# TIBCO ActiveMatrix BusinessWorks™

## Getting Started

Version 6.11.0 | December 2024

# Contents

# 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 ActiveMatrix BusinessWorks™ 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:

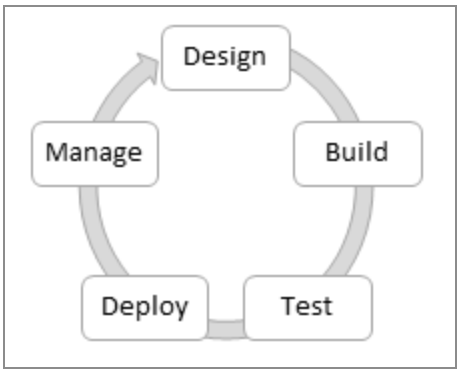
- File Poller Module Tutorial: Guides you through creating a simple process and running it.
- REST Service Tutorial: Walks you through the steps to build and test a simple REST Service using TIBCO Business Studio™ for BusinessWorks™ and the Swagger UI.
- REST Reference Tutorial: Shows you how to create a simple REST Invoke to an existing REST Service defined by a Swagger specification.
- Administration Tutorial: Provides information about the administration framework and deploying sample applications using the provided scripts.

The Orientation section introduces you to the TIBCO Business Studio for BusinessWorks development environment. Before you continue, read the *ActiveMatrix BusinessWorks Concepts* guide to familiarize yourself with the ActiveMatrix BusinessWorks concepts and terminology.

# Orientation

ActiveMatrix BusinessWorks 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 ActiveMatrix BusinessWorks™ 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.

# 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.

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 about samples in TIBCO Business Studio for BusinessWorks, see the section Accessing Samples in the *TIBCO ActiveMatrix BusinessWorks™ Samples* 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

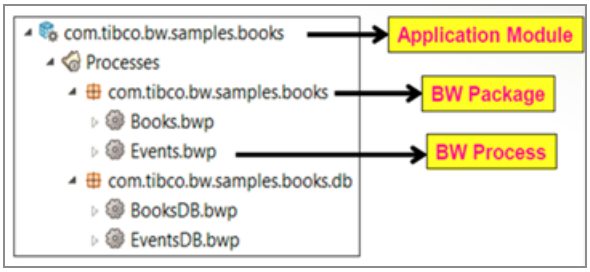
For more information about an application's integration style and other application design considerations, see "Application Design Considerations" in the *ActiveMatrix BusinessWorks Application Development* guide.

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 *ActiveMatrix BusinessWorks 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.




**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 ActiveMatrix BusinessWorks.

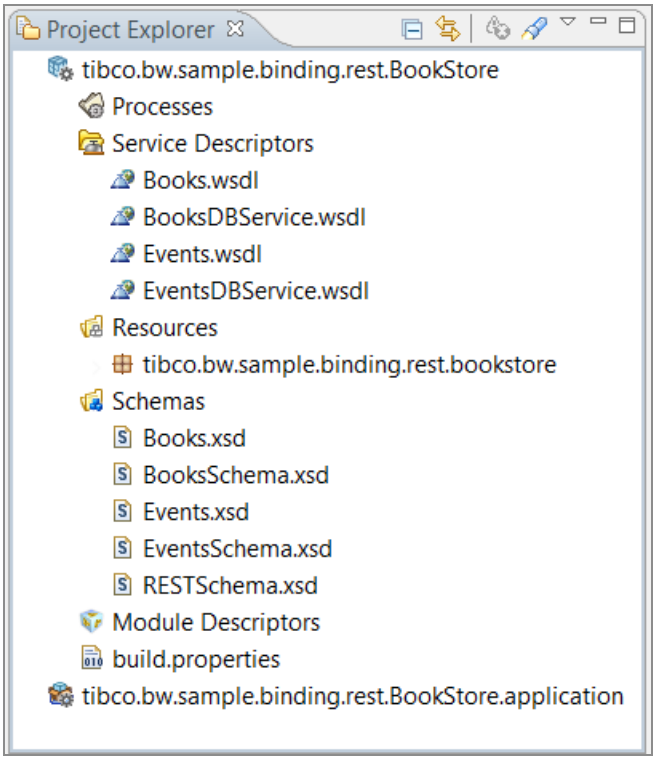
Select a WSDL file in the **Project Explorer** and drop it on the **Process Editor** to implement a web service. Dropping the WSDL file displays a menu for creating services or implementing operations. Response activities are automatically generated.

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.

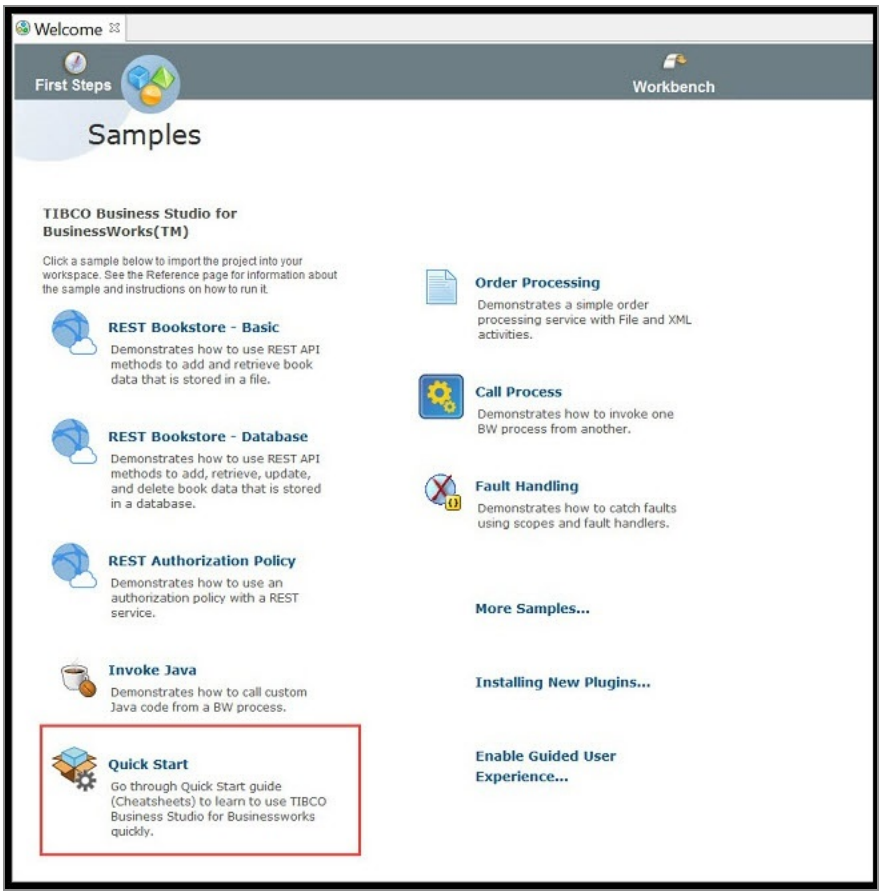
- **Resources**: Contains shared resources used by activities in a process.
- **Schemas**: Stores XSD (schema) files.
- **Service Descriptors**: Stores WSDL and JSON files.

# 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 ActiveMatrix BusinessWorks™ in an existing workspace, open the Welcome page by using **Help > Welcome**.




- Click **Help > Cheat Sheets**.

Cheat sheets are available for the following tasks:

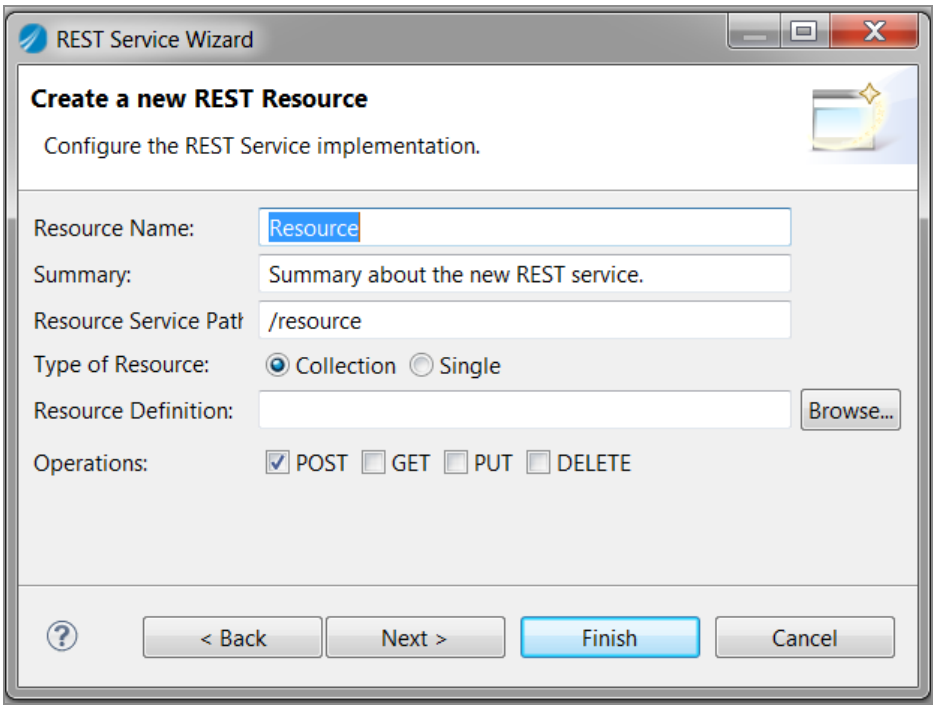
- Adding unit testing to a project.
- Creating a basic SOAP application.
- Creating an application using Java Invoke.
- Configuring a JMS connection and creating a JMS send message application.
- Configuring a JBDC connection using custom driver.
- Developing a REST API to expose a SOAP service.
- Exposing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

# REST Support

The REST Service wizard is used to build RESTful services.

> **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:

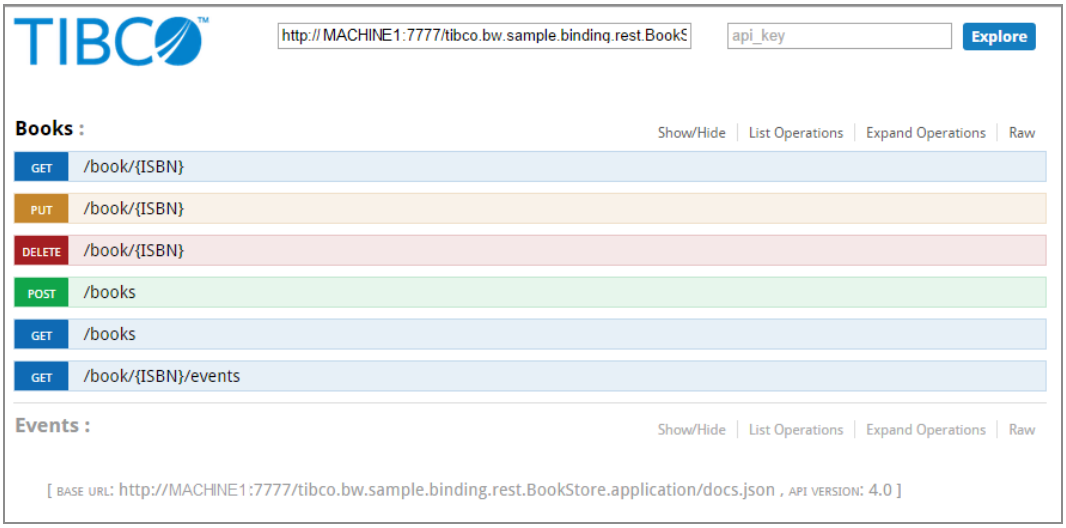
1. Name the REST resource.
2. Choose the resource definition (the XSD schema).
3. Choose the REST operations to implement.

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. 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.

# Creating REST Services in the Rest Service Wizard

You use the REST Service Wizard to create a RESTful service or simply drag and drop a process to the left of the **Process Editor** to create a REST service.

> **Note:** When you create a REST service, make sure to update 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`.

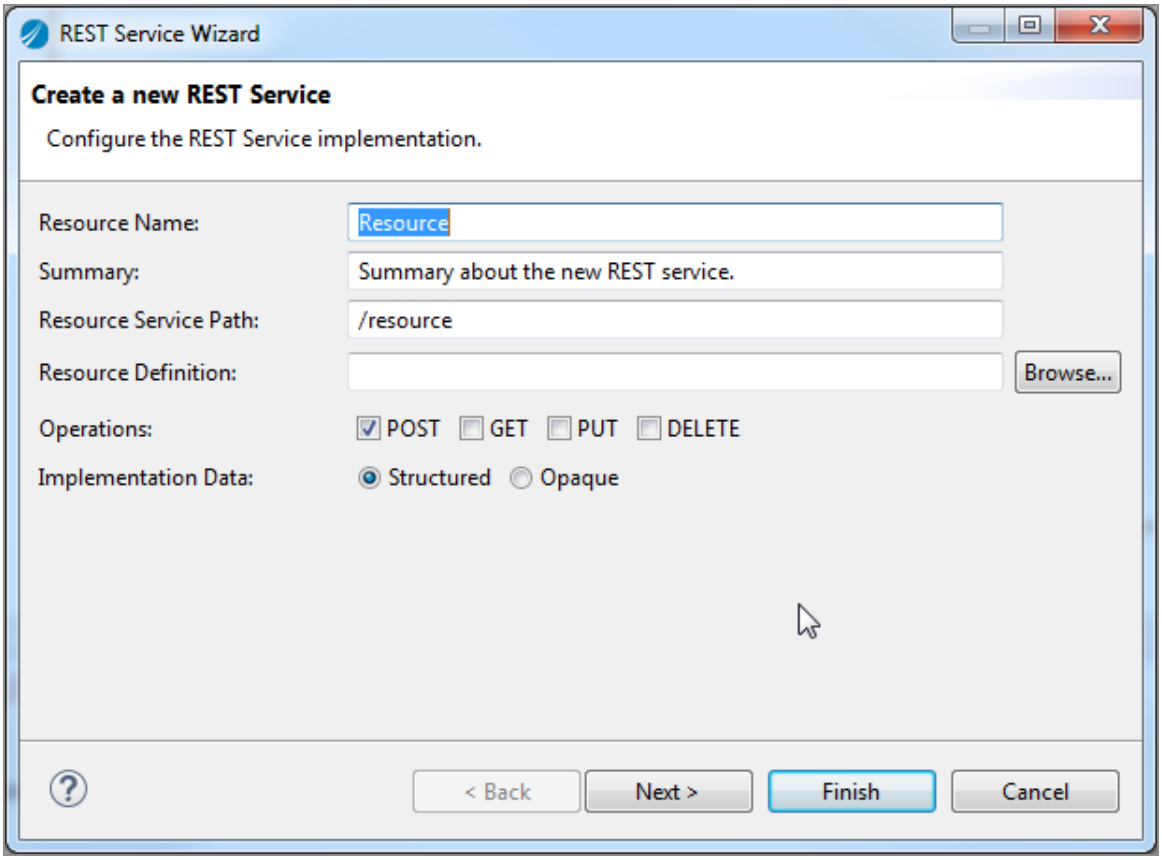
Follow these steps to create a REST service using the REST Service Wizard:

## Procedure

1. Right-click on a process in the **Project Explorer** and select **New > BusinessWorks REST Resource**.

   The REST Service Wizard opens.

   REST Service Wizard




2. Specify a name for the REST resource.
3. Choose the resource definition (the XSD schema).
4. Select the REST operations to implement.
5. Click **Next** to configure the operations or click **Finish**.

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

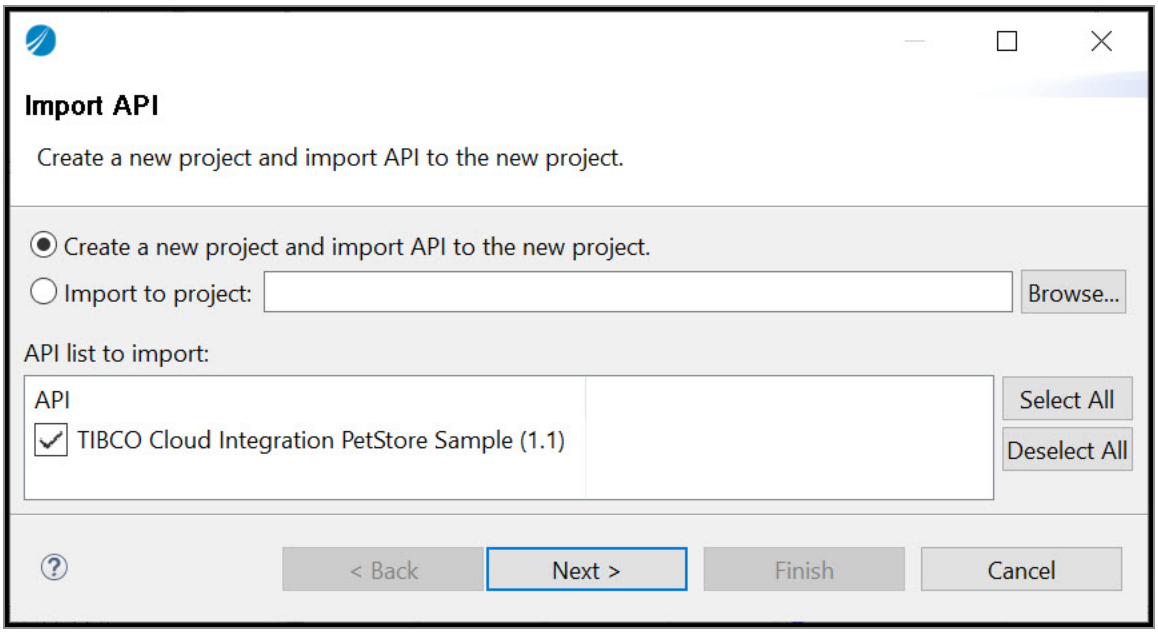
# Importing an API Model into your Workspace

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

### Procedure

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

   The Import API dialog opens.

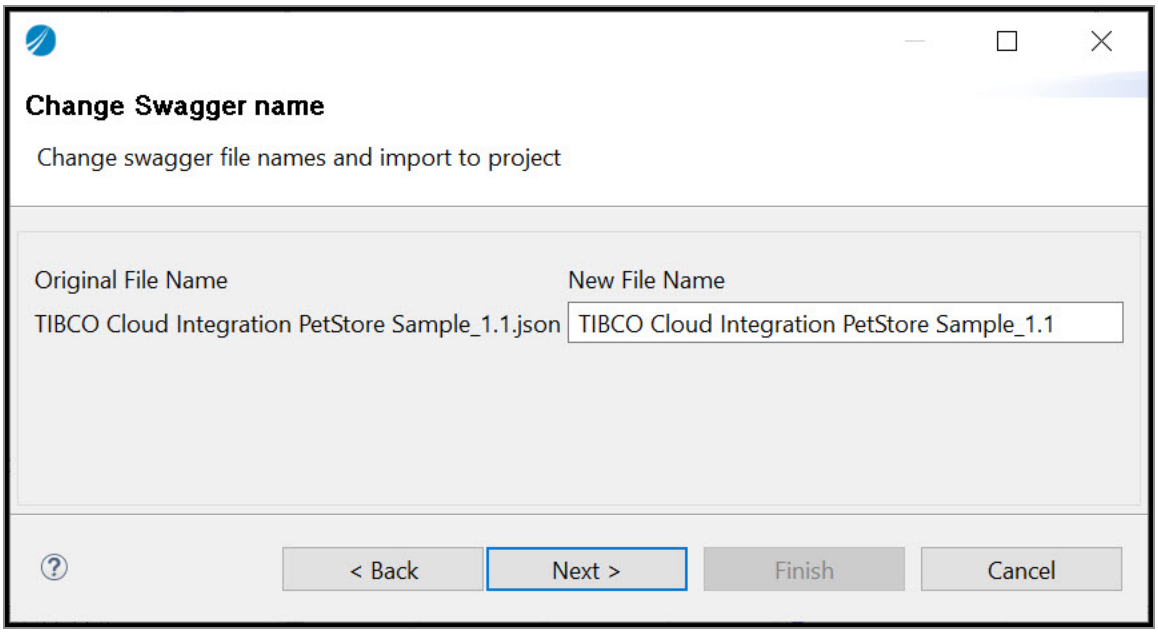



Every API you selected in the **API Explorer** is listed in this dialog. If an API has multiple versions, all versions are listed. By default, all APIs listed here are selected. You can deselect APIs that you do not want to import by clearing its checkbox.

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. ActiveMatrix 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. 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 needs 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 it 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

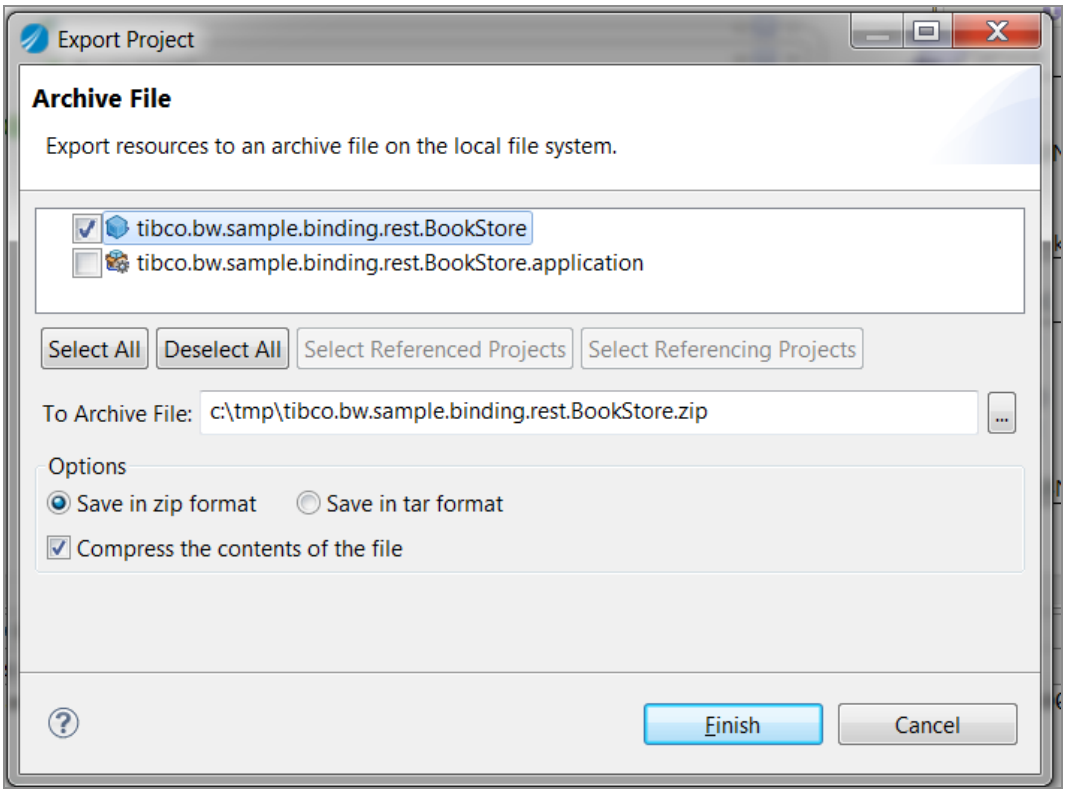
After completing an application module, you must define an application to build a deployment archive file. An application defines all the processes, properties, and resources

that must be included in the archive file. By default, all processes are included.

To create an archive, choose one of the following:

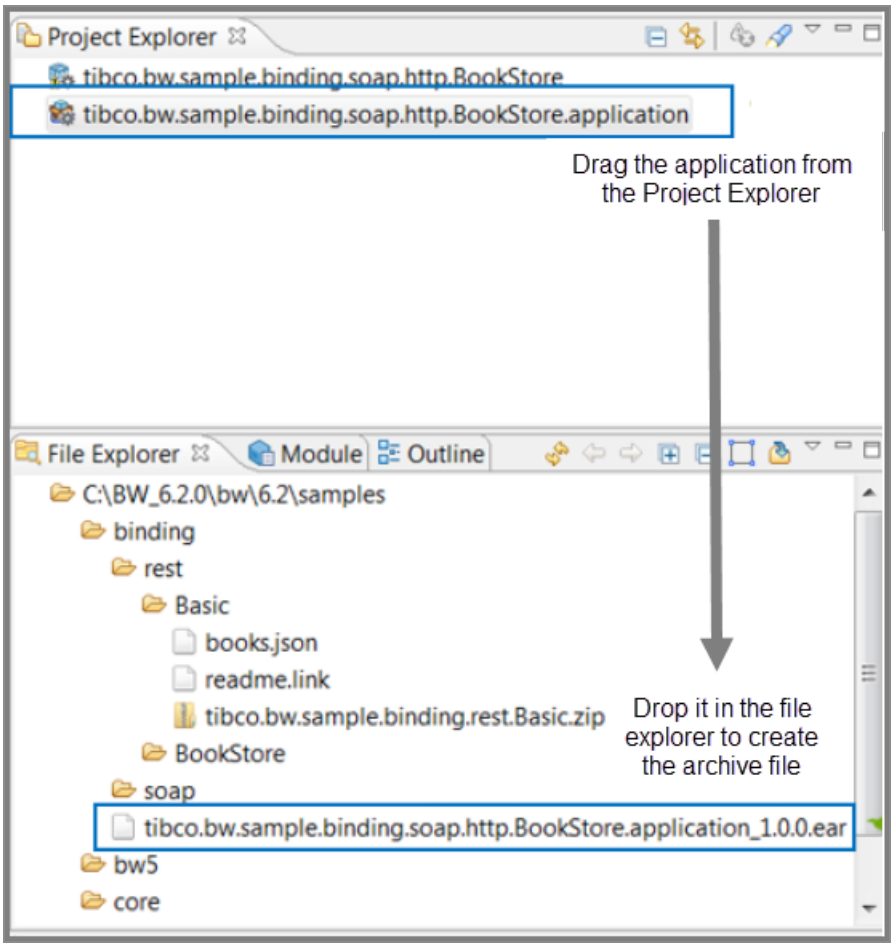
- Right-click the project in the **Project Explorer** and choose **Export > Studio Projects to Archive**. The **Export Project** dialog is displayed.

  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

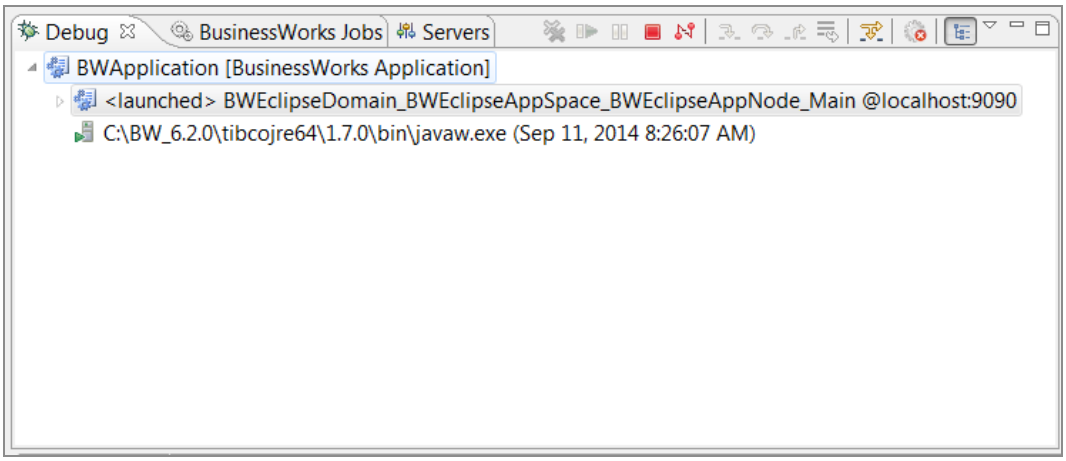



In both scenarios, the archive file is created with all required processes, properties, and resources. In the first scenario, you can name the archive file, select the format, and select the resources to include. In the second scenario, the archive is created for you in the format appropriate for your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

# Debugger

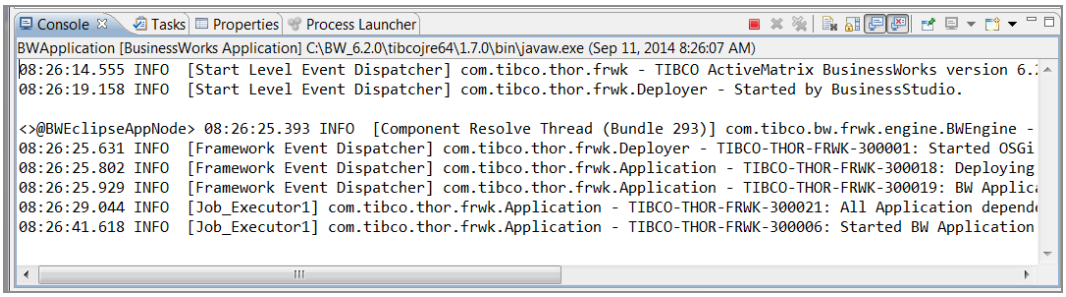
The TIBCO Business Studio for BusinessWorks debugger is used to test processes during the process development stage. By default, the debugger supports all the processes and subprocesses of an application module, shared module and nested shared module. Starting the debugger brings up the **Debug** perspective. This perspective can be used to set breakpoints, step through processes, examine job variables, and activity input/output at each step.

Debug Perspective



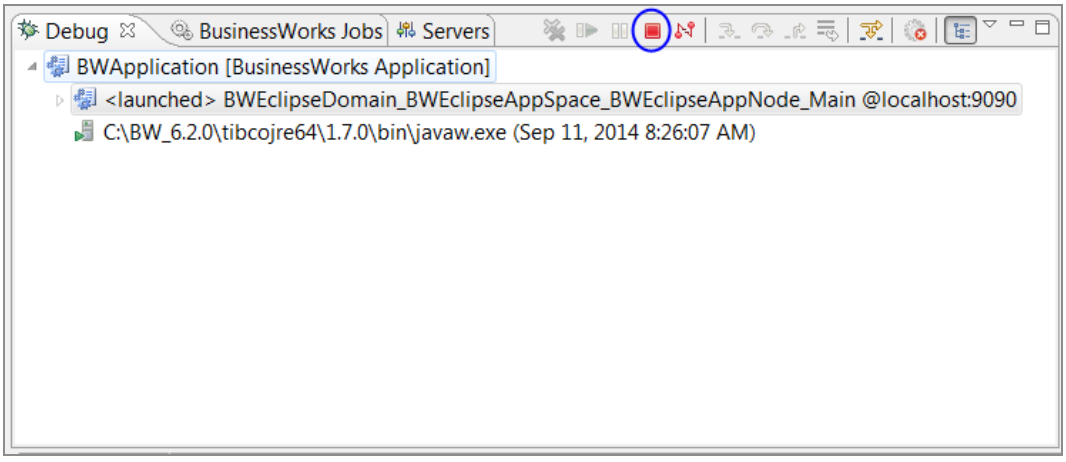

The **Console** view displays the messages and errors returned by the runtime.

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



# Run Time

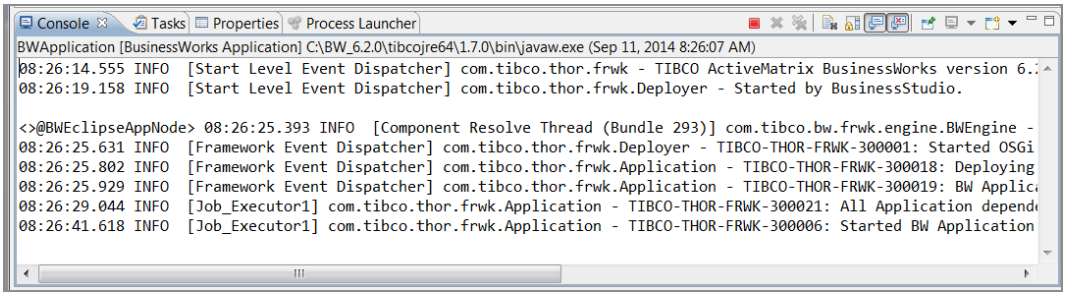
You can run applications in TIBCO Business Studio for BusinessWorks and test them in a runtime environment, which consists of a domain, AppSpace, and an AppNode on your local machine.

These runtime entities were created when you installed ActiveMatrix BusinessWorks. For more information about runtime entities, see the *ActiveMatrix BusinessWorks Concepts* guide.

For information about the administration framework, see the *ActiveMatrix BusinessWorks Administration* 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 allows 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.
```

**Note:** Applications can also be run from the administration framework using the bwadmin command-line utility or the Admin UI. For information about the administration framework, see the *ActiveMatrix BusinessWorks Administration* guide.

# Deployment

Applications can be deployed from TIBCO Business Studio for BusinessWorks using either the **Deployment Wizard** or the **Deployment Server**.

The **Deployment Wizard** is available from the right-click menu once an archive file has been created. You need to provide the name and port for the network to which you want to deploy to, as well as the domain and AppSpace for deployment.

The Deployment Server is a pre-built deployment environment that consists of domains, AppSpaces, and AppNodes. Archive files have been uploaded to this server and applications are ready for deployment.

Network configuration for either a local network or the deployment server is available from the **Deploy view** in TIBCO Business Studio for BusinessWorks.

For more information about deployment in TIBCO Business Studio for BusinessWorks, see the*ActiveMatrix BusinessWorks Application Development* guide.

**Note:** Applications can also be deployed from the administration framework using the bwadmin command line utility or the Admin UI. For information about the administration framework, see the *ActiveMatrix BusinessWorks Administration* guide.

# File Poller Tutorial

This tutorial guides you through creating a File Poller project. This simple project can perform many of the same tasks as those required for a project with a larger scope and more complexity. The File Poller project polls a directory for a specified file and writes a new file to the same directory each time the file is modified.

This tutorial involves:

- Creating a New Project
- Configuring the File Poller Project
- Testing the FilePoller Application in the Debugger

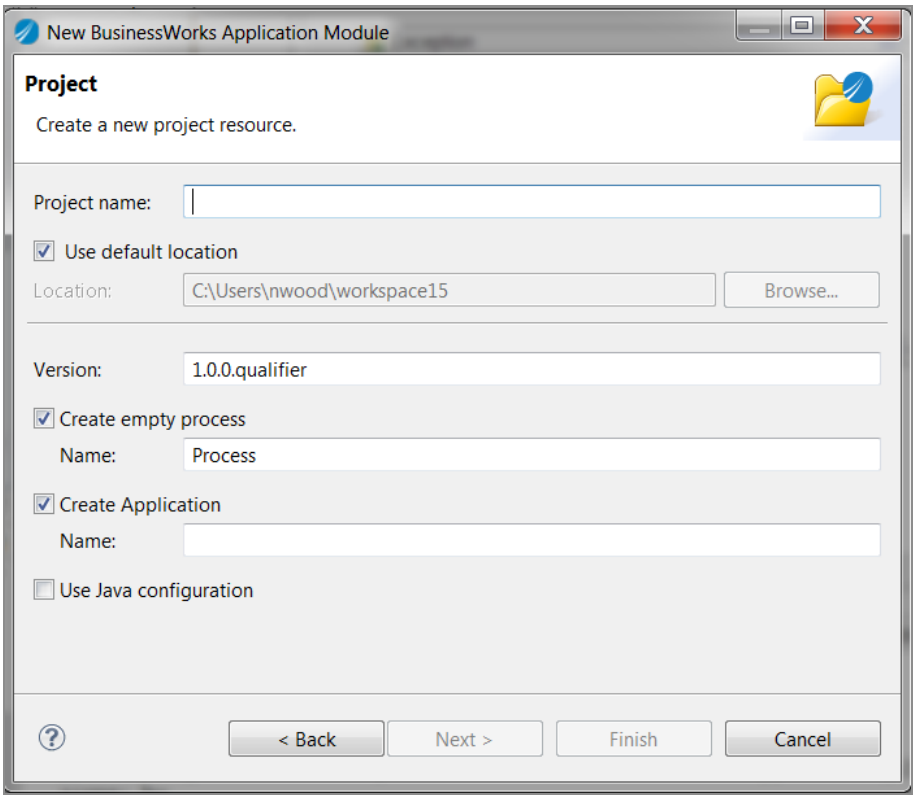
# Creating a New Project

This section guides you through creating a simple project.

### Procedure

1. Start TIBCO Business Studio for BusinessWorks:
    a. On Unix: Select the TIBCO Business Studio for BusinessWorks executable located at `$TIBCO_HOME/studio/version/eclipse/`
    b. On Windows: **Start > All Programs > TIBCO > *TIBCO_HOME* > TIBCO Business Studio for Designers**
2. Open the BusinessWorks Application Module wizard from **File > New > BusinessWorks Resources**.

    The BusinessWorks Resource Wizard opens.
3. In the Select a wizard dialog, select **BusinessWorks Application Module**, and click **Next**. The wizard is displayed:



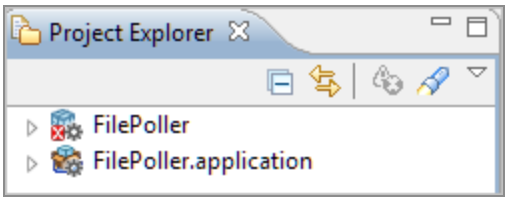

4. In the **Project name** field, enter FilePoller

   Keep the **Use default location**, **Create empty process**, and **Create Application** checkboxes selected.

5. Click **Finish**.

## Result

Two folders are created and are visible in the **Project Explorer**, one for the application and one for the application module.



# Configuring the FilePoller Project

The FilePoller project uses a **File Poller** activity and a **Write File** activity. This project creates a simple process that points to a specified file. The file is polled periodically to determine whether it was changed. The changed file comprises the text written to a new file.
The **File Poller** and **Write File** activities in the **File** palette are used in this process.

### Before you begin

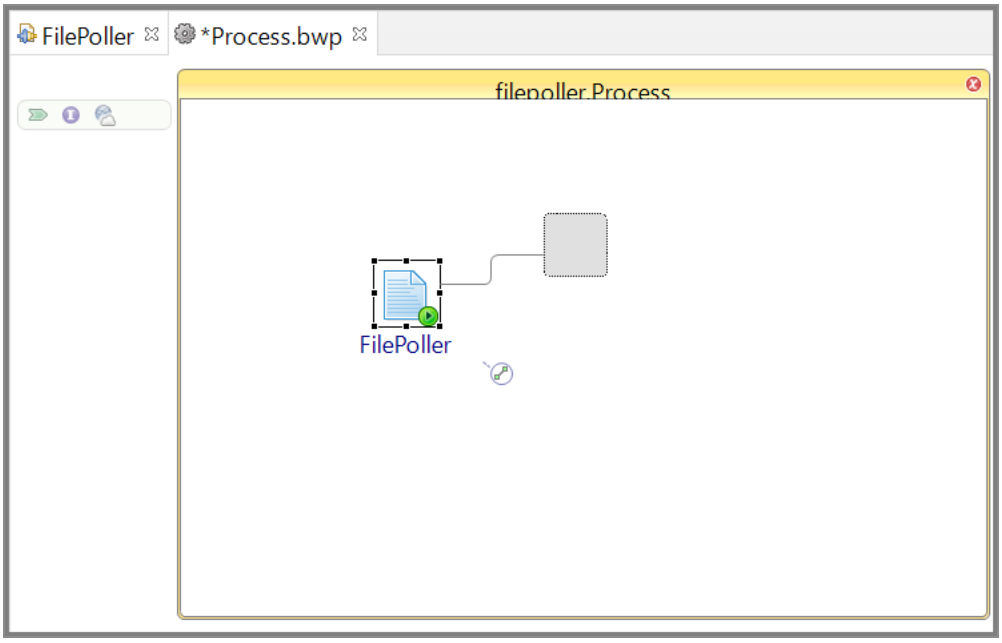
A text file is required, for example, `c:\tmp\fileread.txt`. Type a few lines in the file and save it.

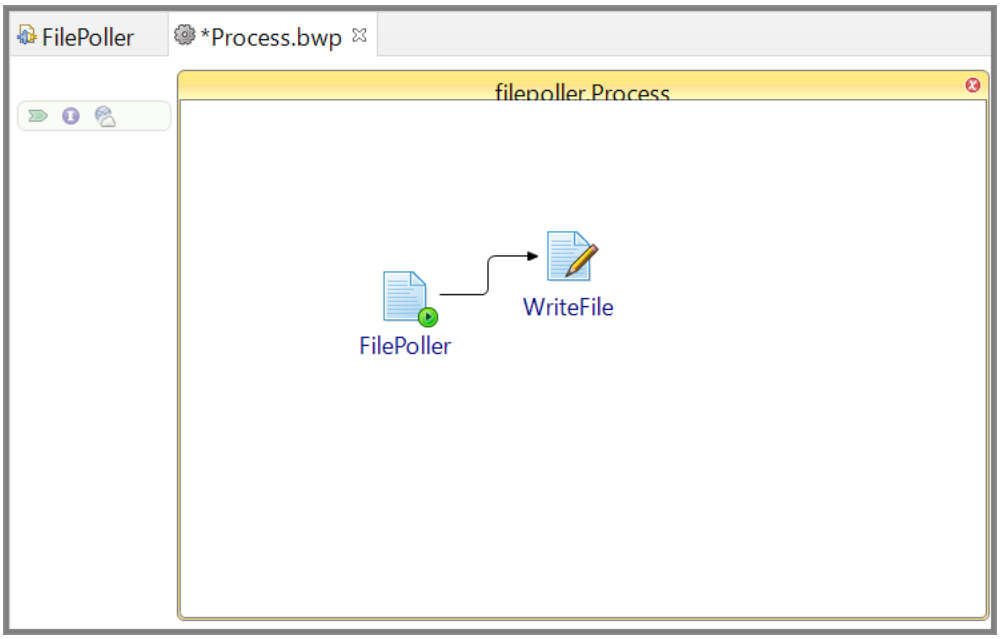
### Procedure

1. Select and drop a **File Poller** activity from the **File** palette to the **Process Editor** window.

   To add an activity to the **Process Editor**, click the activity and drop it on the **Process Editor**. Do not drag and drop the activity.

2. Select the **Write File** activity from the **File** palette. Click in the **Process Editor** next to the **File Poller** activity. You see a gray overlay indicating where you can place the activity, along with the transition arrow. When you drop the activity, the transition from the **File Poller** activity to the **Write File** activity is created.

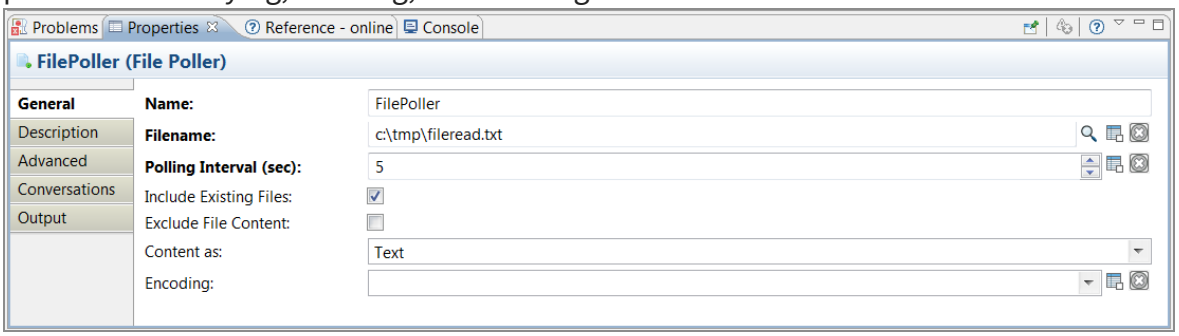





**Note:** To create a transition between two activities click the transition icon and join both the activities in the Process Editor.

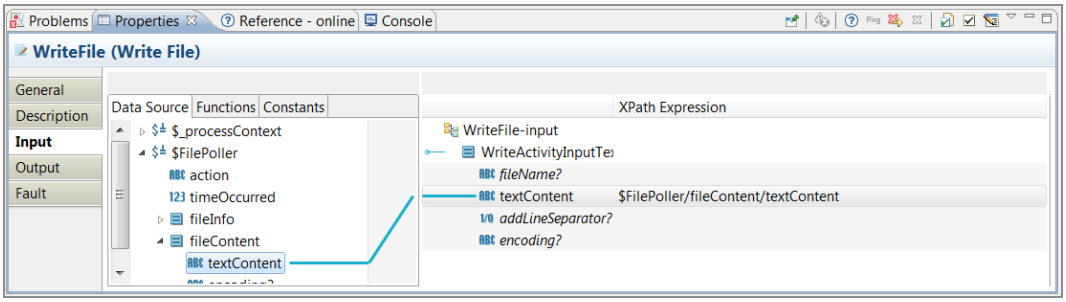
3. Select the **File Poller** activity. In the **Properties** tab, select the **General** tab. Point to the file you created as a prerequisite to this tutorial.

4. Select the **Include Existing Files** checkbox. When selected, the **File Poller** activity polls the existing file regardless of the changes made. The specified file is periodically polled at the specified interval even if the file has not changed. These changes may pertain to modifying, creating, or removing the file.



> **Note:** The Polling Interval (sec): 5 (default) indicates the Frequency with which the File Poller activity monitors this particular file. Any update to this file is transferred to the `output.log` file through the **Write File** activity.

5. Save your project. Click **File > Save** or the **Save** button on the tool bar.
6. Select the **Write File** activity and click the **General** tab.
7. Click the  button in the **Filename** field and specify the output location, such as, `c:\tmp\FilePoller\output.log`. Also, select the **Create Non-Existing Directories** checkbox.
8. Click the **Input** tab to specify input to the **Write File** activity. Drag the `FilePoller\fileContent\textContent` from the **Data Source** pane into the `textContent` field in the **XPath Expression** pane. This writes the content of the polled file to the output file.




9. Save your project.
10. Test this project in the debugger. For more information about the testing procedure, see Testing the File Poller Application in Debugger.

## Result

The **File Poller** activity polls the `fileread.txt` file located in `c:\tmp` file every 5 seconds and any changes made to the file content are written to the `output.log` file at `c:\tmp\FilePoller` by the **Write File** activity.

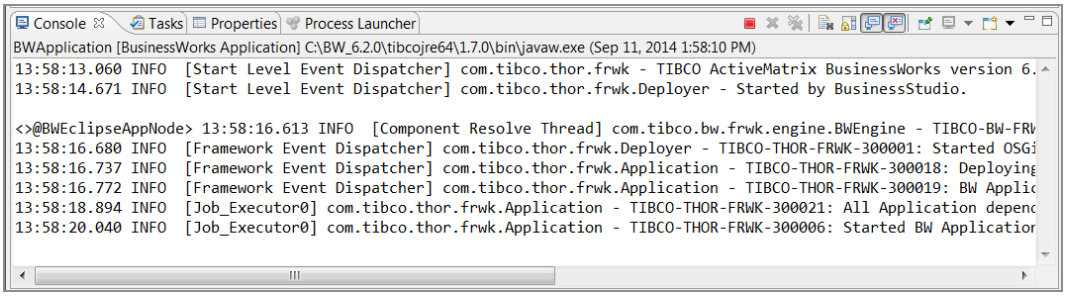
# Testing the FilePoller Application in the Debugger

The debugger provides a simple and fast way of debugging one or more ActiveMatrix BusinessWorks applications in a local runtime environment. The applications must be in the workspace and selected before launching the debugger. After starting the debugging session, the debugger does not provide tooling support for deploying and debugging the application on the same runtime instance. The runtime starts when the debugger is started and stops when the debugger is stopped.

### Procedure

1. Right-click in the **Process Editor** and select the **Debug BusinessWorks Applications** option from the menu. You can also click  to start the debugger or choose **Run > Debug**.

   The following messages are displayed in the **Console view**.

   You are running the instance of the FilePoller application. When the debugger is launched, the perspective changes from **Modeling** to **Debug**.

2. After the application starts, the **File Poller** and **Write File** activities get triggered and the content of the `fileread.txt` is written to `c:\tmp\FilePoller\output.log`.

3. Modify the `c:\tmp\fileread.txt` file. For example, open the `c:\tmp\fileread.txt` file and write Hello BusinessWorks! and save the changes.

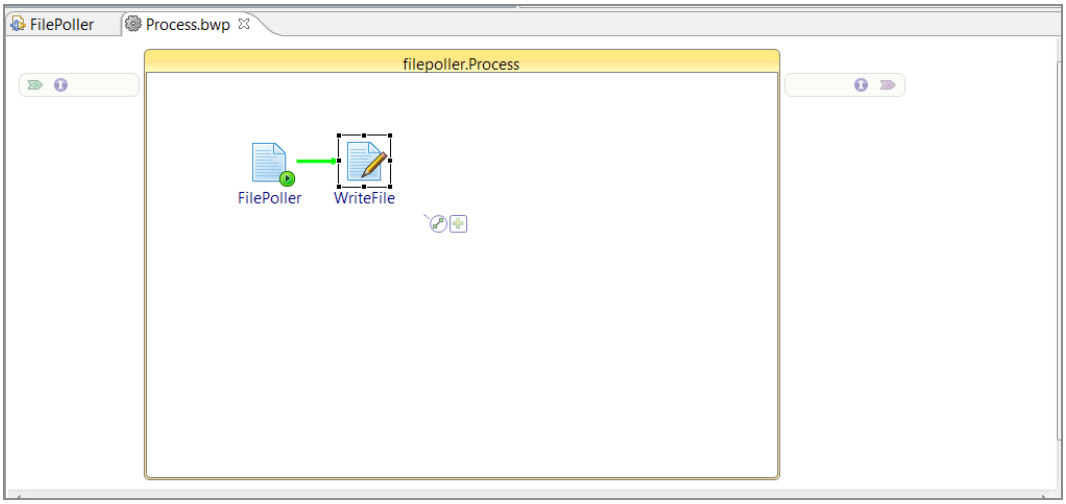
   The **File Poller** activity polls every 5 seconds and transitions the contents of the `fileread.txt` to the **Write File** activity.

4. Open the file `c:\tmp\FilePoller\output.log` and verify the updated information, for example, Hello BusinessWorks! printed in this file.

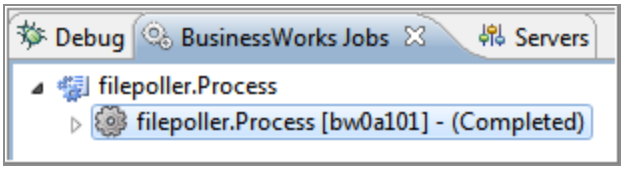
   While keeping the application running, explore adding and deleting the words in the `fileread.txt` file and notice the corresponding changes made to the `output.log`

file.

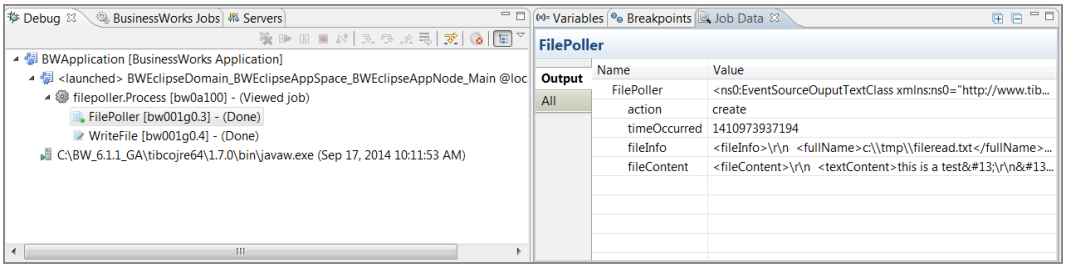
5. The path taken by the engine for running the process is displayed. Transitions turn green to specify that a path was run.




6. Click the **Businessworks Jobs** view on the top left to see the jobs created for the process.




7. Click a particular activity, then click the **Job Data** view on the top right to see the input and output data of the activity.




8. To stop the current job, click the **Stop** button on the **Console view** toolbar.

# 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 *ActiveMatrix BusinessWorks 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.

# Installing PostgreSQL

This topic explains how to install the PostgreSQL database and create the database and tables required for the Bookstore tutorial.

### Procedure

1. Download and install PostgreSQL from http://www.postgresql.org/download/.

   Note the superuser password that you create as part of the installation process.

   > **Note:** If installing on Windows, do not install or run as Administrator.

2. Open a terminal window and navigate to the root folder of the PostgreSQL installation. Open `pg-env.bat` and verify the path settings. Save the file if you have made any changes. Start command line interface and type `pg_env` to load environment variables.

3. Start the server by navigating to the `bin` folder of the install directory and type: `pg_ctl start`.

   Enter the password you created for the superuser.

4. Open another terminal window and navigate to the *BW_HOME*`\samples\binding\rest\BookStore\scripts` folder. Open `readme.txt`. On Unix systems, use the first command in the readme to start the script from the **psql** window. On Windows, copy the second command to start the script from the command line.

5. Navigate to the PostgreSQL `bin` folder and paste the command line into the terminal window. Modify the command as needed. For Windows, use forward slashes in the command.

   Run the command to create the database, the database tables, and to populate the database.

6. Open the PostgreSQL pgAdmin UI utility to see the database and tables.

# Creating a New Process

These steps show how to create a new process.

### Procedure

1. Open TIBCO Business Studio for BusinessWorks.
2. Open the **Design** perspective by clicking 

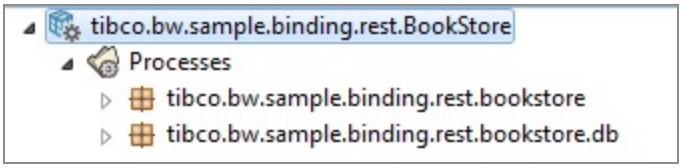
in the upper right.
3. Click the **File Explorer** tab. If the tab is not visible, click **Window > Show View > Other > FileSystem > File Explorer** and click **OK**.
4. Click **File > Switch Workspace** and select or open a clean new workspace.
5. In the samples directory, select **binding > rest > Bookstore** and double-click **tibco.bw.sample.binding.rest.BookStore.zip**.

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

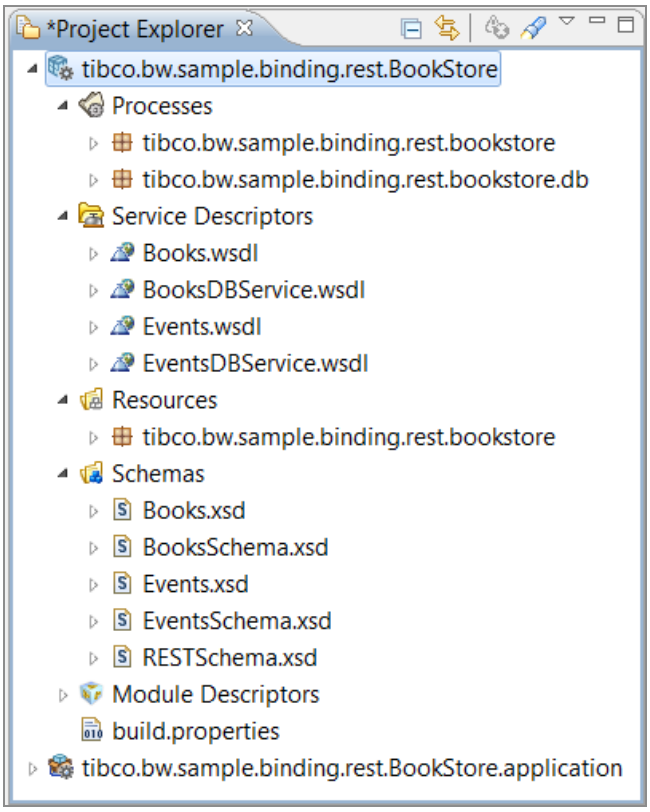
6. In the **Project Explorer**, expand the **tibco.bw.sample.binding.rest.BookStore** project.
7. You can also import the sample using the **File > Import > General > Existing Studio**

**Projects into Workspace > Select Archive File > Browse** option.

8. The project is displayed in the **Project Explorer** panel on the left.



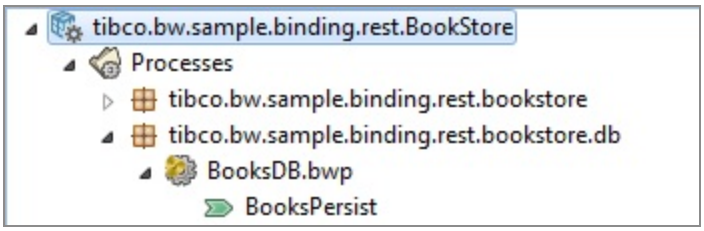

9. Expand the folders in the project to see all the project processes and resources. For more information on folder structure, see the *TIBCO ActiveMatrix BusinessWorks™ Application Development* guide .




10. Expand **Processes** and then expand **tibco.bw.sample.binding.rest.bookstore.db**.

    See **BooksDB.bwp**.

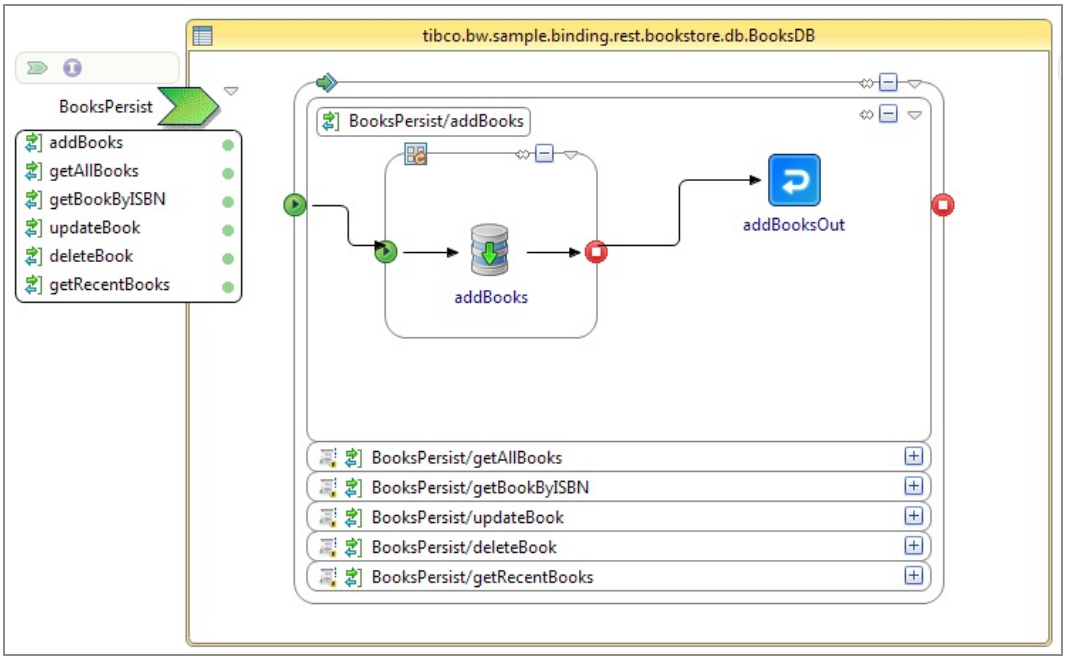
> **Note:** The `.bwp` extension indicates that it is an ActiveMatrix BusinessWorks™ process.




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

    The process comprises:

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




12. Double-click an operation to display the process for example, **BooksPersist > addBooks**.

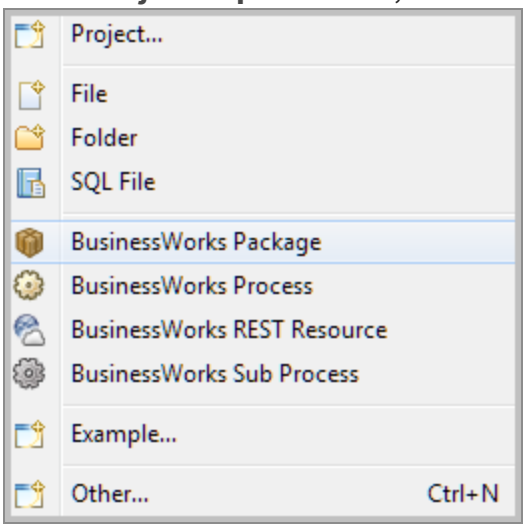
    a. In the addBooks operation, you can see a JDBC activity.

    b. The activity is repeated using a ForEach group.

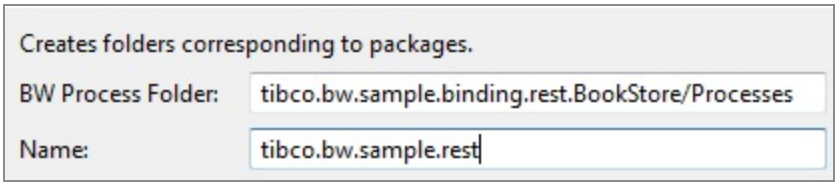
    c. addBooksOut represents the **Reponse** to the web service request.

13. To add a new process package named tibco.bw.sample.rest, right-click on **Processes**

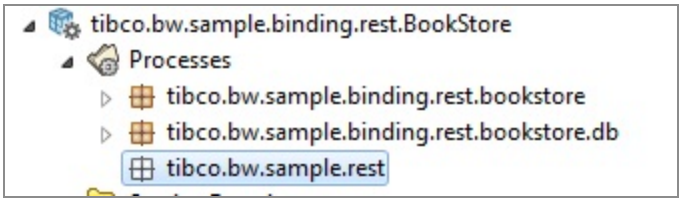
in the **Project Explorer** view, and select **New > BusinessWorks Package**.




14. In the BusinessWorks Package screen, specify tibco.bw.sample.rest in the **Name** field.




15. Click **Finish** and verify that the new package `tibco.bw.sample.rest` has been added in the **Project Explorer** view.




# Building a REST Service

This section details how to build a REST service.

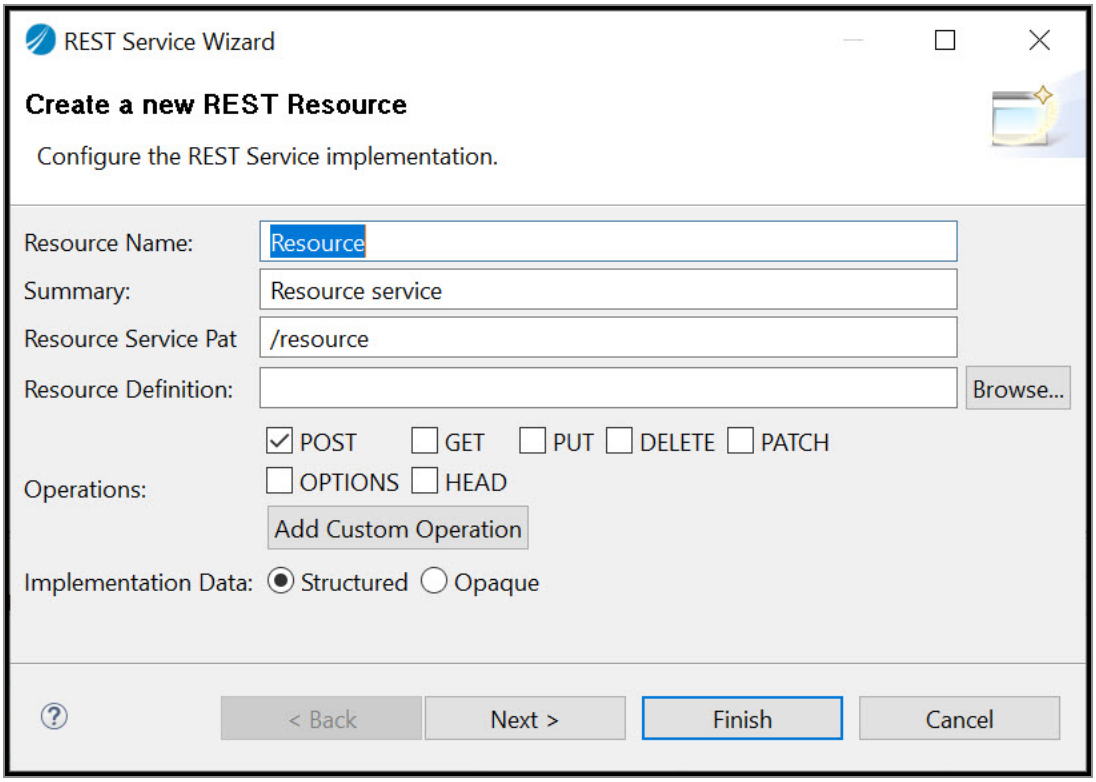
### Before you begin

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

### Procedure

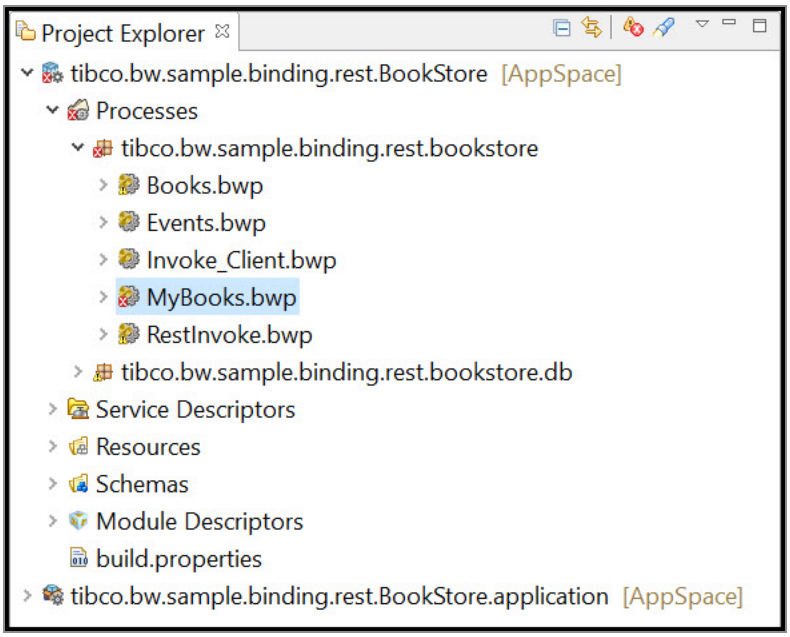
1. To define a REST Resource named MyBooks, select **tibco.bw.sample.binding.rest.BookStore > New > BusinessWorks REST Resource**.

   The REST Service Wizard window is displayed.



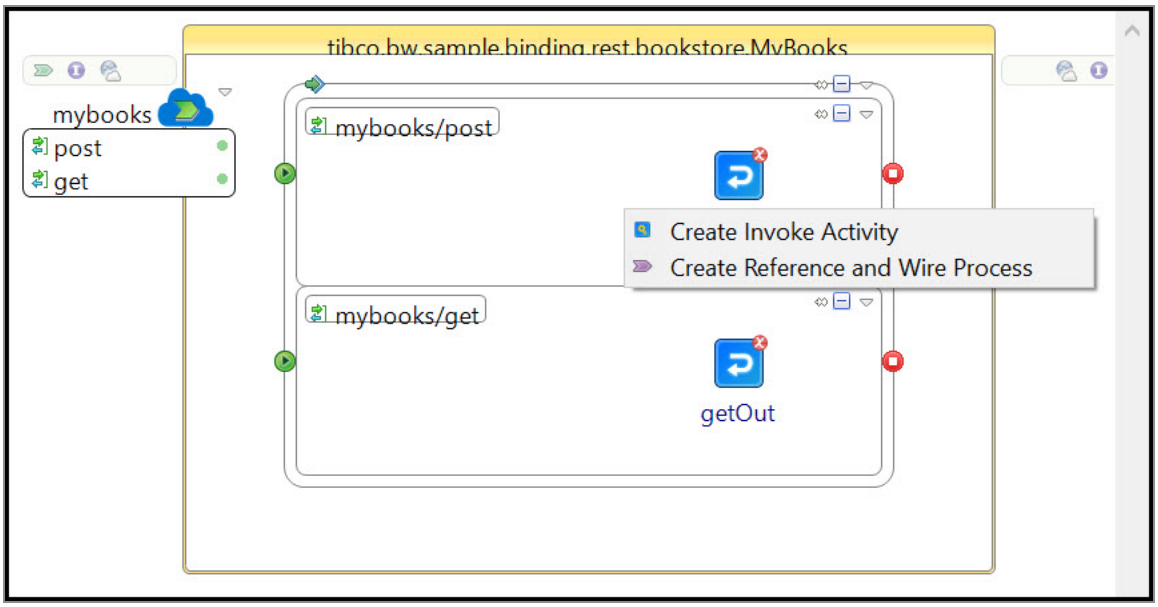

2. Specify the following values in the REST Service Wizard window.
   a. **Resource Name**: MyBooks
   b. **Summary**: Summary about the new REST service. (default)
   c. **Resource Service Path**: Auto-filled
   d. **Resource Definition**: Select **Browse > Schemas > Books.xsd > Books** in the Select 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**.



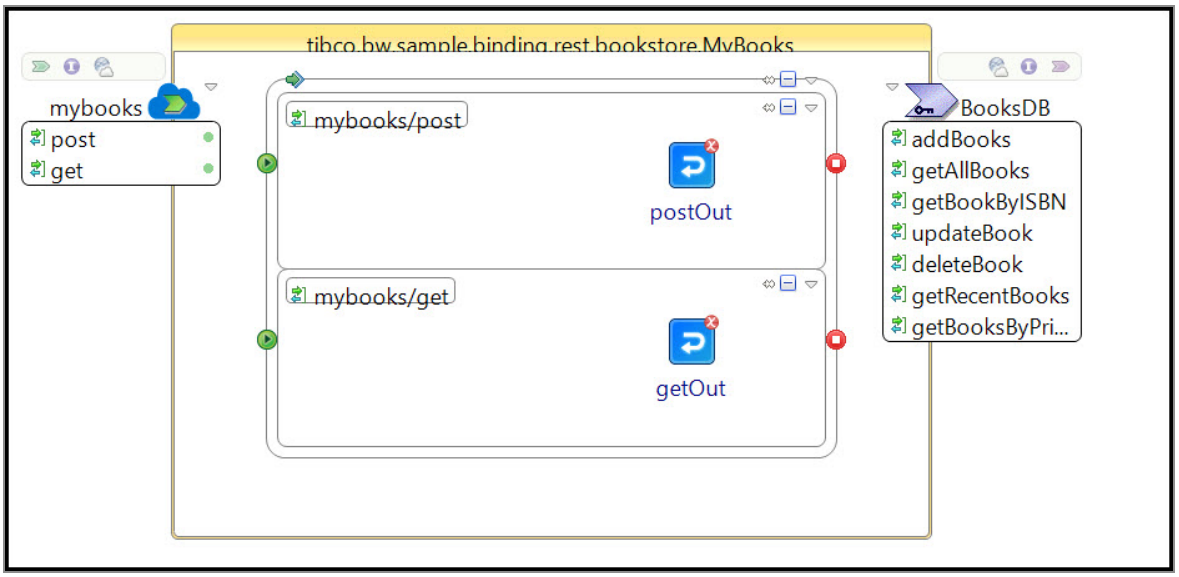

4. Open the **tibco.bw.sample.binding.rest.bookstore.db** package in the **Project Explorer** and select the **BooksDB.bwp** process. Drag it to the **Process Editor** and drop it on the implemented POST operation.

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

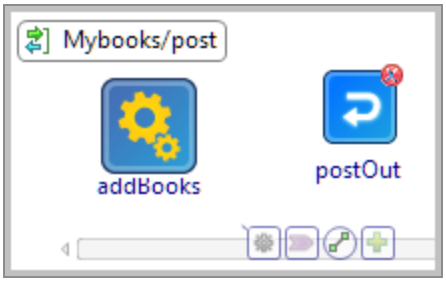



5. Select **Create References and Wire Process**.

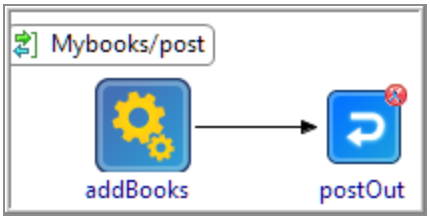
The references are added to the process. The purple chevron indicates the service and its operations that can be referenced by the process.




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**.

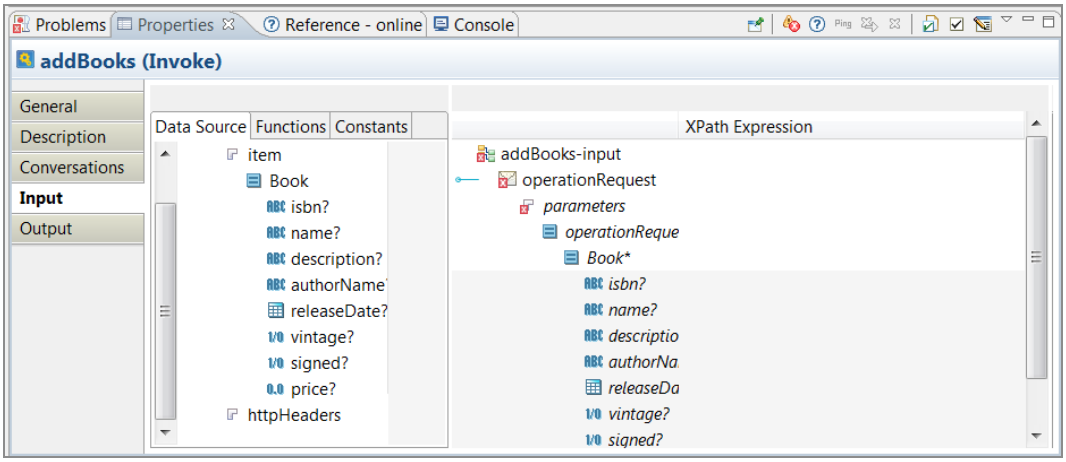



8. Click the **getAllBooks** operation and select, drag, and drop the operation to the left

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

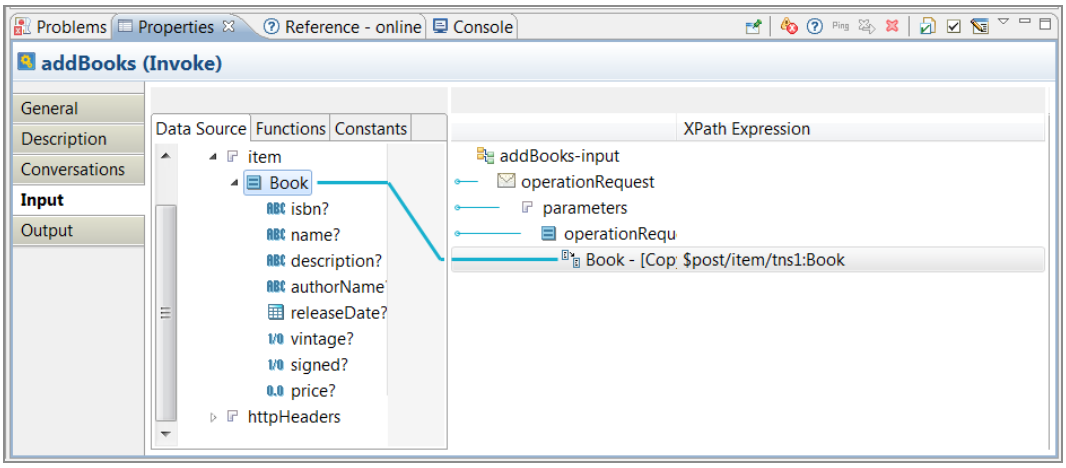
9. Connect **getAllBooks** to **getOut**.


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.

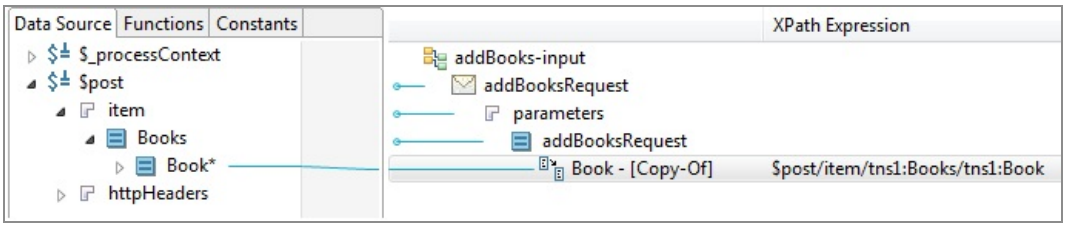



13. Drag the Book element from the left to the Book* element on the right.
14. In the pop-up window, select **Make a Copy of each " Book"** and click **Finish**.

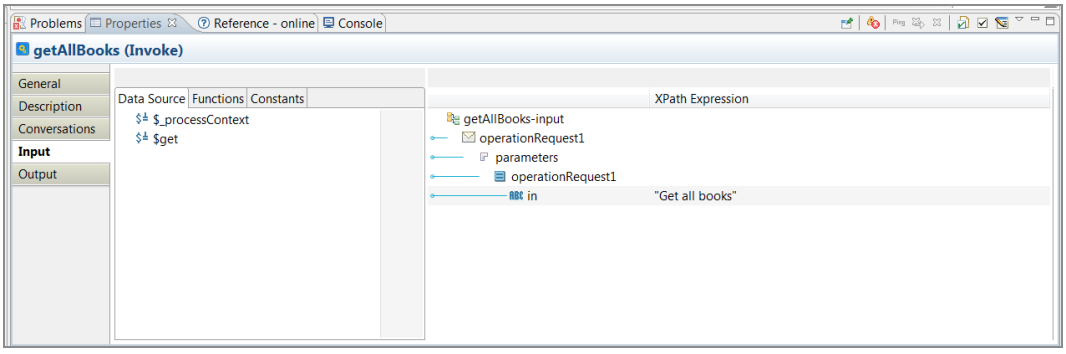
    The **Input** tab looks like this:




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:

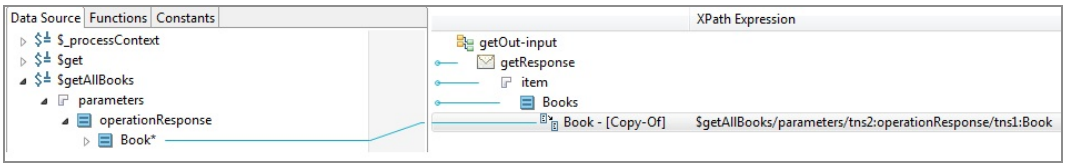



18. Click **getAllBooks** and select **Properties > Input**.
19. In the **XPath Expression** pane, add a dummy value to the input element, such as, "Get All Books". The input must be in quotes.




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

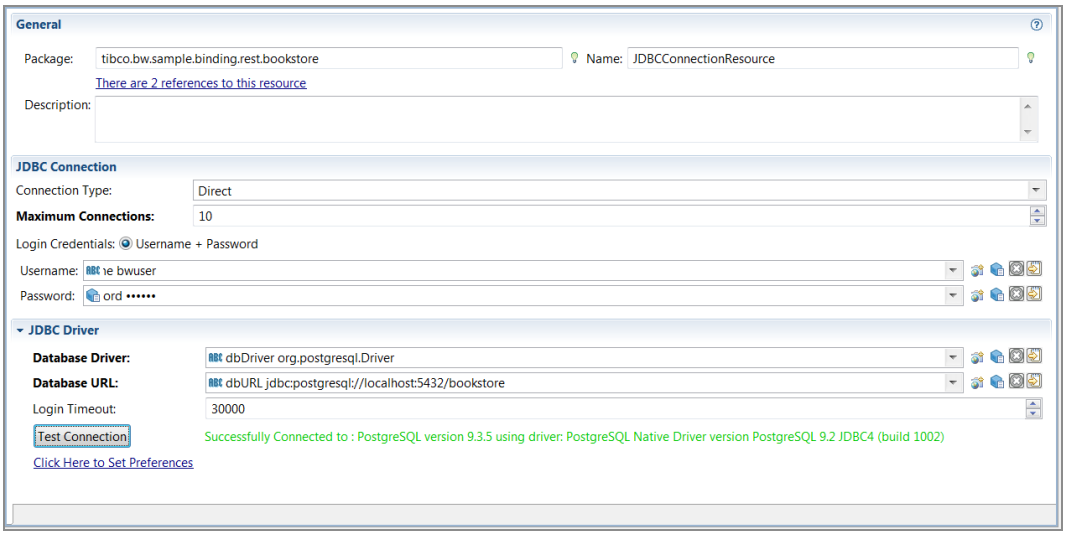
Your project is complete without any errors.

# Testing the REST Service

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.bw.sample.binding.rest.BookStore.application** process and expand the **Package Unit > Properties** folder.
2. Provide valid values for the application properties including a valid username, password, and database URL to connect to your PostgreSQL database if different from the default setting.
3. Verify your JDBC connection.
    a. Expand the **Resources** folder in the Project Explorer for the **tibco.bw.sample.binding.rest.BookStore** process.
    b. Double-click **JDBCConnectionResource.jdbsResource**.
    c. In the **JDBC Driver** section of the window, click **Test Connection** to verify the connection. If you change the JDBC driver folder from the default, click **Click Here to Set Preferences** and set the JDBC driver folder to the folder where you downloaded PostgreSQL JDBC Driver.

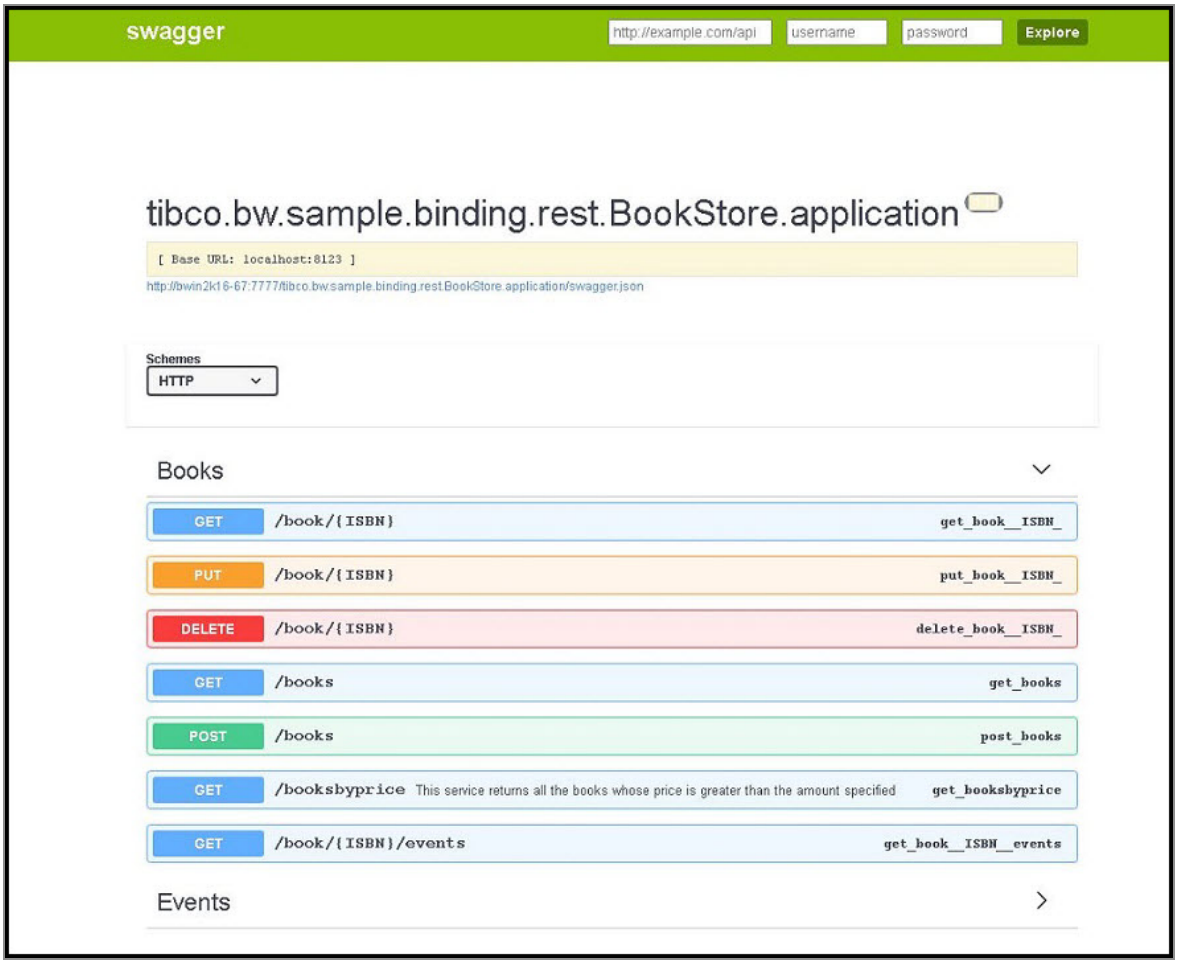



4. Click **File > Save**.
5. In the **Project Explorer**, expand the **Processes** directory if it is not expanded and double-click **MyBooks.bwp**.
6. Click **Run > Debug Configurations**.
7. In the left-hand tree of the **Debug Configuration** wizard, expand **BusinessWorks Application** and select **BWApplication**.
8. Click the **Applications** tab and then click the **Deselect All** button if you have multiple applications. Select the checkbox next to **tibco.bw.sample.binding.rest.BookStore.application**.
9. Click **Debug**. This runs the sample in **Debug** mode.

   The **Console view** is opened and shows engine messages similar to: Started BW Application [tibco.bw.sample.binding.rest.BookStore.application:1.0].
10. In the **Console view**, press Enter to display the prompt: `<>@BWEclipseAppNode>`

    Enter the OSGi command `lrestdoc`. This lists the Swagger UI URL as the discovery URL: [Application Name]: tibco.bw.sample.binding.rest.BookStore.application [Discovery Url]: http://localhost:7777/tibco.bw.sample.binding.rest.BookStore.application
11. Open the Google Chrome browser.
12. Open http://localhost:7777/tibco.bw.sample.binding.rest.BookStore.application
13. Click **Books** or **Events** to see the operations. Click **MyBooks** to see the REST service

operations you just added. For more information, see the section Testing the POST and GET Operations.




14. 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
    }
```

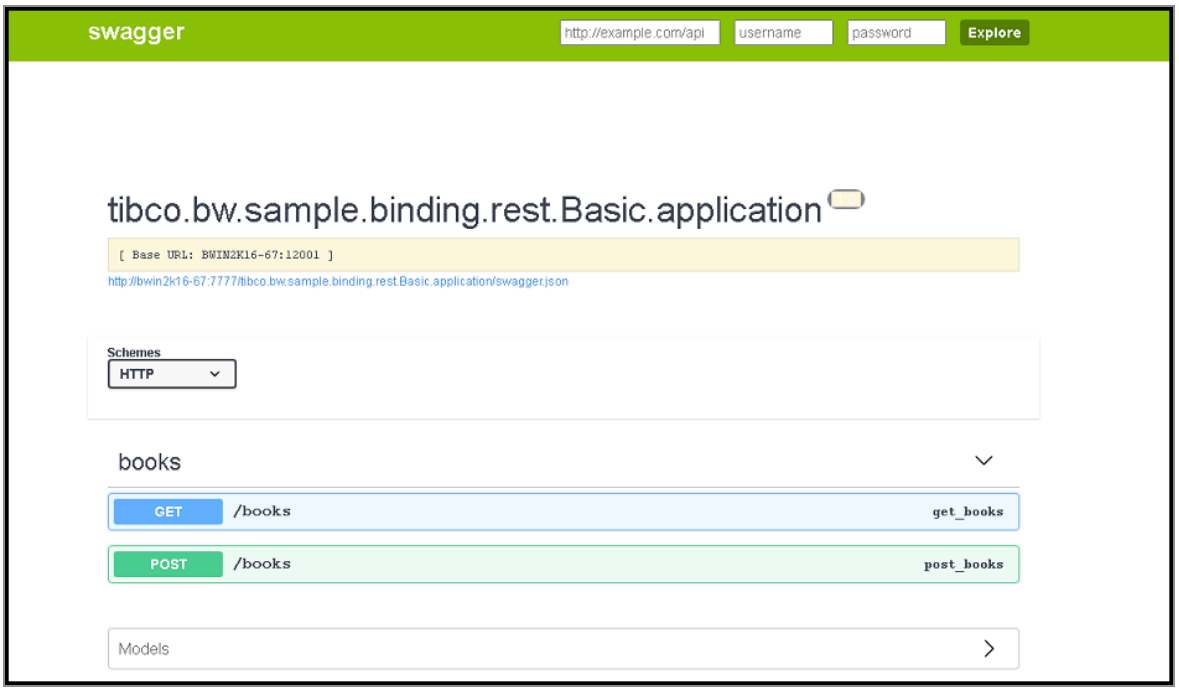

# Testing the POST and GET Operations

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

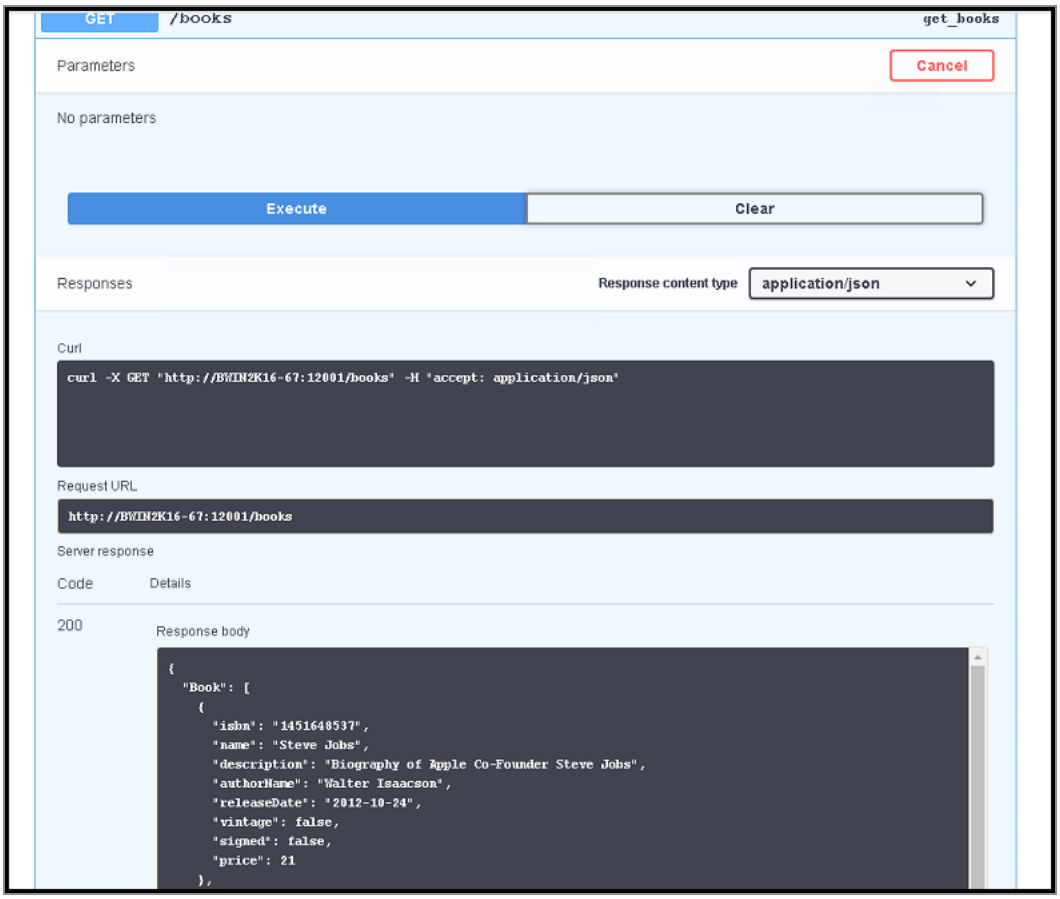
1. Click **books**.

   It expands and displays the POST and GET operations.




2. Click the **POST** icon to display its details.
3. Click the **Try it out!** button.
4. Provide values to the Books parameter and then click **Execute**. You can use the JSON payload in the `<BW_HOME>\samples\binding\rest\BookStore\samplejson` folder.
5. Now click **GET** to display its details.
6. Click **Try it out!**.
7. Click **Execute**.

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



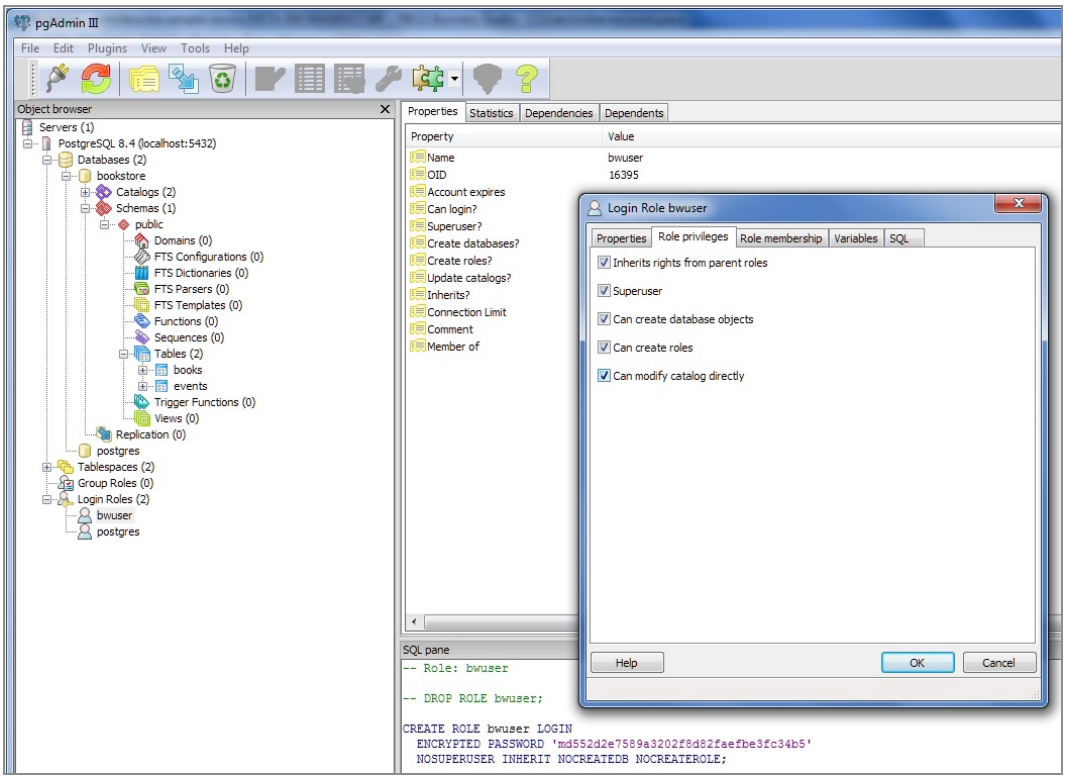

8. After you have finished, go back to TIBCO Business Studio™ for BusinessWorks™ and click in the **Console view** to stop the process.

# Troubleshooting

Your 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 |
|---|---|
| Unable to insert rows into the database using the `dbsetup.sql` script in the scripts folder. | Use the sample JSON payload from the sample json folder to post the data. |

| 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. Use the `lrestdoc` command in the **Console view** to get the Swagger UI URL. |
| Problem markers are visible in the project. | Clean the project by invoking **Project > Clean** or by switching to a clean new workspace. |
| Getting the File was not found exception. | Ensure that the `books.json` and `book_put.json` files are present at the location described in the Input_File and Input_File_1 module properties. |
| The PostgreSQL server does not start. | Make sure you are not running as Administrator. |
| The database and database tables are not created. | Open the `readme.txt` file for the sample, located in the *`BW_HOME`*`\samples\binding\rest\BookStore\scripts` folder. Run the `dbsetup.sql` script from a command line, not the **psql** window. |
| Getting an unregistered user error message while running the process. | Select all the checkboxes in the **Role Privileges** tab in the pgAdmin UI and run the process again. See the image below. |


pgAdmin III
File Edit Plugins View Tools Help
Object browser
Servers (1)
PostgreSQL 8.4 (localhost:5432)
Databases (2)
bookstore
Catalogs (2)
Schemas (1)
public
Domains (0)
FTS Configurations (0)
FTS Dictionaries (0)
FTS Parsers (0)
FTS Templates (0)
Functions (0)
Sequences (0)
Tables (2)
books
events
Trigger Functions (0)
Views (0)
Replication (0)
postgres
Tablespaces (2)
Group Roles (0)
Login Roles (2)
bwuser
postgres
Properties Statistics Dependencies Dependents
Property Value
Name bwuser
OID 16395
Account expires
Can login?
Superuser?
Create databases?
Create roles?
Update catalogs?
Inherits?
Connection Limit
Comment
Member of
Login Role bwuser
Properties Role privileges Role membership Variables SQL
Inherits rights from parent roles
Superuser
Can create database objects
Can create roles
Can modify catalog directly
Help OK Cancel
SQL pane
-- Role: bwuser
-- DROP ROLE bwuser;
CREATE ROLE bwuser LOGIN
ENCRYPTED PASSWORD 'md552d2e7589a3202f8d82faefbe3fc34b5'
NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE;

# 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 invoke must be accessible from the reference process at the time of its invocation.

## 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 tibco.bw.sample.binding.rest in the **Project Name** text box. Do not change the remaining default settings.
5. Click **Finish**. This step creates a new application module with an empty process.

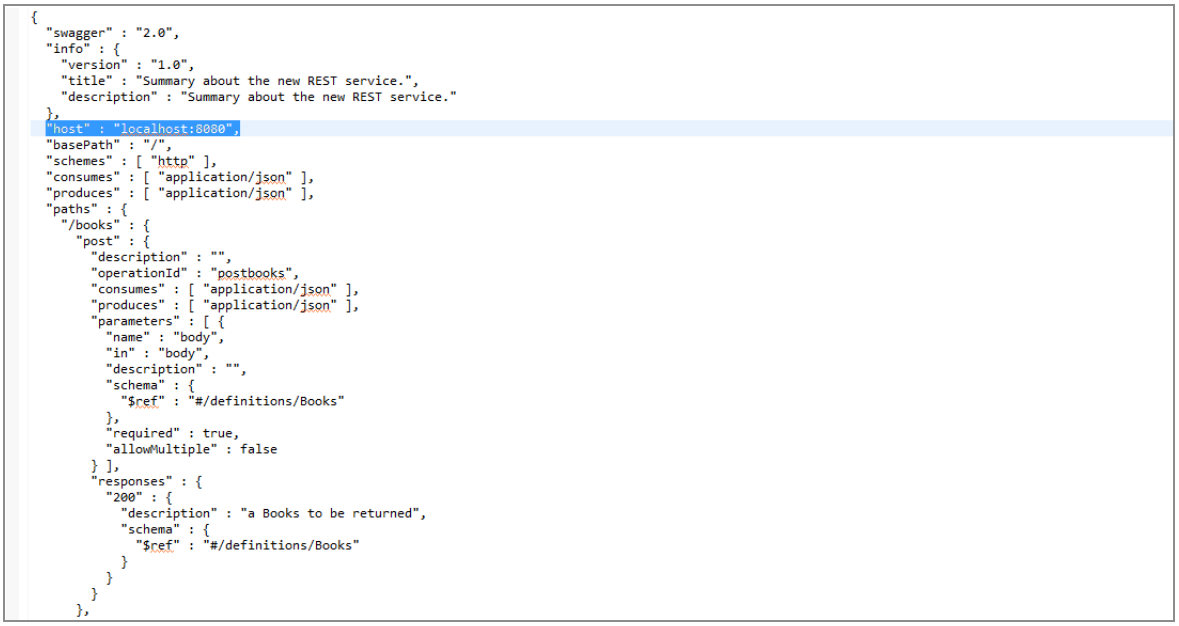
## Importing the JSON File into your Project

1. In the **Project Explorer**, expand tibco.bw.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

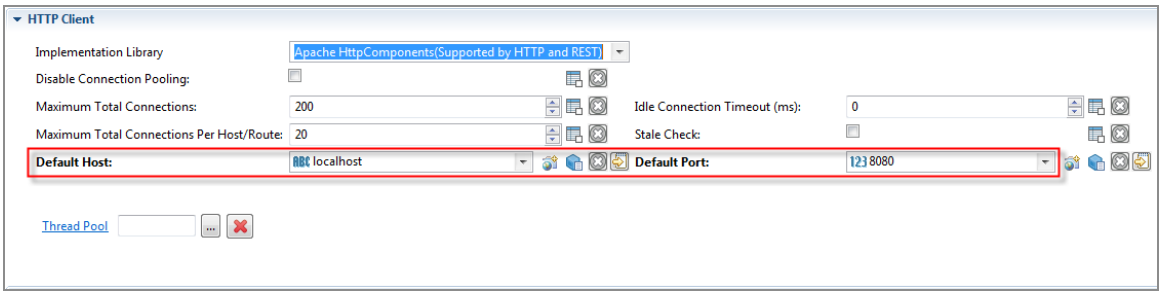
1. In the **Project Explorer**, completely expand the **tibco.bw.sample.binding.rest** folder under **Service Descriptors**.
2. Select the **/books** under **Books.json** and drag and drop it to the right side of the process in the Process Editor. The references are added to the process. The purple chevron 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.
10. Select the checkbox under **Fix** and click **OK**.
11. 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.
12. Expand **$get** in the **Data Source** tab completely.
13. In the XPath Expression pane, expand the **post-input** completely.
14. Drag and drop **Book*** from the **Data Source** tab to the **Book*** under post-input in the **XPath Expression** pane.
15. In the Drop dialog, select **Make a copy of each "book"** radio button and click **Finish**.
16. Click **Show Check and Repair** icon in the icon bar on the upper right corner of the Properties view.

17. Select the checkbox under **Fix** and click **OK**.
18. Click **Show Check and Repair** icon again. Select the checkbox under **Fix** and click **OK**.
19. In the **Project Explorer**, select `Books.json` under **Service Descriptors** of **tibco.bw.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"
            }
          }
        }
      },
```

20. Expand the Resources folder under the **tibco.bw.sample.binding.rest.basic** application module.
21. Double-click **HttpClientResource.httpClientResource**.
22. In the **HTTP Client** section, change the Default Host and Default Port to the values in the `Books.json` file and select the **Default Confidentiality** checkbox.

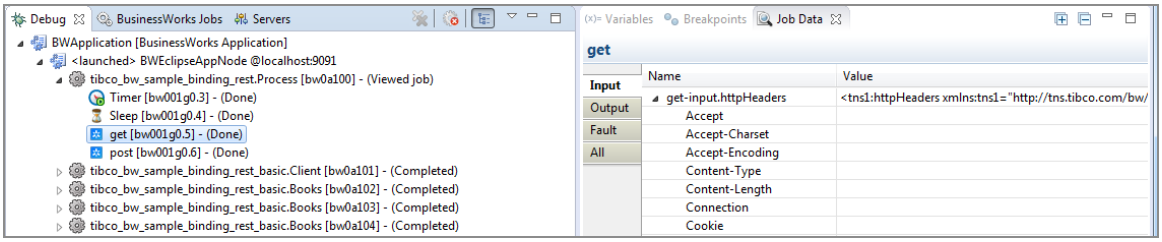


23. Click **File > Save All**.

## 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, then click **Deselect All** if you have multiple applications.
4. Select the checkboxes next to **tibco.bw.sample.binding.rest.basic.application** and **tibco.bw.sample.binding.restapp**.
5. Click **Debug**. This runs the sample in debug mode. The Console view is opened and shows engine messages similar to: Started BW Application [ tibco.bw.sample.binding.rest.Basic.application:1.0]
6. In the **Debug** view, expand **BWApplication [BusinessWorks Application] > <launched> BWEclipseAppNode > tibco.bw.sample.binding.rest.Process** and select **get**.
7. In the **JobData** view, you can see the job data of the **get** activity.



# Administration Tutorial

The administration framework supports application deployment either through the BWAdmin command line utility or the Admin UI. The scripts provided can be used to set up runtime entities that are useful for testing purposes. This tutorial walks you through running scripts and navigating runtime entities in the Admin UI.

The administration framework contains:

- The Admin UI hosted on TIBCO® Enterprise Administrator.
- A powerful back-end BWAgent designed to scale across large numbers of actual or virtual machines to manage large-scale deployment.
- A simple, flexible, and easy-to-use BWAdmin command line utility.

This section shows how to:

- Create runtime entities (Domains, AppSpaces, and AppNodes) and upload and deploy archive files using scripts.
- Start and stop applications using the Admin UI.
- Navigate runtime entities using the Admin UI.

For more information on runtime entities, see Administration Concepts in the*TIBCO ActiveMatrix BusinessWorks™ Concepts* guide.

For more information on the commands used in this sample, see Getting Started in the *TIBCO ActiveMatrix BusinessWorks™ Administration* guide.

# Running Admin Sample Scripts

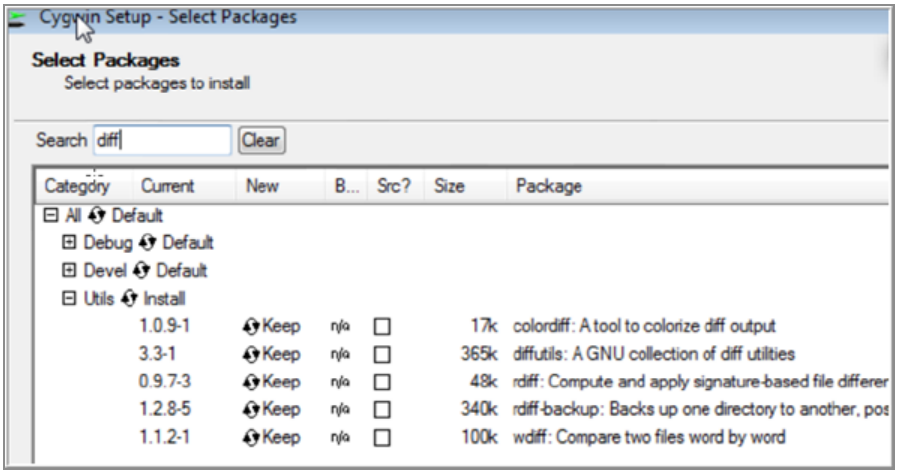
This tutorial walks you through running the Bookstore and Acme administration scripts.

### Before you begin

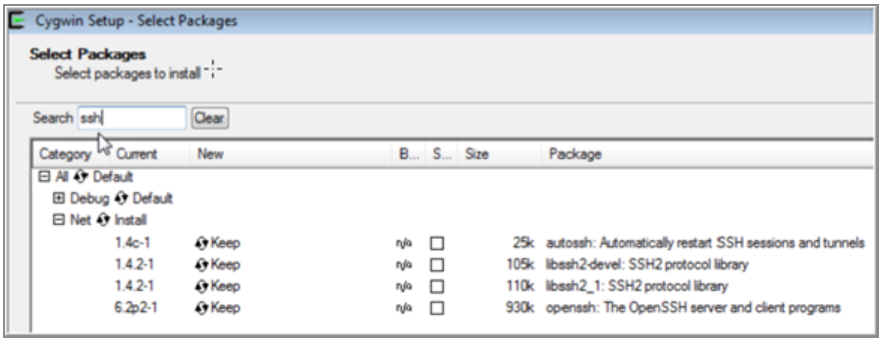
If you want to run scripts, the installation directory cannot contain one or more spaces. For example, on Windows, TIBCO ActiveMatrix BusinessWorks™ should not be installed in the `Program Files` folder.

1. Optional. Install TIBCO Enterprise Message Service™ 8.x or higher if you want to run the `Acme.com` applications.
2. Optional. Install PostgreSQL 9.3.x if you want to run the Bookstore sample.
    - For more information, see Installing PostgreSQL.
3. On Windows, you must install the latest Cygwin 64-bit version from http://www.cygwin.com/. Scripts must be run with Cygwin. Install the **Utils** and **Net** package that contains the `diff.exe` and `ssh.exe` executables.
    - Select the **Utils** and **Net** packages and change the option from Default to Install. Refer to the following images that show the **Utils** and **Net** packages.

    Utils Package

    Net Package

## Procedure

1. Install ActiveMatrix BusinessWorks™. For example,

a. On Unix, install ActiveMatrix BusinessWorks into `/opt/tibco/bw-6.x` as *TIBCO_HOME*.

b. On Windows, install ActiveMatrix BusinessWorks into `/opt/tibco/bw-6.x` as *TIBCO_HOME*.

From here onwards, the following instructions use the directory path mentioned above to refer to certain installed files. Replace this path with the install directory of your choice.

> **Important:** On Windows OS, avoid installing ActiveMatrix BusinessWorks in a directory with space. The product binary supports it, however, the scripts in `${BW_HOME}/scripts/admin` and `${BW_HOME}/samples/core/admin` are bash scripts and work best with directory paths without space.

2. Install TIBCO Enterprise Administrator 2.0 or compatible version, in the same *TIBCO_HOME*. When the TIBCO Enterprise Administrator installer prompts for *JAVA_HOME* path, point to `/opt/tibco/bw-6.x/tibcojre64/1.7.0` on UNIX. On Windows, point to `c:/tibco/bw-6.x/tibcojre64/1.7.0`.

   You can install TIBCO Enterprise Administrator in a separate *TIBCO_HOME*. If you choose to do that, refer to Step 4.

3. Install TIBCO Enterprise Message Service 8.1, or compatible version, in the same *TIBCO_HOME*. You can install the TIBCO Enterprise Message Service in a separate *TIBCO_HOME*. If you have installed the TIBCO Enterprise Message Service in a separate *TIBCO_HOME*, refer to Step 4.

4. On UNIX: If you have installed TIBCO Enterprise Administrator and/or TIBCO Enterprise Message Service in a separate <TIBCO_HOME>, then ActiveMatrix BusinessWorks, open `/opt/tibco/bw-6.x/bw/6.x/scripts/bashrc.sh` in a text editor and adjust *TEA_HOME* and *EMS_HOME* to point to where you have installed them.

5. This step is applicable only to Windows OS. From a proper text editor (Do not use Notepad.exe), open and edit `c:/tibco/bw-6.x/bw/6.x/scripts/bashrc.sh`. Search and replace all occurrences of `C:/` (or whatever letter drive that you have installed ActiveMatrix BusinessWorks) with `/cygdrive/c/` (or `/cygdrive/<drive-letter-where-you-installed-bw6>`).

   Adjust *TEA_HOME* and *EMS_HOME* to point to the location, where you have installed them.

6. Source the `bashrc.sh` script from your `~/.bashrc` or `~/.profile` files.
   - **On UNIX**: Locate your `~/.bashrc` or `~/.profile` file and add the following line at the end:

     ```
     source /opt/tibco/bw-6.x/bw/6.x/scripts/bashrc.sh
     ```

   - **On Windows**: Edit the `.bashrc` file located at *`CYGWIN_HOME`*`/home/<`*`UserName`*`>` to add the following line at the end:

     ```
     source c:/tibco/bw-6.x/bw/6.x/scripts/bashrc.sh
     ```

7. To verify that the environment variables are configured correctly, open a new command-line window, navigate to `<`*`TIBCO_HOME`*`>/bw/6.x/scripts/admin/`, and run `./bw6env.sh`. This script prints the required and optional environment variable configurations on the screen.

   **Note:** The script returns the environment variable configurations only if the environment variables are set correctly.

   The following is a sample output on UNIX:

   ```
   Admin@WINAA-2:admin admin$ ./bw6env.sh
   BW 6 Environment Configurations

   Required Environment Variables:
     TIBCO_HOME = /opt/tibco/bw-6.x
     BW_HOME    = /opt/tibco/bw-6.x/bw/6.x
     JAVA_HOME  = /opt/tibco/bw-6.x/tibcojre64/1.7.0

   Optional Environment Variables:
     TEA_HOME   = /opt/tibco/bw-6.x/tea/2.1
     EMS_HOME   = /opt/tibco/bw-6.x/ems/8.1

   Required Binaries on $PATH:

   Optional Binaries on $PATH:
   ```

8. Change to the `admin` folder by typing: admin
9. Issue the following command to create the BookStore sample, create the domains for

the Acme sample, create the Samples-Domain, and deploy all the sample archive files:

`runAll.sh`

**Note:**

- Running the `runAll.sh` script automatically updates the Admin "local" mode to "enterprise" mode.
- You can use `-clean` option that cleans TIBCO Enterprise Administrator Server Data Store and ActiveMatrix BusinessWorks™ Domain Data Store.
- This script may take up to 10 or 15 minutes to complete. To see how long it takes to run the sample, run the `time runAll.sh` command to measure the time the script takes to complete.
- If you do not want to run all the samples at the same time, then run the following commands:
  a. `bootstrap.sh -clean`

     **Caution:** Running the `bootstrap.sh -clean` with the `-clean` option cleans the TIBCO Enterprise Administrator server and the ActiveMatrix BusinessWorks™ domain datastore. Use the `-clean` option only if you want to clear the TIBCO Enterprise Administrator server and ActiveMatrix BusinessWorks™ domain datastore.

  b. `runBookStore.sh` - to run the REST BookStore sample, or
  c. `runSamples.sh` - to run other samples

10. Open a web browser and access the Admin UI at `http://localhost:8777/tea`
11. Log in using:

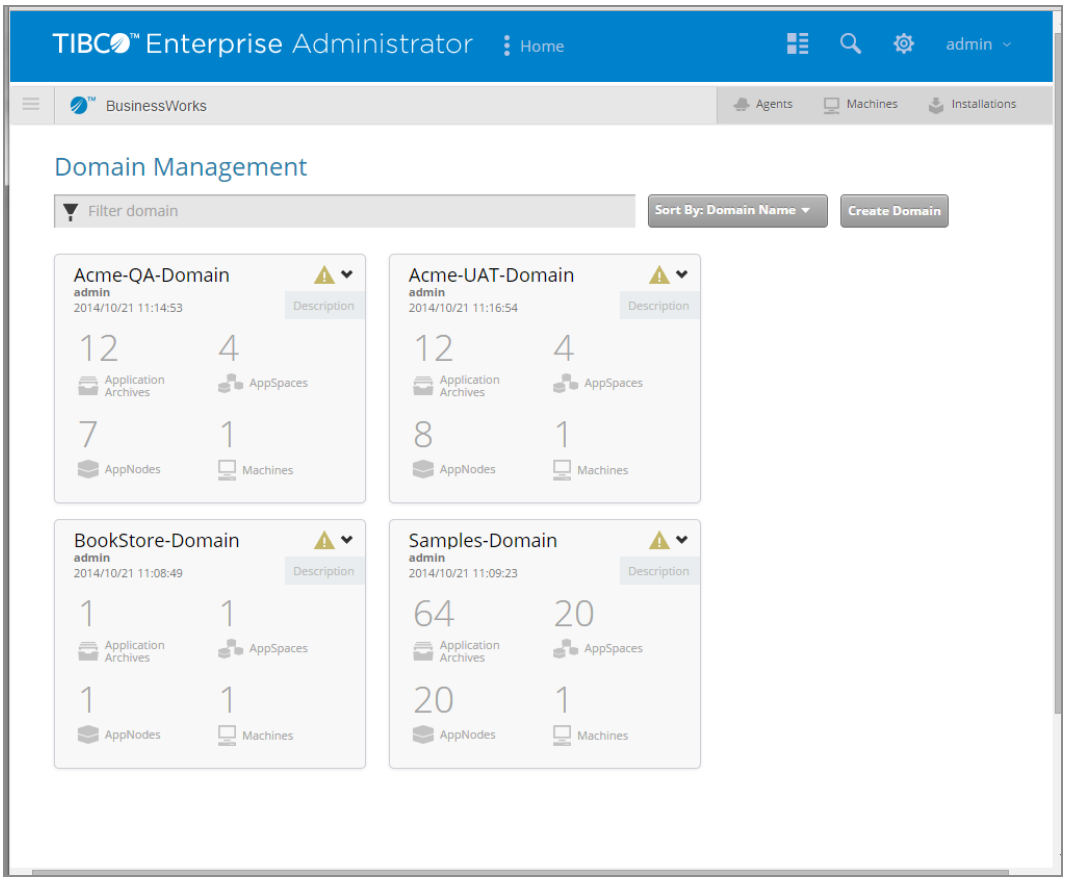
    **Username:**admin

    **Password:** admin

12. Click **BusinessWorks** in the **Products** list to see the following domains in the **Domain Management** screen.

- Acme-QA-Domain
- Acme-UAT-Domain
- BookStore-Domain
- Samples-Domain

Drill down into any of the domains to see the created runtime entities, such as AppSpaces, AppNodes, application archives, and applications.




# Testing the Deployed RESTful BookStore Application from Admin UI

The runAll script creates four domains, with multiple AppSpaces and AppNodes. The instructions in this topic show how to navigate the Admin UI to view the runtime entities.

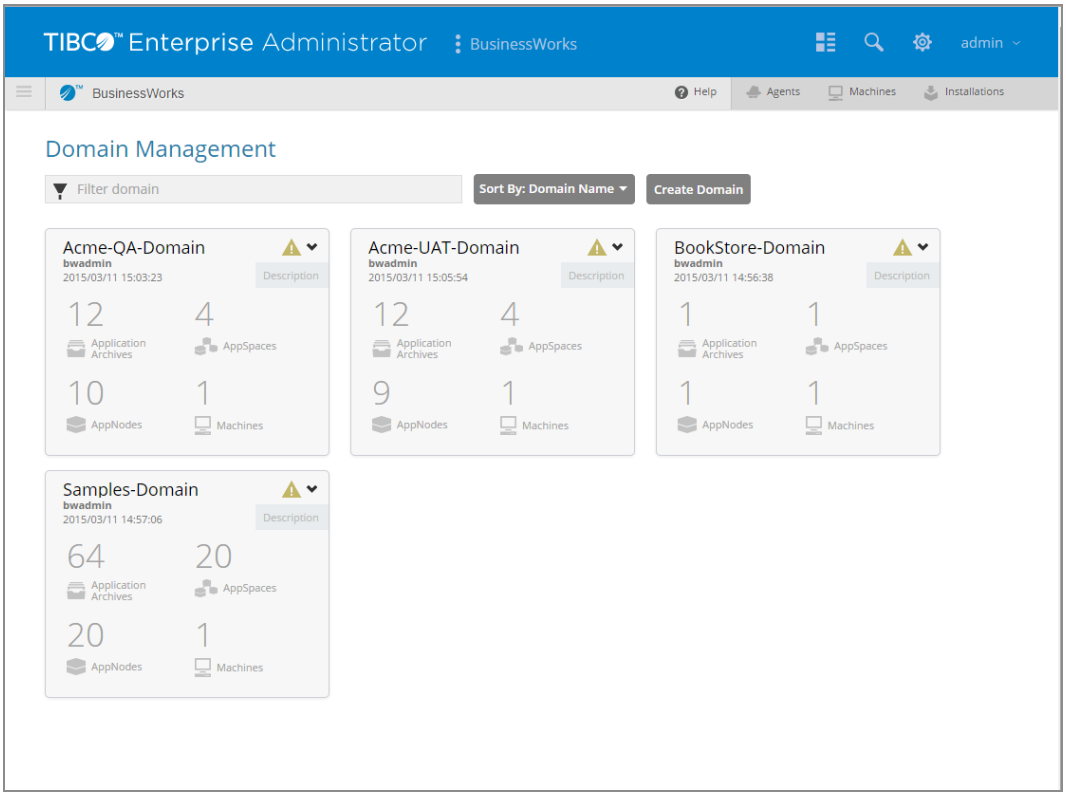
### Procedure

1. In the TIBCO® Enterprise Administrator browser window, click **TIBCO Enterprise**

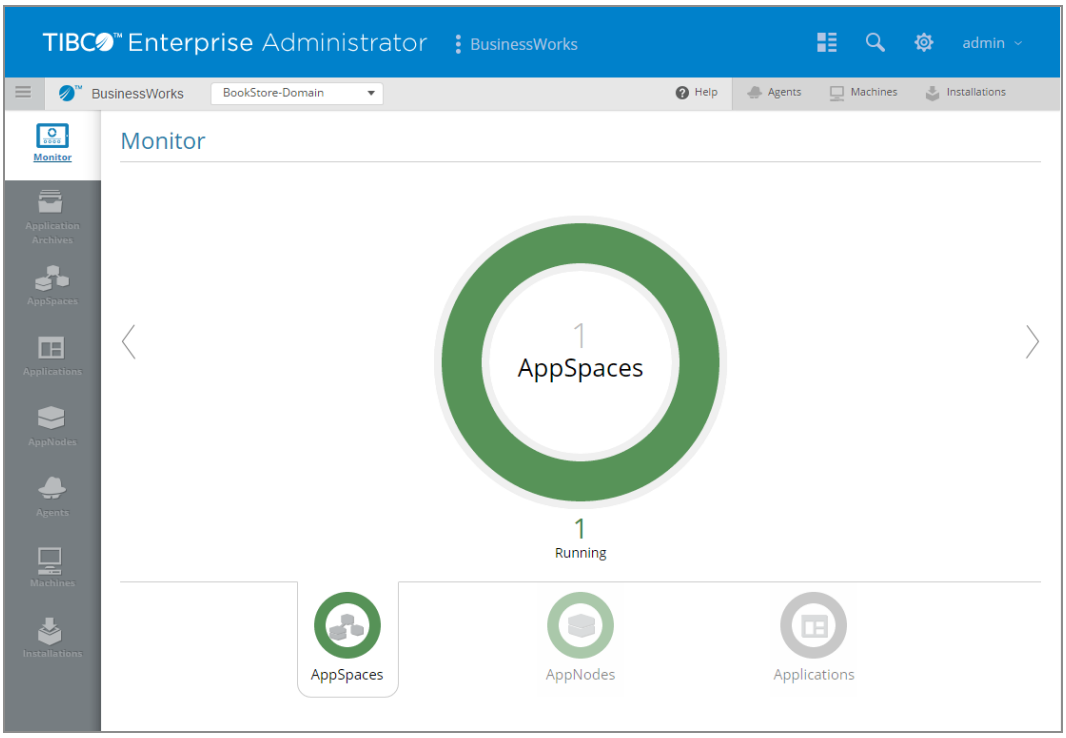
**Administrator** at the top of the page to open the home page.

2. Click the **BusinessWorks** icon in the **Products** list to display the **Domains** page.

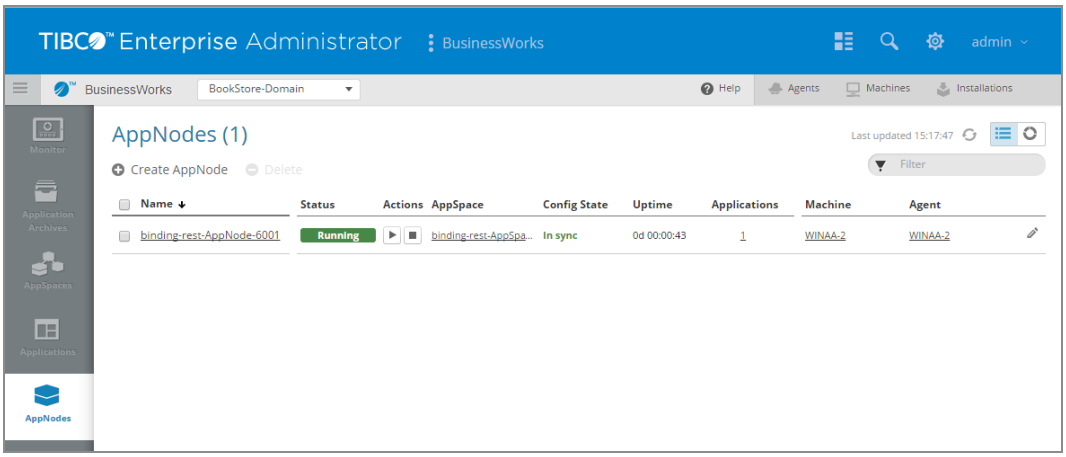
   The **Domains Management** page displays.




3. Click **BookStore-Domain** to drill down into domain details, then click **Monitor** to see the dashboard.



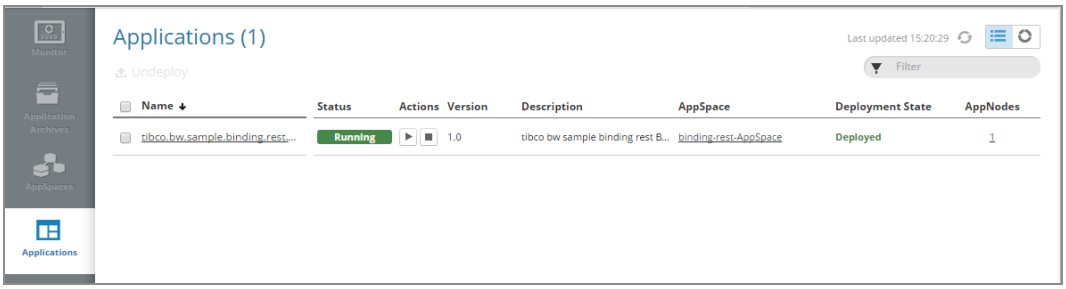

4. Click an entity in the side bar, such as **AppNodes**, to pivot views.



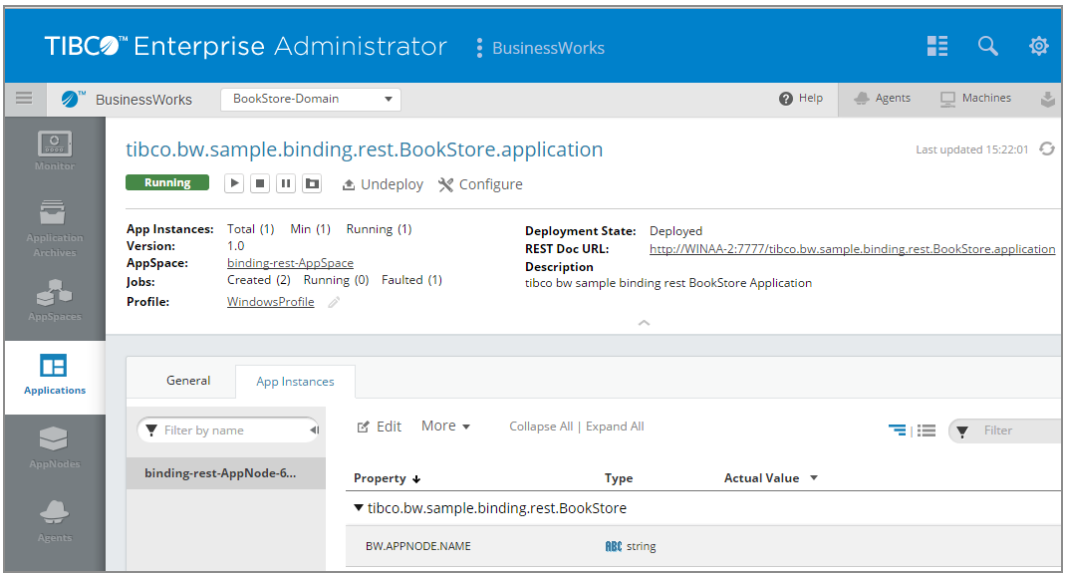

5. To view the BookStore application deployed earlier, select **Applications** on the left.

   A single application archive is displayed.

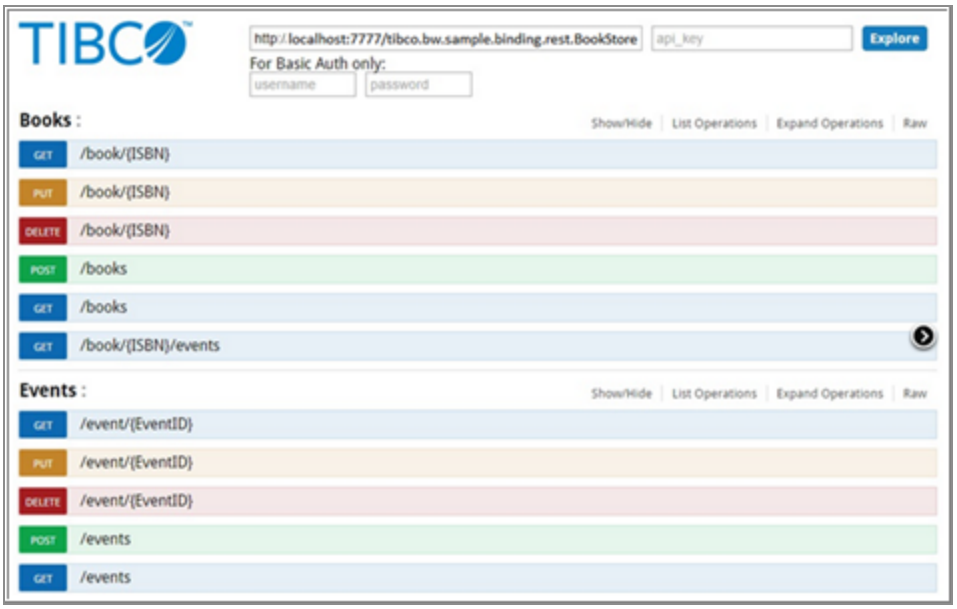

6. Go to **Applications** to view the deployed applications.




7. In the Applications view, click the link for **REST Doc URL**.




8. Test the deployed application in the Google Chrome browser using Swagger UI. Open **localhost:7777/tibco.bw.sample.binding.rest.BookStore.application**.




9. To run the REST operations exposed by the BookStore application, click the **GET/books** icon and then click **Try it Out!**.

   This displays a list of all books. Locate and copy an ISBN.

10. Click the **GET/books{ISBN}** icon to get a book by its ISBN.
11. Enter the ISBN and then click **Try it Out!**.

    The book details can be seen in the Response Body.

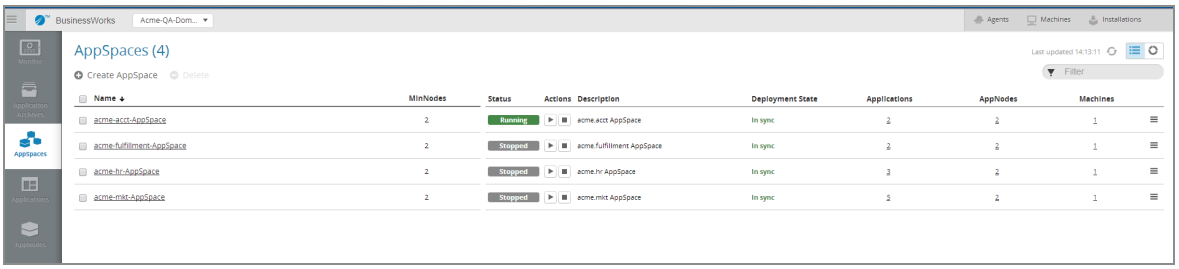
# Defining and Deploying Multiple AppSpaces and AppNodes

The `runAll` script defines multiple AppSpaces and AppNodes and deploys multiple applications to these AppNodes. This topic shows how to navigate the Admin UI to locate these runtime entities.

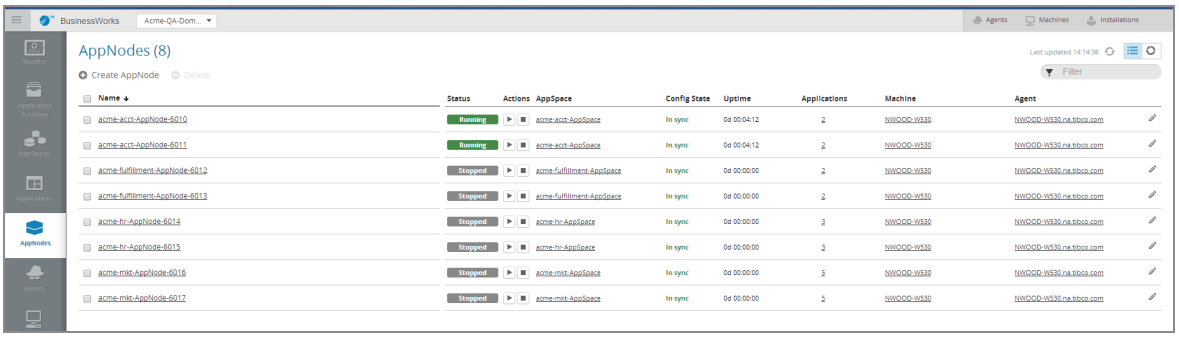
### Procedure

1. Navigate to the **Domain Management** page by clicking the **TIBCO Enterprise Administrator** icon and choosing **BusinessWorks** in the **Products** area.
2. Choose the **Acme-QA-Domain** to see the **Monitor** page that shows the status of AppSpaces, AppNodes, and applications.

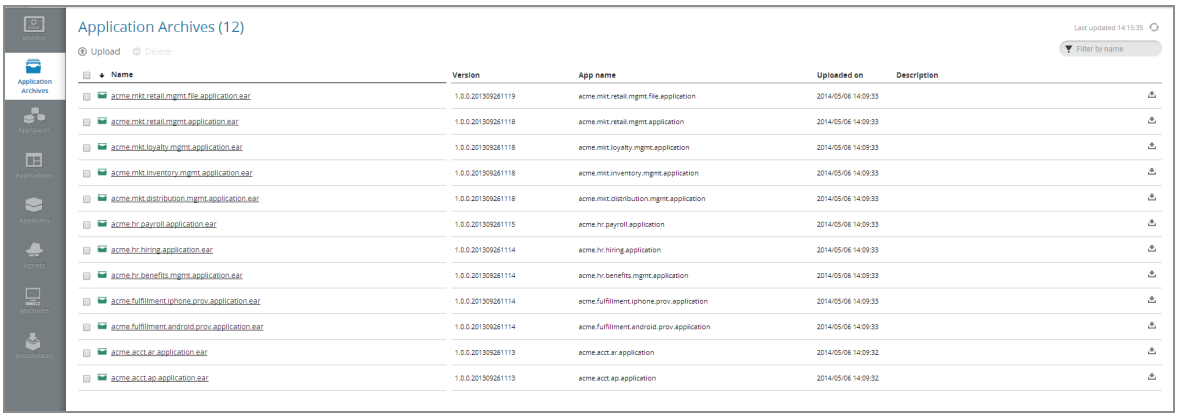
3. Click **AppSpaces** in the side bar to display the **AppSpaces** page. (You can also click the **AppSpace** icon on the **Monitor** page.) Change the view of the page by clicking the icons in the upper right of the screen.




4. Now, view the AppNodes. Click **AppNodes** in the side bar.




5. View the application archives by clicking **Application Archives** in the side bar.




6. Run the `killall.sh` command to shut down all running components.
7. Run the `bwclean.sh` command to clean up.

# Core Admin Sample Scripts

The sample scripts provide a simple and fast way to run the core Admin samples.

These are bash scripts. On Windows, install Cygwin64.

Admin scripts are available in the following folders: *$BW_HOME*/samples/core/admin and *$BW_HOME*/scripts/admin

For information about running the sample scripts, see the "Running Admin Sample Scripts" section in the *TIBCO ActiveMatrix BusinessWorks™ Getting Started* guide.

This sets the *TIBCO_HOME*, *BW_HOME*, *TEA_HOME*, *EMS_HOME*, and *JAVA_HOME* environment variables necessary to run the admin scripts.

All scripts support the `-h` and command-line argument with full documentation of what each script does.

## Location of the Admin Scripts

The admin scripts are available in the following folders:

- The sample scripts are available in `$BW_HOME/samples/core/admin`.
- The scripts that are generic for ActiveMatrix BusinessWorks are available in `$BW_HOME/scripts/admin`.

The scripts are updated to rely on the PATH setting to find the generic scripts. To make this easier to configure, after installation you can generate `$BW_HOME/scripts/bashrc.sh` that can be sourced from your `~/.bashrc`.

Source the `$BW_HOME/scripts/bashrc.sh` to set up the following environment variables required to run the scripts mentioned in the table below:

| Variable | Required |
|---|---|
| TIBCO_HOME | Yes |
| BW_HOME | Yes |
| TEA_HOME | No. But required if you run TIBCO® Enterprise Administrator on this machine. |
| EMS_HOME | No. But required if Enterprise Message Service™ is configured on this machine. |
| PATH | This variable is auto-populated based on the values set for the above variables. |

## Core Admin Scripts

The following table lists some of the available scripts; browse the folder to see the complete list.

*Core Admin Scripts*

| Script | Description | Script Location |
|---|---|---|
| `AppManage.sh` | This is a ActiveMatrix BusinessWorks 6.x utility program that emulates ActiveMatrix BusinessWorks 5.x AppManage commands.<br><br>The main purpose of this utility is to demonstrate how the AppManage commands from ActiveMatrix BusinessWorks 5.x translate to the corresponding ActiveMatrix BusinessWorks 6.x BWAdmin commands.<br><br>This utility creates a `cmd/AppManage_deploy.cmd` folder that contains BWAdmin commands and uses `bwadmin -f cmd/AppManage_deploy.cmd` to run it.<br><br>**Note:** Not all AppManage commands are implemented in this emulation utility.<br><br>**ActiveMatrix BusinessWorks Augmented Options**:<br><br>• `-appSpace` or `- a` - AppSpace name to be used for application lifecycle.<br>• `-profile` or `-p` - Configuration Profile to use for deployment. This profile must be available in the EAR file.<br>• `-profileFile`- Configuration profile file to use for deployment. | *$BW_HOME* `/samples/core/admin` |

| Script | Description | Script Location |
|---|---|---|
| | • `-debug` - Turn on debug tracing for this utility.<br>• `-sapp` - Single application per AppSpace deployment mode. Each AppSpace supports only one application deployment.<br>• `-mapp` - Multiple applications per AppSpace deployment mode. Each AppSpace supports one or more application deployments.<br>**Note:** ActiveMatrix BusinessWorks supports both `-sapp` and `-mapp` modes. The default is `-mapp` mode. | |
| `bootstrap.sh` | Usage: `bootstrap.sh [-h\|-help] [-clean] [-forceClean\|-force]`<br>This utility is a wrapper script around the following scripts:<br>• `killtea.sh`<br>• `killbwagent.sh`<br>• `teaclean.sh` only if `-clean` or `-forceClean` options is used<br>• `bwclean.sh` if and only if `-clean` or `-forceClean` options is used<br>• `genbwagentini.sh`<br>• `tea.sh`<br>• `bwagent.sh`<br>• `registeragent.sh`<br>`[-h]` or `[-help]` - Prints this usage message. | *$BW_HOME* `/scripts/admin` |

| Script | Description | Script Location |
|---|---|---|
| | `-clean` Cleans TIBCO Enterprise Administrator Server Data Store and ActiveMatrix BusinessWorks Domain Data Store.<br><br>**Note:** The -clean command on the Data Store is not reversible, so back up your data stores before using the command. Use this option carefully, as you may lose all your configurations if you do not have a backup.<br><br>`-forceClean` Same as `-clean`, except it avoids prompting the user to confirm with clean.<br><br>`-force` Same as `-forceClean`<br><br>`-forceclean` Same as `-forceClean`<br><br>This script assumes that the following products are installed correctly and the environment variables are set accordingly:<br><br>*TIBCO_HOME* = *TIBCO_HOME* directory where you installed ActiveMatrix BusinessWorks.<br><br>*TEA_HOME* = Parent directory to TIBCO Enterprise Administrator's `/bin` directory.<br><br>Supports generation of `bwagent.ini` file for either Database/TIBCO EMS™, or Database/TIBCO FTL® as the technology type. | |
| `bounce.sh` | This utility does the following:<br><br>1. Stops TIBCO Enterprise Administrator Server and BWAgent Processes. | *$BW_HOME* `/scripts/admin` |

| Script | Description | Script Location |
|---|---|---|
| | 2. Restarts TIBCO Enterprise Administrator Server and BWAgent Processes.<br>3. Registers BWAgent to TIBCO Enterprise Administrator Server.<br>`[-h]` or `[-help]` - Prints this help message and exits. | |
| `bounceagent.sh` | Ends and restarts BWAgent process.<br>`[-h]` or `[-help]` - Prints this help message and exits. | *$BW_HOME* `/scripts/admin` |
| `bwadmin.sh` | This is a utility script that wraps around the `bwadmin` executable.<br>`[-h]` or `[-help]` - Prints this help message and exits.<br>`[-network <bwagent Network Name>]` - Connects to a named BWAgent network. This is an optional argument.<br>By default, this script uses `$BW_HOME/config/bwagent.ini`<br>`[<bwadminArgs> ...]` - Use BWAdmin to run commands found in the input files.<br>Start BWAdmin in the interactive mode if cmdFile is not specified.<br>A BWAgent network name is a named directory under `${TIBCO_HOME}/bw/networks` and contains the corresponding `bwagent.ini`.<br>**How to Set Up a Newly Named Network**<br>1. Obtain a `bwagent.ini` created for | *$BW_HOME* `/scripts/admin` |

| Script | Description | Script Location |
| --- | --- | --- |
| | the named BWAgent network. For example, a named network called "acmeNetwork".<br>2. Create the `acmeNetwork` directory under `${TIBCO_HOME}/bw/networks`. For example, `mkdir ${TIBCO_HOME}/bw/networks/acmeNetwork`.<br>3. Copy `bwagent.ini` to the above directory.<br>4. Rerun `bwadmin.sh -network acmeNetwork`. | |
| `bwagent.sh` | This script starts the BWAgent in the background and waits until it is fully initialized, or the maxWait time (<n> * 2 sec) expires.<br>`[-h]` or `[-help]` - Prints this usage message.<br>`[-network <Network>]` - Starts up BWAgent using the configuration of a named network.<br>`[-maxWait <n>]` - Maximum amount of wait time (2 sec increment) for BWAgent startup success.<br>The default value for <n> is 30, which means 30 * 2 sec = 60 seconds | *$BW_HOME* `/scripts/admin` |
| `bwclean.sh` | This utility script cleans up ActiveMatrix BusinessWorks Domain Data and internal Data Store. The end effect of this clean up is similar to a fresh installation of ActiveMatrix BusinessWorks.<br>`[-force]` or `[-forceClean]` - Proceeds | *$BW_HOME* `/scripts/admin` |

| Script | Description | Script Location |
|---|---|---|
| | with wiping ActiveMatrix BusinessWorks Domain Data and internal Data Store without prompting user reconfirmation.<br><br>By default, the script prompts user confirmation. | |
| `configureBWEngineGroup.sh` | This utility configures AppNodes in a Domain or AppSpace to form a fault-tolerant group and cross-engine persistence<br><br>`[-h]` or `[-help]` - Prints this usage message.<br><br>`[-c]` or `[-clean]` - Cleans up and drops all the previously configured database tables.<br><br>Use this option carefully. This operation cannot be undone. Do not specify both `-setup` and `-cleanup` on the same run.<br><br>`[-s]` or `[-setup]` - Does the one-time setup of the BWEngine database. When this option is used, `-domain` and `-appspace` arguments are not needed and are not used even if specified. `${BW_HOME}/config/sqlscripts/<dbtype>/create.sql` is used to set up the database tables and configuration.<br><br>`[-b]` or `[-bootstrap]` - Does the cleanup, then set up.<br><br>`[-t]` or `[-dbtype]` - This is the default value is `postgresql`.<br><br>`-cf <config.sh>` - Sources configuration from the specified `<config.sh>` file. | *$BW_HOME*`/scripts/admin` |

| Script | Description | Script Location |
| --- | --- | --- |
| | By default, <*$BW_HOME*>/scripts/admin/config/bwengine-group-<dbtype>.sh<br><br>[-d] or [-domain] - Domain Name<br><br>[-a] or [-appspace <appspace>] - AppSpace Name. All AppNodes in the specified Domain and AppSpace are configured to form a Fault-Tolerant group and across engine persistence. | |
| deploy.sh | Usage: deploy.sh -ear <EARFile> [-h\|-help] [-domain <DomainName>] [-appspace <AppSpaceName>] [-redeploy\|-force] [-profile <Profile>]<br><br>Deploys the specified ActiveMatrix BusinessWorks EAR file into -domain <DomainName> -appspace <AppSpaceName><br><br>[-h] or [-help]- Prints this help message.<br><br>-ear <*EARFile*> - Enterprise Archive file to deploy<br><br>[-domain <DomainName>] - Domain Name - Optional parameter<br><br>If it is not specified, DomainName is computed from ${USER}-Domain<br><br>This utility creates the Domain if it does not exist.<br><br>[-appspace <AppSpaceName>] - AppSpace Name - Optional parameter<br><br>If it is not specified, AppSpaceName is computed from the name of the EAR file. | *$BW_HOME* /scripts/admin |

| Script | Description | Script Location |
| --- | --- | --- |
| | This utility creates the AppSpace and AppNode if they do not exist.<br><br>`[-redeploy\|-force]`: Redeploy if the application has been previously deployed.<br><br>The application is not redeployed if it exists and this option is not specified.<br><br>`[-profile <Profile>]`: Profile name to use for this deployment.<br><br>If it is not specified, the default profile as packaged in the Enterprise Archive file is used.<br><br>`[-mapp]`: Optional flag to set multiple applications per AppSpace mode. This is the default mode for ActiveMatrix BusinessWorks.<br><br>`[-debug]`: Prints debug tracing for this script `./deploy.sh` | |
| `genbwagentini.sh` | This script auto generates `${BW_HOME}/config/bwagent.ini` based on configurations defined in `./config/bwadmin-default- config.sh`<br><br>`-h` or `-help` - Prints this help message.<br><br>The following variables are required from `./config/bwadmin-default- config.sh`:<br><br>• BWAgentNetworkName - Name of BWAgent Network.<br>• BWMachines - Defined as a list of machine names (as obtained through `hostname -f`). If you have only one machine to configure, do not add it to this list because this | *$BW_HOME* `/scripts/admin` |

| Script | Description | Script Location |
| --- | --- | --- |
| | script auto-configures it as a standalone BWAgent Network.<br><br>This script uses `hostname -f` to determine the name of the machine it is run on. It then determines whether this machine is in the BWMachines list.<br><br>You can assume that the `discoveryURL` of the bwagent.ini is comparable to that of a Database server's URL, and `BWAgentNetworkName` is then comparable to the database name. You can configure both to access the specific instance of the database uniquely.<br><br>If the `KEEP_BWAGENT_INI` environment variable is defined, `bwagent.ini` generation is skipped.<br><br>You can edit either the `./config/bwadmin-default- config.sh` file, or make a copy of it, edit it, and then set an environment variable `BWADMIN_CONFIG` to point to it. For example, export `BWADMIN_CONFIG=~/config/bwadmin-my-config.sh`<br><br>Generates a `bwagent.ini` file for either Database/EMS, or Database/TIBCO FTL® as the technology type. | |
| `kill.sh` | Ends all processes that match the specified <*process name*><br><br>`-h` or `-help` - Prints this help message.<br><br><*process name*> - Name of the process you want to end. This script ends all instances of the `pid` that matches this name. | *`$BW_HOME`*`/scripts/admin` |

| Script | Description | Script Location |
|---|---|---|
| killall.sh | This script finds and ends all instances of processes that match the following names:<br>• tea<br>• bwagent<br>• bwappnode<br>• bwadmin<br>-h or -help - Prints this help message. | *$BW_HOME* /scripts/admin |
| killbwagent.sh | This script finds and ends all instances of processes that match bwagent.<br>-h or -help - Prints this help message. | *$BW_HOME* /scripts/admin |
| killbwappnodes.sh | This script finds and ends all instances of processes that match bwappnode.<br>-h or -help - Prints this help message. | *$BW_HOME* /scripts/admin |
| killtea.sh | This script finds and ends all instances of processes that match tea.<br>-h or -help - Prints this help message. | *$BW_HOME* /scripts/admin |
| killtibemsd64.sh | This script finds and ends all instances of processes that match tibemsd.<br>-h or -help - Prints this help message. | *$BW_HOME* /scripts/admin |
| recreatedb.sh | This script cleans up and recreates the PostgreSQL database needed by the ActiveMatrix BusinessWorks BookStore REST sample available in: ${BW_HOME}/samples/binding/rest/BookStore<br>-h and -help - Prints this help message. | *$BW_HOME* /samples/core/admin |
| registeragent.sh | This utility registers the local BWAgent | *$BW_* |

| Script | Description | Script Location |
| --- | --- | --- |
| | with the TIBCO Enterprise Administrator server.<br><br>`-h` or `-help` - Prints this help message.<br><br>This utility assumes that the following environment variables have been set:<br><br>`export TIBCO_HOME="<Where ActiveMatrix BusinessWorks is installed>"`<br><br>At least one of the following environment variables is set:<br><br>`export TEA_HOME="Where TIBCO Enterprise Administrator is installed in the form of $TIBCO_HOME/tea/<version>"`<br><br>Or,<br><br>`export TEA_HOSTNAME=<HostName>`<br><br>If the *TEA_HOSTNAME* environment variable is set, it assumes that the TIBCO Enterprise Administrator server is running remotely from the local BWAgent instance.<br><br>If the *TEA_HOSTNAME* environment variable is not set, this script registers the local BWAgent to the locally running TIBCO Enterprise Administrator server. | *HOME* `/scripts/admin` |
| `runAcme.sh` | Creates <domain> and deploys all EAR files found under `${BW_HOME}/samples/core/admin/ears/acme`.<br><br>`-h` or `-help`: Displays this usage message<br><br>`<domain>` - can be "Acme-QA-Domain" or "Acme-UAT-Domain". When not specified, the default is "Acme-QA-Domain" | *$BW_HOME* `/samples/core/admin` |

| Script | Description | Script Location |
|---|---|---|
| | `<mode>` - `[-sapp ]` or `[-mapp]`<br><br>`-sapp`- Single App AppSpace deployment mode. Each AppSpace supports only one application deployment.<br>`-mapp` - Multiple App AppSpace deployment mode. Each AppSpace supports one or more application deployments.<br><br>**Note:** ActiveMatrix BusinessWorks supports both `-sapp` and `-mapp` modes. The default is `-mapp` mode.<br><br>This script dynamically creates a `bwadmin` command file in `cmd/<domain>-<mode>.cmd` and runs it. | |
| `runAll.sh` | This utility is a wrapper script that performs the following:<br>• bootstrap.sh - only if running in a single machine setup<br>• runBookStore.sh<br>• runSamples.sh<br>• runAcme.sh -domain Acme-QA-Domain<br>• runAcme.sh -domain Acme-UAT-Domain<br><br>`-h` or `-help` - Displays this usage message and exits<br><br>`-clean` - Cleans the TIBCO Enterprise Administrator Server Data Store and ActiveMatrix BusinessWorks Domain Data Store. | *$BW_HOME* `/samples/core/admin` |

| Script | Description | Script Location |
|---|---|---|
| | The Data Store clean is not reversible. Make sure you back up your data stores before running this command. Use this option with utmost care, otherwise you risk losing all your configurations.<br><br>`-forceClean` - Same as `-clean`, except it avoids prompting you to confirm with clean.<br><br>`-force` - Same as `-forceClean`<br><br>`<mode>` - `[-sapp \| -mapp]`<br><br>`-sapp` - Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment.<br><br>`-mapp` - Multiple Applications per AppSpace deployment mode. Each AppSpace supports one or more application deployments.<br><br>**Note:** ActiveMatrix BusinessWorks supports both `-sapp` and `-mapp` modes. The default is `-mapp` mode.<br><br>Generates the `bwagent.ini` file for either Database/EMS™, or Database/FTL® technology type. | |
| `runBookStore.sh` | Creates BookStore-Domain and deploys all EAR files found under `${BW_HOME}/samples/core/admin/ears/bookstore`<br><br>`-h` or `-help` - Displays this usage message.<br><br>`<mode>` - `[-sapp \| -mapp]` | *`$BW_HOME`*`/samples/core/admin` |

| Script | Description | Script Location |
|---|---|---|
| | -sapp - Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment.<br><br>-mapp - Multiple Application per AppSpace deployment mode. Each AppSpace supports one or more application deployments.<br><br>**Note:** ActiveMatrix BusinessWorks supports both -sapp and -mapp modes. The default is -mapp mode.<br><br>This script dynamically creates a bwadmin cmd file in cmd/Samples-Domain-<mode>.cmd and runs it. | |
| runSamples.sh | Creates Samples-Domain and deploys all EAR files found under ${BW_HOME}/samples/core/admin/ears/samples<br><br>-h or -help - Displays this usage message.<br><br><mode>: [-sapp] or [-mapp]<br><br>-sapp: Single Application per AppSpace deployment mode. Each AppSpace supports only one application deployment.<br><br>-mapp: Multiple Application per AppSpace deployment mode. Each AppSpace supports one or more application deployments. | *$BW_HOME* /samples/core/admin |

| Script | Description | Script Location |
|---|---|---|
| | **Note:** ActiveMatrix BusinessWorks supports both `-sapp` and `-mapp` modes. The default is `-mapp` mode. This script dynamically creates a `bwadmin` command file in `cmd/Samples-Domain-<mode>.cmd` and runs it. | |
| `showprocs.sh` | Shows process ID and complete binary path of all processes required in ActiveMatrix BusinessWorks:<br>• `tibemsd`<br>• `tea`<br>• `bwagent`<br>• `bwappnode`<br>• `bwadmin` | *$BW_HOME* `/scripts/admin` |
| `tea.sh` | This script starts TIBCO Enterprise Administrator in the background and waits until it is completely initialized, or the maxWait time (<n> * 2 sec) expires.<br>`-h` or `-help` - Prints this usage message.<br>`[-maxWait <n>]` - Max number of wait time (2 sec increment) for TIBCO Enterprise Administrator Server startup success.<br>The default value for <n> is 30, which means 30 * 2 sec = 60 seconds. | *$BW_HOME* `/scripts/admin` |
| `teaclean.sh` | This utility script cleans the TIBCO Enterprise Administrator Server's configuration data store.<br>The end effect of this clean up is similar to | *$BW_HOME* `/scripts/admin` |

| Script | Description | Script Location |
|---|---|---|
| | a fresh installation of TIBCO Enterprise Administrator.<br><br>`-h` or `-help` - Prints this usage message<br><br>`-force` or `-forceClean` - Proceeds with wiping ActiveMatrix BusinessWorks Domain Data and internal data store without prompting user reconfirmation.<br><br>By default, the script prompts user confirmation. | |
| `tibemsd64.sh` | This script starts `tibemsd64` in the background and waits until it is completely initialized, or the maxWait time (<n> * 2 sec) expires.<br><br>`-h` or `-help` - Prints this usage message<br><br>`[-maxWait <n>]` - Max number of wait time (2-seconds increment) for `tibemsd64` startup success.<br><br>The default value for <*n*> is 30, which means 30 * 2 sec = 60 seconds.<br><br>This script is only supported on UNIX-based systems.<br><br>For Windows, use Windows Systems Services to start or stop `tibemsd64`. | *$BW_HOME* `/scripts/admin` |

**Note:** Each `runAcme.sh`, `runBookStore.sh`, `runSamples.sh`, `deploy.sh`, and `AppManage.sh` generates `bwadmin` commands before execution.

The generated `bwadmin` command files are found under the `cmd` subdirectory.

# 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 ActiveMatrix BusinessWorks™ page:

- *TIBCO ActiveMatrix BusinessWorks™ Release Notes*
- *TIBCO ActiveMatrix BusinessWorks™ Installation*
- *TIBCO ActiveMatrix BusinessWorks™ Application Development*
- *TIBCO ActiveMatrix BusinessWorks™ Bindings and Palettes Reference*
- *TIBCO ActiveMatrix BusinessWorks™ Concepts*
- *TIBCO ActiveMatrix BusinessWorks™ Error Codes*
- *TIBCO ActiveMatrix BusinessWorks™ Getting Started*
- *TIBCO ActiveMatrix BusinessWorks™ Maven Plug-in*
- *TIBCO ActiveMatrix BusinessWorks™ Migration*
- *TIBCO ActiveMatrix BusinessWorks™ Performance Benchmarking and Tuning*
- *TIBCO ActiveMatrix BusinessWorks™ REST Implementation*
- *TIBCO ActiveMatrix BusinessWorks™ Refactoring Best Practices*
- *TIBCO ActiveMatrix BusinessWorks™ 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.