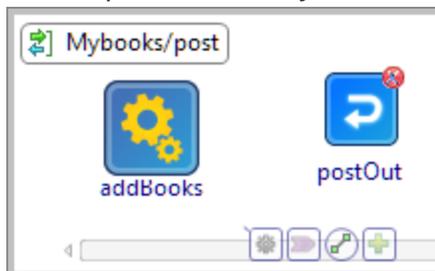


- Select **Create References and Wire Process**.

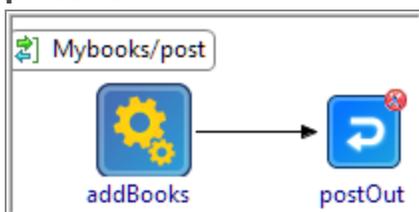
The references are added to the process. The purple chevron indicates the service and its operations that can be referenced by the process.



6. To update the POST process to invoke the appropriate external service operation:
 - a. Click the **addBooks** operation.
 - b. Select and drag the operation to the left of the **postOut** activity and drop it. An Invoke process activity is created.



7. Click the newly added activity. Select the  icon and connect **addBooks** to **postOut**.



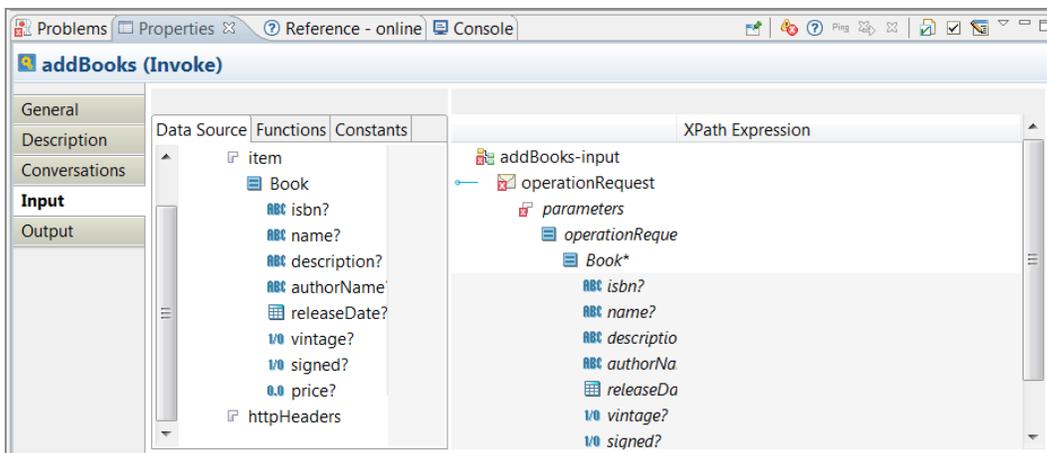
8. Click the **getAllBooks** operation and select, drag, and drop the operation to the left

of the **getOut** activity in the OUT process.

9. Connect **getAllBooks** to **getOut**.

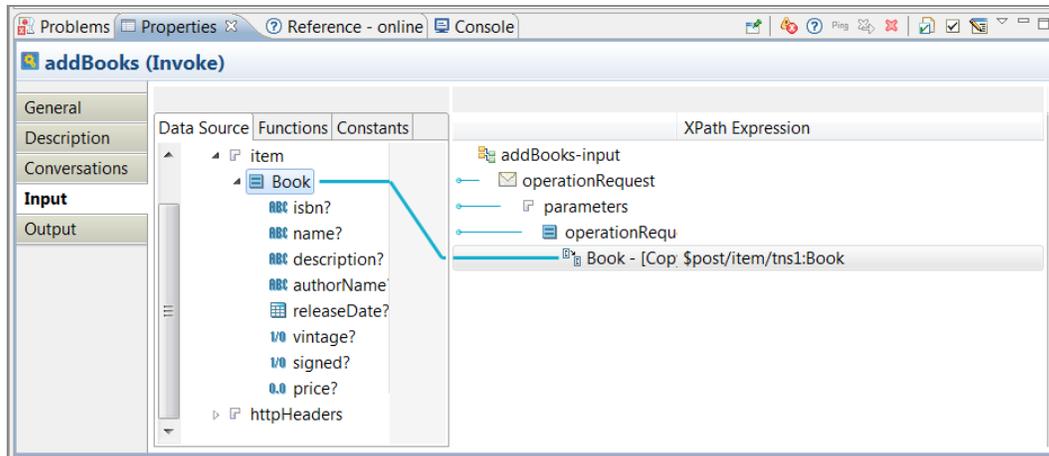


10. Save your changes.
11. Click the **addBooks** activity and select **Properties > Input**.
12. Expand the data tree in the **Data Source** pane to locate the Book element.

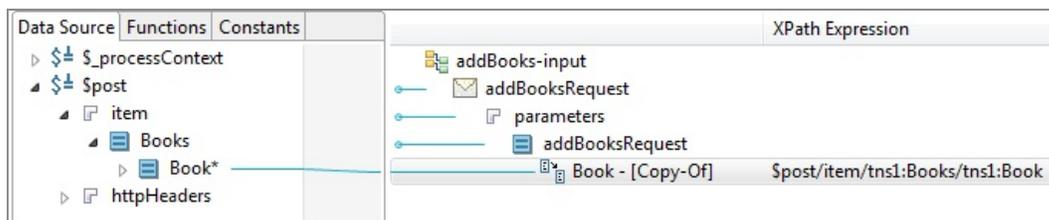


13. Drag the Book element from the left to the Book* element on the right.
14. In the pop-up window, select **Make a Copy of each "Book"** and click **Finish**.

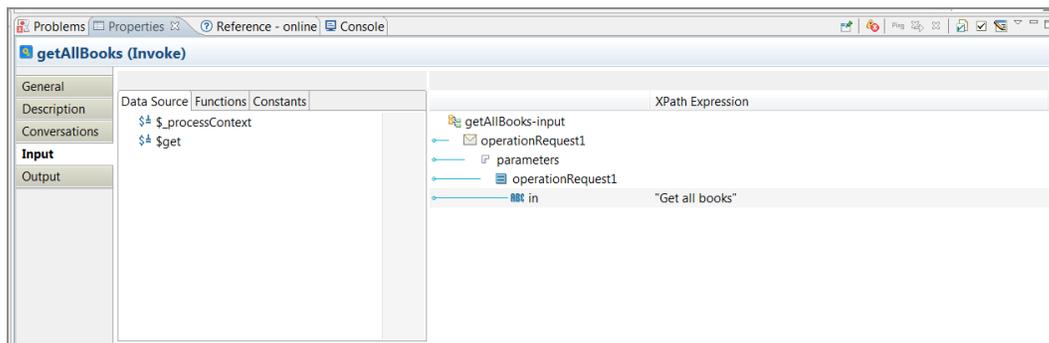
The **Input** tab looks like this:



15. Save your changes.
16. Click the **postOut** activity and open the **Properties > Input** tab. Expand the **post** activity and drag the **Book*** element from left to right.
17. In the pop-up window, select the **For each** option and click **Next**. Click **Finish** on the **Auto-Map** window. The **Properties > Input** tab looks similar to this:

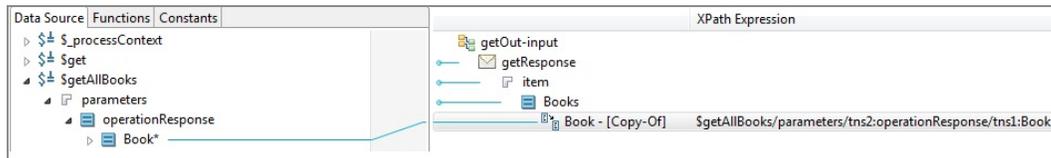


18. Click **getAllBooks** and select **Properties > Input**.
19. In the **XPath Expression** pane, add a dummy value to the input element, such as, "Get All Books". The input must be in quotes.



20. Click the **getOut** activity in the **Process Editor**, and select the **Properties > Input**

tab. Expand the **getAllBooks** activity and choose **Book*** to map the **Book*** element from left to right. In the pop-up window, choose **Make a Copy of each "Book"** and click **Finish**. The tab looks similar to this:



Result

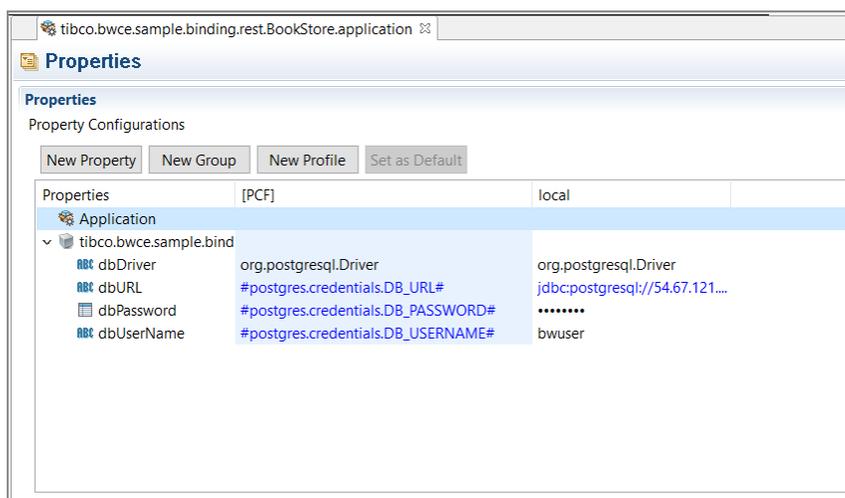
Your project is complete without any errors.

Testing the REST Service in Cloud Foundry

You can now test the REST service using the built-in tester and the Swagger UI.

Procedure

1. In the **Project Explorer**, expand the **tibco.bwce.sample.binding.rest.BookStore.application** process and expand the **Package Unit > Properties** folder.
2. In the **Properties** window, expand the **tibco.bwce.sample.binding.rest.BookStore.application** and set the default **Application Profile** to **PCF** as shown in the next image. The bracketed profile in the column head is the one that is selected:

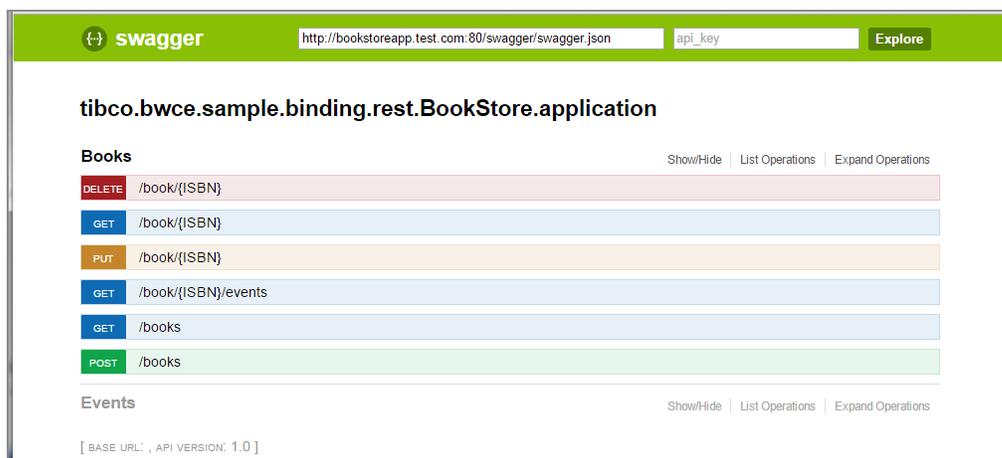


3. Expand the **Package Unit** and select **Overview**.
4. In the **Overview** window, select **Export Application for Deployment**.
5. Enter the location of your **EAR** file (this would be the same directory as the `manifest.yml` file).
6. In your Cloud Foundry environment, navigate to the directory containing your EAR.
7. Run

```
cf push -f manifest.yml
```

If the application deploys successfully, a routable URL is available.

8. Open the Google Chrome browser and open `http://<BWCE-APP-URL>/swagger`. Click **Books** or **Events** to see the operations. Click **MyBooks** to see the REST service operations you just added. See the section called [Testing the POST and GET Operations](#) for information.



9. Expand the Books and Events headers, and test out the operations as listed below.

Result

Click **Books** or **Events** in the Swagger UI to view the following operations for Books and Events:

Books

- Post books
- GET books

- GET book by ISBN
- PUT book by ISBN
- DELETE book by ISBN

Events

- POST Events
- GET Events
- GET Event by EventID
- PUT Event by EventID
- DELETE Event by EventID

GET books returns an output similar to the following:

```
{
  "Book": [
    {
      "isbn": "0061122416",
      "name": "The Alchemist",
      "description": "Every few decades a book is published that changes
the lives of its readers forever. The Alchemist is such a book",
      "authorName": "Paul Coelho",
      "releaseDate": "2006-04-25",
      "vintage": true,
      "signed": true,
      "price": 11.9
    },
    {
      "isbn": "0071450149",
      "name": "The Power to Predict",
      "description": "How Real Time Businesses Anticipate Customer
Needs, Create Opportunities, and Beat the Competition",
      "authorName": "Vivek Ranadive",
      "releaseDate": "2006-01-26",
      "vintage": false,
      "signed": true,
      "price": 15.999
    }
  ]
}
```

GET books by ISBN returns an output similar to the following for ISBN 0061122416:

```
{
  "isbn": "0061122416",
  "name": "The Alchemist",
  "description": "Every few decades a book is published that changes
the lives of its readers forever. The Alchemist is such a book",
  "authorName": "Paul Coelho",
  "releaseDate": "2006-04-25",
  "vintage": true,
  "signed": true,
  "price": 11.9
}
```

Testing the POST and GET Operations

An available RESTful service displays the GET operation in the Swagger UI. The POST operation is tested using the JSON service. It is important to test these operations by doing some simple tasks. This section explains how to test the POST and GET operations you just added.

Procedure

1. Click **MyBooks**. It expands and displays the POST and GET operations.
2. Click the **POST** icon to display its details.
3. Provide values for the Books parameter.
4. Click the **Try it out!** button.
5. Now click the **GET** icon to display its details.
6. Click the **Try it out!** button.

The response displays a list of books returned by the REST service from the database.



Testing the REST Service in OpenShift

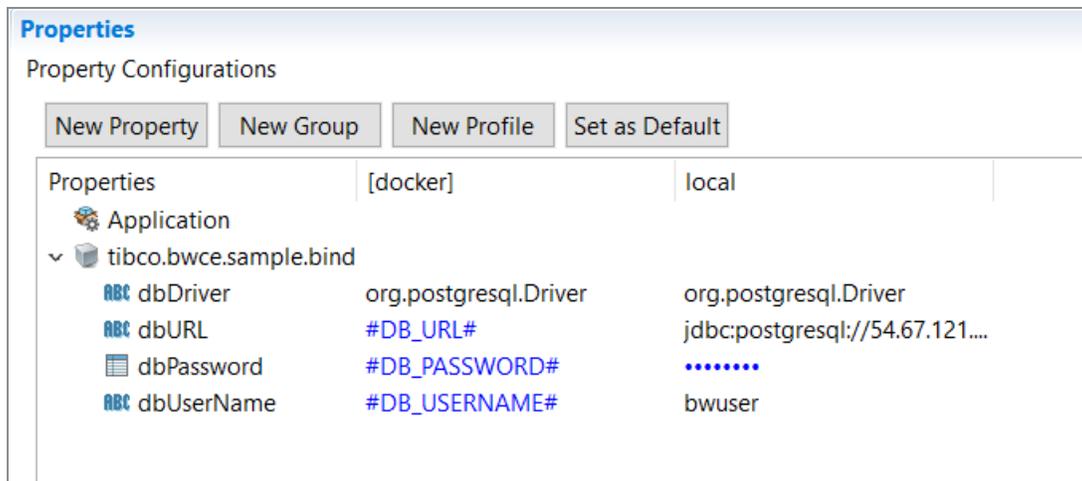
You can deploy the REST service in OpenShift environment and test by using the built-in tester and the Swagger UI.

Before you begin

Ensure that the OpenShift environment is set up to deploy the application.

Procedure

1. In the **Project Explorer**, expand the **tibco.bwce.sample.binding.rest.BookStore.application** process and expand the **Package Unit > Properties** folder.
2. In the **Properties** window, expand the **tibco.bwce.sample.binding.rest.BookStore.application** and set the default **Application Profile** to **Docker** as shown in the following image:



3. Expand the **Package Unit** and select **Overview**.
4. In the **Overview** window select **Export Application for Deployment**.
5. In your Docker environment, navigate to the directory containing your EAR file.
6. To login to the OpenShift environment, run the following command:

```
oc login <OpenShift_Server_URL>
```

7. If the project is not created, create an OpenShift project by running the following command:

```
oc new-project <PROJECT_NAME>
```

8. To build an application docker image, run the following command:

```
docker build -t <APPLICATION_NAME> .
```

9. To tag an application docker image with your docker repository, run the following command:

```
docker tag <APPLICATION_IMAGE_NAME><DOCKER_REPOSITORY_NAME>/<APPLICATION_IMAGE_NAME>
```

10. To push a docker image to the docker repository, run the following command.

```
docker push <DOCKER_REPOSITORY_NAME>/<APPLICATION_IMAGE_NAME>
```

i Note: Here the docker hub registry is used to push the docker image. You can push images to OpenShift internal registry or any other public registry.

11. To use the OpenShift configuration file (YAML or JSON) and push the application on OpenShift, run the following command.

```
oc create -f <MANIFEST_FILENAME>
```

```
apiVersion: v1
kind: Service
metadata:
  name: restapp
  labels:
    app: restapp
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: restapp
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: restapp
spec:
  replicas: 1
  selector:
    app: restapp
  template:
    metadata:
      name: restapp
      labels:
        app: restapp
    spec:
      containers:
        - name: restapp
          image: <DOCKER_REPOSITORY_NAME>/<APPLICATION_IMAGE_NAME>
          imagePullPolicy: Always
          env:
            - name: BW_LOGLEVEL
              value: ERROR
            - name: DB_URL
              value: <supported_database>://<username:password>@<machine:port>/<database_name>
            - name: DB_USERNAME
              value: <username>
            - name: DB_PASSWORD
              value: <password>
          ports:
            - containerPort: 8080
```

12. To create a route for application, run the following command.

```
oc expose service <APPLICATION_SERVICE_NAME>
```

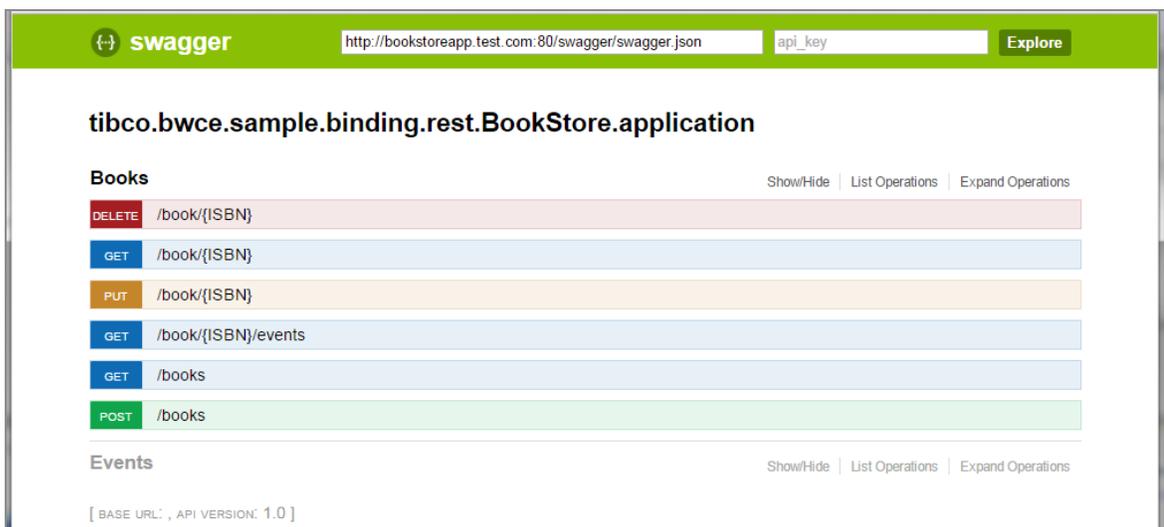
```
C:\Users\rubirada\Downloads>oc expose service restboostore
route "restboostore" exposed
```

13. To view the route of an application, run the following command:

```
oc get route
```

```
C:\Users\rubirada\Downloads>oc get route
NAME          HOST/PORT          PATH          SERVICES          PORT          TERMINATION  WILDCARD
jms-consul    jms-consul-test.os.tibcopcf110.com  PATH          jms-consul        8080          None         None
restboostore  restboostore-test.os.tibcopcf110.com  restboostore  app                app           None         None
```

14. Open the Google Chrome browser and open `http://ROUTABLE_URL/swagger`.



15. In the Swagger UI, click **Books** or **Events** to see the operations and then click **MyBooks** to see the REST service operations you just added. For more information, see [Testing the POST and GET Operations](#).
16. Expand the Books and Events headers, and test the operations as follows.

Result

In the Swagger UI, click **Books** or **Events** to view the following operations for the Books and Events headers:

Books

- POST books
- GET books
- GET book by ISBN
- PUT book by ISBN
- DELETE book by ISBN

Events

- POST Events
- GET Events
- GET Event by EventID
- PUT Event by EventID
- DELETE Event by EventID

GET books returns the following similar output:

```
{
  "Book": [
    {
      "isbn": "0061122416",
      "name": "The Alchemist",
      "description": "Every few decades a book is published that changes
the lives of its readers forever. The Alchemist is such a book",
      "authorName": "Paul Coelho",
      "releaseDate": "2006-04-25",
      "vintage": true,
      "signed": true,
      "price": 11.9
    },
    {
      "isbn": "0071450149",
      "name": "The Power to Predict",
      "description": "How Real Time Businesses Anticipate Customer
Needs, Create Opportunities, and Beat the Competition",
      "authorName": "Vivek Ranadive",
      "releaseDate": "2006-01-26",
      "vintage": false,
      "signed": true,
      "price": 15.999
    }
  ]
}
```

```
]
}
```

GET books by ISBN returns the following similar output for ISBN 0061122416:

```
{
  "isbn": "0061122416",
  "name": "The Alchemist",
  "description": "Every few decades a book is published that changes
the lives of its readers forever. The Alchemist is such a book",
  "authorName": "Paul Coelho",
  "releaseDate": "2006-04-25",
  "vintage": true,
  "signed": true,
  "price": 11.9
}
```

Troubleshooting

You may encounter some errors while executing or running the process. The following are some of the possible errors you may encounter and their resolutions.

| Error Encountered | Resolution |
|---|---|
| The REST Swagger UI page is not visible. | Verify that the application has started and that you are accessing the correct URL. |
| Problem markers are visible in the project. | Clean the project by invoking Project > Clean or by switching to a clean new workspace. |
| The database and database tables are not created. | Ensure that the Cloud Foundry PostgreSQLService is configured correctly. |

REST Reference Tutorial

The REST reference tutorial shows you how to create a simple REST Invoke to an existing REST Service defined by a Swagger specification.

You cannot convert REST reference to SOAP or vice versa.

Before you begin

The REST service which you want to consume has to be deployed in your Cloud Foundry environment.

Creating a New Application

1. Open TIBCO Business Studio for BusinessWorks.
2. Open the **Design** perspective by clicking the **Design** icon in the upper right corner.
3. Click **File > New > Other > BusinessWorks > BusinessWorks Application Module** and click **Next**.
4. Enter in the **Project Name** text box. Do not change the remaining default settings. Click **Finish**. This creates a new application module with an empty process.

Obtaining the REST Service Swagger File

 **Note:** The REST service for which you want to obtain the Swagger file must be running. In this example the **tibco.bwce.sample.binding.rest.BookStore.application** should be deployed in your Cloud Foundry environment.

To obtain a Swagger file for a REST service, do the following:

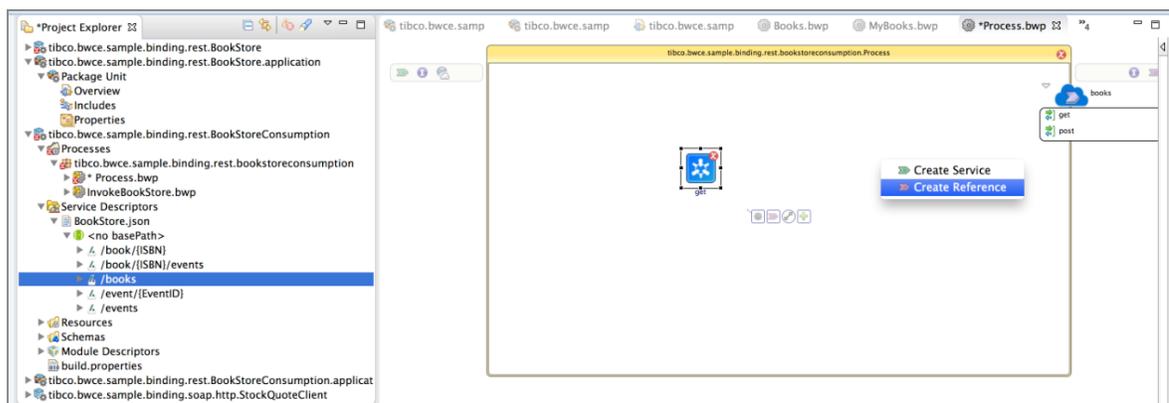
1. Append `swagger` to the routable URL to access the Swagger doc. The format is `http://<routable url>/swagger`.
2. Enter `http://<routable url>/swagger/swagger.json` in the browser and copy entire JSON file to `Books.json` file.

Importing the JSON File into your Project

1. In the **Project Explorer**, expand `tibco_bwce_sample_binding_rest` application module.
2. Right-click **Service Descriptors** and select **Import > Import... > General > File System** and click **Next**.
3. In the File system dialog, click the **Browse** button and browse to the location of the `Books.json` file.
4. Select the checkbox next to **Books.json** in the left pane and click **Finish**.

Creating the REST Reference

1. In the **Project Explorer**, expand the `tibco_bwce_sample_binding_rest` **Service Descriptors** folder completely .
2. Select the `/books` under **Books.json** and drag and drop it to the right side of the process in the Process Editor. The references are added to the process. The purple chevron indicates the service and its operations.



3. In the **Process Editor**, right-click **Add Activity > General Activities > Timer**. Optionally, you can configure the **Sleep** activity with **IntervalInMillisec** as 3000 in a similar manner and connect the **Timer** with **Sleep**.
4. Drag the **get** operation under the purple chevron and drop it on the right of **Timer** activity (or **Sleep** if configured) and connect the **Timer** activity with the **get** activity.
5. Drag the **post** operation under the purple chevron and drop it on the right of the **get** activity , connect the **get** activity with the **post** activity .
6. Right-click the **get** activity select **Show Properties View**.

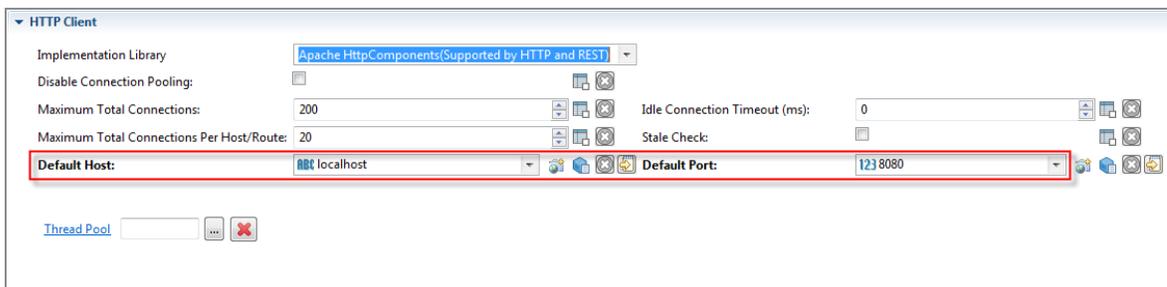
7. In the **Properties** view, select the **Input** tab and click **Show Check and Repair** icon in the icon bar on the upper right corner of the Properties view.
8. Select the checkbox under **Fix** and click **OK**.
9. Click **Show Check and Repair** icon again. Select the checkbox under **Fix** and click **OK**.
10. Select the **post** activity and right click and select **Show Properties View**. In the **Properties View**, select the **Input** tab and select **Data Source** tab.
11. Expand **\$getin** in the **Data Source** tab completely.
12. In the XPath Expression pane, expand the **post-input** completely.
13. Drag and drop **Book*** from the **Data Source** tab to the **Book*** under post-input in the **XPath Expression** pane.
14. In the Drop dialog, select **Make a copy of each "book"** radio button and click **Finish**.
15. Click **Show Check and Repair** icon in the icon bar on the upper right corner of the Properties view.
16. Select the checkbox under **Fix** and click **OK**.
17. Click **Show Check and Repair** icon again. Select the checkbox under **Fix** and click **OK**.
18. In the **Project Explorer**, select **Books.json** under **Service Descriptors** of **tibco_bwce_sample_binding_rest_basic** application module, and right click **Open With > Text Editor** and locate the "host" attribute. Make a note of the host name and port number.

```

{
  "swagger": "2.0",
  "info": {
    "version": "1.0",
    "title": "Summary about the new REST service.",
    "description": "Summary about the new REST service."
  },
  "host": "localhost:8080",
  "basePath": "/",
  "schemes": [ "http" ],
  "consumes": [ "application/json" ],
  "produces": [ "application/json" ],
  "paths": {
    "/books": {
      "post": {
        "description": "",
        "operationId": "postbooks",
        "consumes": [ "application/json" ],
        "produces": [ "application/json" ],
        "parameters": [ {
          "name": "body",
          "in": "body",
          "description": "",
          "schema": {
            "$ref": "#/definitions/Books"
          }
        } ],
        "required": true,
        "allowMultiple": false
      },
      "responses": {
        "200": {
          "description": "a Books to be returned",
          "schema": {
            "$ref": "#/definitions/Books"
          }
        }
      }
    }
  }
}

```

19. Expand the Resources folder under the tibco_bwce_sample_binding_rest_basic application module completely.
20. Double-click **HttpClientResource.httpClientResource**.
21. In the HTTP Client section, change the Default Host and Default Port to the values in the Books.json file you obtained in step 18 above.



22. Click **File > Save All**.

Testing the REST Reference

You can now test the REST service using the built-in tester and the Swagger UI. To do so follow these steps:

1. Click **Run > Debug Configuration**.
2. In the left pane of the **Debug Configuration** wizard, expand **BusinessWorks Application** and select **BWApplication**.

3. Click the **Applications** tab and then click **Deselect All** if you have multiple applications. Select the checkboxes next to **tibco_bw_sample_binding_rest_basic_application**.
4. Click **Debug**. This runs the sample in debug mode. The Console view is opened and shows engine messages similar to: Started BW Application [tibco_bwce_sample_binding_rest_basic_application:1.0]
5. In the **Debug** view, expand **BWApplication [BusinessWorks Application] > <launched> BWEclipseAppNode > tibco_bwce_sample_binding_rest_Process** and select **get**.
6. In the **JobData** view, you can see the job data of the **get** activity.

The screenshot shows the TIBCO BusinessWorks IDE interface. On the left, the 'Debug' view displays a tree structure of the application components. The 'get' activity is selected and highlighted in blue. On the right, the 'Job Data' view shows the data for the selected 'get' activity, including input, output, and fault information.

| Input | Name | Value |
|--------|-----------------------|--|
| Output | get-input.httpHeaders | <tnsl:httpHeaders xmlns:tnsl="http://tns.tibco.com/bw/ |
| Fault | Accept | |
| All | Accept-Charset | |
| | Accept-Encoding | |
| | Content-Type | |
| | Content-Length | |
| | Connection | |
| | Cookie | |

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 Maven Plug-in*
- *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

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

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

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

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

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

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>.

Copyright © 2015-2024. Cloud Software Group, Inc. All Rights Reserved.