



# **TIBCO ActiveMatrix® Service Grid**

## **Administration Tutorials**

*Software Release 3.4  
April 2019*

## Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

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

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

TIBCO, the TIBCO logo, Two-Second Advantage, TIB, Information Bus, ActiveMatrix, Business Studio, Enterprise Message Service, Hawk, and Rendezvous are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

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

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. Please see the readme.txt file for the availability of this software version on a specific operating system platform.

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

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

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

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2010-2019. TIBCO Software Inc. All Rights Reserved.

# Contents

---

<b>TIBCO Documentation and Support Services</b>	<b>6</b>
<b>Overview</b>	<b>8</b>
<b>Tutorial Prerequisites</b>	<b>9</b>
Set Up the Runtime Environment	9
Create a UDDI Server in Administrator	9
Configure Administrator Command-Line Properties	10
Set Up the Service Invocation Environment	10
<b>How to Configure Logging</b>	<b>11</b>
Creating a Logging Appender	11
GUI	11
CLI	12
Navigating to a Logging Configurations List	13
Creating a Logging Configuration for a Host or a Node	13
GUI	13
CLI	14
Applying a Logging Configuration	14
GUI	14
CLI	14
<b>How to Deploy and Run the Hello World Application</b>	<b>16</b>
Completing Hello World Prerequisites	16
Creating the Hello World Application	16
Creating an HTTP Connector Resource Template	16
Creating and Installing the HTTP Connector Resource Instance	17
Distributing the Hello World Application	17
Deploying and Starting the Hello World Application	18
Generating the Hello World WSDL File	18
Invoking the Hello World Service	18
Viewing the Hello World Service in the UDDI Registry Server	19
<b>How to Deploy and Run the Enhanced Hello World Application</b>	<b>21</b>
Completing Enhanced Hello World Prerequisites	21
Deploying and Starting the Date Manager Application from the CLI	21
Creating the Hello World Application	22
Creating an HTTP Connector Resource Template	22
Creating and Installing the HTTP Connector Resource Instance	22
Creating and Installing the HTTP Client Resource Instance	23
Deploying and Starting the Enhanced Hello World Application	23

Distributing the Enhanced Hello World Application .....	23
Generating the Hello World WSDL File .....	23
Invoking the Hello World Service .....	23
<b>How to Deploy and Run the Java8 Hello World Application .....</b>	<b>24</b>
<b>How to Deploy and Run REST Binding Samples .....</b>	<b>25</b>
<b>How to Deploy and Run the Phonebook Application .....</b>	<b>26</b>
Completing Phonebook Prerequisites .....	26
Creating the Phonebook Application .....	26
Creating an HTTP Connector Resource Template .....	27
Creating and Installing the HTTP Connector Resource Instance .....	27
Creating and Installing the JDBC Shared Resource Instance .....	27
Distributing the Phonebook Application .....	27
Deploying and Starting the Phonebook Application .....	27
Generating the Phonebook WSDL File .....	28
Invoking the Phonebook Service .....	28
<b>How to Deploy and Run the Hello World Web Application .....</b>	<b>29</b>
Completing Hello World Web Application Prerequisites .....	29
Creating the Hello World Web Application .....	29
Creating an HTTP Connector Resource Template .....	29
Creating and Installing the HTTP Connector Resource Instance .....	29
Distributing the Hello World Web Application .....	30
Invoking the Hello World Web Application .....	30
Deploying and Starting the Hello World Web Application .....	30
<b>How to Deploy and Run the Content-Based Routing Mediation Application .....</b>	<b>31</b>
Creating the Routing and Target Service Mediation Applications .....	31
Creating HTTP Connector Resource Templates .....	31
Creating and Installing Resource Instances .....	32
Distributing the Routing and Target Service Applications .....	32
Deploying and Starting the Routing and Target Service Applications .....	33
Generating the Routing WSDL File .....	33

# TIBCO Documentation and Support Services

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

## Product-Specific Documentation

Documentation for TIBCO ActiveMatrix® Service Grid is available on the <https://docs.tibco.com/products/tibco-activematrix-service-grid> page.

Use of the following features, installation profiles and development tools requires a TIBCO ActiveMatrix Service Grid license:



- TIBCO ActiveMatrix Policy Director Governance, TIBCO ActiveMatrix SPM Dashboard, and TIBCO ActiveMatrix SPM Runtime Server profiles; and
- TIBCO ActiveMatrix Service Grid development tools for Java, Webapp and Spring components.

Customers with only a TIBCO ActiveMatrix Service Bus license are not licensed to use these features, tools or profiles.

The following documents form the documentation set:

- *TIBCO ActiveMatrix Service Grid Concepts*: Read this manual before reading any other manual in the documentation set. This manual describes terminology and concepts of the platform. The other manuals in the documentation set assume you are familiar with the information in this manual.
- *TIBCO ActiveMatrix Service Grid Development Tutorials*: Read this manual for a step-by-step introduction to the process of creating, packaging, and running composites in TIBCO Business Studio.
- *TIBCO ActiveMatrix Service Grid Composite Development*: Read this manual to learn how to develop and package composites.
- *TIBCO ActiveMatrix Service Grid Java Component Development*: Read this manual to learn how to configure and implement Java components.
- *TIBCO ActiveMatrix Service Grid Mediation Component Development*: Read this manual to learn how to configure and implement Mediation components.
- *TIBCO ActiveMatrix Service Grid Mediation API Reference*: Read this manual to learn how to develop custom Mediation tasks.
- *TIBCO ActiveMatrix Service Grid Spring Component Development*: Read this manual to learn how to configure and implement Spring components.
- *TIBCO ActiveMatrix Service Grid WebApp Component Development*: Read this manual to learn how to configure and implement Web Application components.
- *TIBCO ActiveMatrix Service Grid REST Binding Development*: Read this manual to learn how to configure and implement REST components.
- *TIBCO ActiveMatrix Service Grid Administration Tutorials*: Read this manual for a step-by-step introduction to the process of creating and starting the runtime version of the product, starting TIBCO ActiveMatrix servers, and deploying applications to the runtime.
- *TIBCO ActiveMatrix Service Grid Administration*: Read this manual to learn how to manage the runtime and deploy and manage applications.

- *TIBCO ActiveMatrix Service Grid Hawk ActiveMatrix Plug-in*: Read this manual to learn about the Hawk plug-in and its optional configurations.
- *TIBCO ActiveMatrix Service Grid Policy Director Governance Custom Actions*: Read this manual to learn how you can configure and enforce policies for ActiveMatrix and external services hosted in third party containers, using TIBCO ActiveMatrix Policy Director Governance.
- *TIBCO ActiveMatrix Service Grid Service Performance Manager API Reference*: Read this manual to learn how to use the SPM APIs.
- *TIBCO ActiveMatrix Service Grid Error Codes*: Read this manual to know more about the error messages and how you could use them to troubleshoot a problem.
- *TIBCO ActiveMatrix Service Grid Installation and Configuration*: Read this manual to learn how to install and configure the software.
- *TIBCO ActiveMatrix Service Grid Security Guidelines*: Read this manual to learn more about security guidelines and recommendations for TIBCO ActiveMatrix Service Grid.
- *TIBCO ActiveMatrix Service Grid Release Notes*: Read this manual for a list of new and changed features, steps for migrating from a previous release, and lists of known issues and closed issues for the release.

### **How to Contact TIBCO Support**

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking Register on the website.

### **How to Join TIBCO Community**

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](https://community.tibco.com). For a free registration, go to <https://community.tibco.com>.

# Overview

These tutorials demonstrate how to deploy and run SOA applications developed in TIBCO Business Studio in TIBCO ActiveMatrix Administrator and send SOAP requests from the TIBCO Business Studio Web Services Explorer.

Tutorial Facts lists the location of the sample projects, the names of the deployed applications, and the ports of the HTTP connector resources used by the applications. Before proceeding with a tutorial, complete the steps in [Tutorial Prerequisites](#):

## *Tutorial Facts*

Tutorial	Sample Project Location	Deployed Applications	Ports
Hello World	TIBCO_HOME/amx/<version>/samples/java/helloworld1.zip	helloworld1	9095
Enhanced Hello World	TIBCO_HOME/amx/<version>/samples/java/helloworld2.zip	helloworld2 datemanager	9096 9097
Phonebook	TIBCO_HOME/amx/<version>/samples/java/phonebook.zip	phonebook	9098
Hello World Web Application	TIBCO_HOME/amx/<version>/samples/webapp/helloworld.zip	helloworld	9099
Content-Based Routing	TIBCO_HOME/amx_it_mediation/<version>/samples/ContentBasedRouting TIBCO_HOME/amx_it_mediation/<version>/samples/TargetService	querygds targetservice	8777 8666



# Tutorial Prerequisites

---

Complete the tutorial prerequisite tasks.


## Set Up the Runtime Environment

### Procedure

1. Create and start an Administrator server as described in the installation manual for your product. When you create the Administrator server, create a development node and accept the default values for the environment and node names.
2. Invoke the Administrator GUI. The URL is `http://machinename:port/amxadministrator/loginForm.jsp`, where *machinename* is the machine on which you created the Administrator server and *port* is the port on which Administrator clients access the Administrator server.
3. Log in with the credentials (specified when you created the Administrator server. After successful login, Administrator displays the Welcome screen.
4. If you want to register deployed services in a UDDI registry, install and start TIBCO ActiveMatrix Runtime UDDI Server.

## Create a UDDI Server in Administrator

### Procedure

1. In the Administrator GUI, select **Infrastructure > Servers**.
2. Click  **New**.  
The New Server dialog displays.
3. In the Name field, type SOAUDDI.
4. In the Type field, ensure that **UDDI** is selected.
5. In the UDDI Server Type field, select **TIBCO**.
6. In the Hostname/IP field, type the name of the host on which the UDDI server is running.
7. In the Port field, type 58080.
8. In the Username and Password fields, type admin and admin.
9. Click the **Test Connection** button.
10. In the Publication Business drop-down list, type Services deployed in TIBCO ActiveMatrix.
11. Check the **Automatic Publication** checkbox.
12. Click **Save**.

### Result

When you deploy an application, Administrator will publish the endpoints exposed by the application in the UDDI server.

## Configure Administrator Command-Line Properties

### Procedure

1. In a terminal window, open `CONFIG_HOME/admin/enterpriseName/samples/remote_props.properties` in a text editor.
  - a) Replace the host portion of the value of the `adminURL` property with the host on which the Administrator server is running.
  - b) Replace the port portion of the value of the `adminURL` property with the port on which the Administrator server is running.
  - c) Replace the username and password properties with the credentials you specified when you created the Administrator server.
2. Save the properties file.

## Set Up the Service Invocation Environment

Create a project in TIBCO Business Studio to contain the concrete WSDL files of deployed service bindings. To test a deployed service, you open the concrete WSDL file in the TIBCO Business Studio Web Services Explorer and generate SOAP requests.

### Procedure

1. Start TIBCO Business Studio.
2. Select **File > New > Project...**  
The New Project wizard displays.
3. Select **General > Project** and click **Next >**.
4. In the File name field, type `ConcreteWSDLs` and click **Finish**.  
A general project named `ConcreteWSDLs` displays in the Project Explorer view.

# How to Configure Logging

Instruction on configuring logging is provided in the following topics.

## Creating a Logging Appender

You can create a logging appender from the GUI or by using the CLI. Three types of appenders are supported: Clear Text File, CBE XML File, and JMS.

### GUI

#### Procedure

1. Select **Shared Objects > Logging Appendors**.
2. Click **New**.  
The New Logging Appender dialog displays with the drop-down list of the logging appender type expanded.
3. Select an appender type from the Type list.
  - **JMS Appender** - Append events to a log service.
  - **CBE XML File Appender** - Appends events to a file in [Common Base Event \(CBE\)](#) format.
  - **Clear Text File** - Appends events to a file in clear text format.

The dialog redraws with the appender-specific fields.

4. Accept the default name or type a name for the appender in the Name field.
5. Fill in the fields and click **Save**.  
The appender is added to the Logging Appendors table and is selected.

#### Result

The screenshot shows the 'New Logging Appender' dialog box. The 'Name' field contains 'DevNodeLog'. The 'Type' dropdown menu is open, showing 'CBE XML File Appender'. The 'Description (optional)' field is empty. The 'File Path' field contains './logs/DevLogs.log'. The 'Max File Size (MB)' field contains '10'. The 'Max Backup Index' field contains '5'. The 'Save' and 'Revert' buttons are at the bottom.

## CLI

### Procedure

1. In the data file, `logappender_data.xml`, specify the type of the appender in the `xsi:type` attribute.  
File Log Appender

```
<LogAppender
  xsi:type="amxdata:FileLogAppender"
  name="HelloWorldFileAppender"
  description="This is File LogAppender"
  filePath="C:/amx-3admin/tibco/cfgmgt/tibcohost/Admin-amxadmin-instanceOne/
  nodes/DevNode/logs/HelloWorld.log"
  maxSize="10000" maxBackupNum="5"/>
```

#### CBE File Appender

```
<LogAppender xsi:type="amxdata:FileLogAppender"
  name="myFileLogAppender"
  description="This is File LogAppender"
  filePath="C:/amx-3admin/tibco/cfgmgt/tibcohost/Admin-amxadmin-instanceOne/
  nodes/DevNode/logs/HelloWorld-CBE.log"
  type="cbe"
  maxSize="1000"
  maxBackupNum="5"/>
```

#### JMS Appender without payload support

```
<LogAppender xsi:type="amxdata:JmsLogAppender"
  name="myJmsLogAppender"
  description="This is Jms LogAppender without payload support"
  jmsConnectionFactoryName="cl_logservice_jmsConnectionFactory"
  jmsConnectionName="cl_logservice_jndiConnectionConfig"
  jmsDestination="cl_logservice_jmsConnectionDestination"
  type="jndi"
  sync="true"/>
```

#### JMS Appender with payload support

```
<LogAppender xsi:type="amxdata:JmsLogAppender"
  name="myJmsLogAppender"
  description="This is Jms LogAppender with payload support"
  jmsConnectionFactoryName="cl_logservice_jmsConnectionFactory"
  jmsConnectionName="cl_logservice_jndiConnectionConfig"
  jmsDestination="cl_logservice_jmsConnectionDestination"
  type="jndi"
  sync="true"
  payloadURL="c:/payloadURL"
  sharedDiskURL="c:/sharedDiskURL"/>
```

2. In the Build file, `logappender_build.xml`, for the `AMXAdminTask` element, set the `action` attribute to `add` and the `objectSelector` attribute to `LogAppender`.  

```
<AMXAdminTask action="add" objectSelector="LogAppender"/>
```
3. Invoke the command-line interface on the build file (`logappender_build.xml`) using the following command:  

```
ant -f logappender_build.xml create
```

### Result

Refer to *Composite Development* for information on how to retrieve log entries from the destination queue of a JMS appender.

## Navigating to a Logging Configurations List

### Procedure

1. Navigate to a list of hosts, nodes, or applications.
2. Select a host, node, or application.
3. Click the **Configuration** tab.
4. Click the **Logging** link.  
The logging configurations table for the host, node, or application displays.

## Creating a Logging Configuration for a Host or a Node

You can create a logging configuration for a host or node from the GUI or by using the CLI. Basic Mode and Advanced Mode are available for setting the logging. In Basic Mode, you can choose a log level for the File and Jms appender. In Advanced Mode, you have the option to set up a new appender.

### GUI

#### Procedure

1. Click **Infrastructure** and select Hosts or Nodes.  
Hosts or Nodes panel appears with a list.
2. Select a host or node.  
Details of the host or node displays.
3. Click **Configuration > Logging**.
4. Click **Basic Mode** or **Advanced Mode**.

Mode	Procedure
<b>Basic</b>	<ol style="list-style-type: none"> <li>1. Click <b>Add</b>. A row is added to the list.</li> <li>2. In the Logger Name column, type a logging configuration name.</li> <li>3. Select the FileAppender log level.</li> <li>4. Select JmsAppender log level.</li> <li>5. Click <b>Save And Apply</b>, or <b>Save</b>, or <b>Revert</b>.</li> </ol>
<b>Advanced</b>	<ol style="list-style-type: none"> <li>1. Click <b>Add</b>. A row is added to the list.</li> <li>2. In the Logger Name column, type a logging configuration name or select from the list.</li> <li>3. In the Additivity column, select an additivity.</li> <li>4. Click <b>Set Appender</b>. A row is added to the list.</li> <li>5. In the Appender column, select an appender from the list.</li> <li>6. In the Level column, select a logging level.</li> <li>7. If you want to add a new appender, click New Appender. If not, go to the next step. See <a href="#">Creating a Logging Appender</a>.</li> <li>8. Click <b>Apply</b> or <b>Save</b> or <b>Revert</b>.</li> </ol>

## CLI

### Procedure

1. In the data file (host\_data.xml or node\_data.xml) specify Logger, AppenderRef, and Appender elements.

```
<Logger xsi:type="amxdata:Logger" name="HelloWorldLogger" additivity="false">
  <AppenderRef xsi:type="amxdata:AppenderRef" effectivelevel="INFO">
    <Appender xsi:type="amxdata_reference:LogAppender_reference"
name="HelloWorldFileAppender"/>
  </AppenderRef>
</Logger>
```

2. In the build file (host\_build.xml or node\_build.xml) set the action attribute of the AMXAdminTask element to add or set and the objectSelector attribute to *Path/Logger*, where *Path* is the navigation path to the logger. For example, to set the logging configurations for all application loggers in a data file (host\_data.xml or node\_data.xml), action is set and objectSelector is Environment/Application/Logger:

```
<AMXAdminTask action="set" objectSelector="Environment/Application/Logger"/>
```

3. Invoke the command-line interface on the build file (host\_build.xml or node\_build.xml) with target name=setLog.

## Applying a Logging Configuration

You can apply a logging configuration from the GUI or from the CLI.

## GUI

### Procedure

1. Select the object for which logging is being configured.
2. Navigate to a logging configurations list and click a logging configuration.
3. Click **Apply**.

### Result

The logging configuration is propagated to the object.

## CLI

### Procedure

1. In the data file (host\_data.xml or node\_data.xml), specify a Logger definition in the full format. An example for a Node is shown below. In this example, the Node element contains a logging configuration for a node named admin01-node. The logging configuration named com.tibco specifies an appender that logs all Debug, Info, Warn, Error and Fatal events to a file specified in the logging appender named node\_file. The log messages are passed to the root parent logging configuration.

```
<Node xsi:type="amxdata:Node" name="admin01-node">
  <Logger xsi:type="amxdata:Logger" name="com.tibco" additivity="true">
    <AppenderRef xsi:type="amxdata:AppenderRef" effectiveLevel="DEBUG">
      <Appender xsi:type="amxdata_reference:LogAppender_reference"
name="node_file"/>
    </AppenderRef>
  </Logger>
</Node>
```

2. In the build file (`host_build.xml` or `node_build.xml`), set the:
  - `action` attribute of the `AMXAdminTask` element to `deploy` or `deployLog`
  - `objectSelector` attribute to `Environment/Object`, where *Object* is the object for which logging is being configured

An example for a Node is shown below.

```
<AMXAdminTask  
  action="deploy"  
  objectSelector="Environment/Node"/>
```

3. Invoke the command-line interface on the build file (`host_build.xml` or `node_build.xml`) with `target name=deployLog`.  
The logging configuration is propagated to the object.

# How to Deploy and Run the Hello World Application

---

This how-to describes how to deploy and run the Hello World application in TIBCO ActiveMatrix Administrator.

## Completing Hello World Prerequisites

### Procedure

- In TIBCO Business Studio, create a distributed application archive (DAA) by following the instructions in the How To Create and Package a Hello World Application TIBCO Business Studio cheat sheet or extract `jv.helloworld1.soa/Deployment Artifacts/jv.helloworld1.soa.daa` from `helloworld1.zip`.


## Creating the Hello World Application

### Procedure

1. Click the **Applications** tab.
2. Click the **New, New Application**.  
The New Application wizard displays.
3. Select the radio button for **Create the application from a DAA or EAR file** and click the **Browse** button.
4. Navigate to the folder containing the file `jv.helloworld1.soa.daa`, click the DAA, and click **Open**.
5. In the Application Name field, replace the default name with `helloworld1`.
6. In the Environment Name drop-down list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Choose to import the listed feature and click **Next**.
10. Check the checkbox for the node where you want to deploy your application and click **Next**.
11. Accept default values for the configuration properties and click **Next**.  
The validation screen displays.
12. Click **Save and Exit**.

## Creating an HTTP Connector Resource Template

### Procedure



1. Select **Shared Objects > Resource Templates**.
2. Click  **New**.  
The Add Resource Template dialog displays.
3. In the Type drop-down list, select **HTTP Connector**.  
The HTTP connector configuration fields display.
4. In the Name field, replace the default name with `hello1Connector`.



5. Accept the default Machine Name.
6. In the Port field, type 9095.
7. Accept default values for all other fields.
8. Click **Save**.

## Creating and Installing the HTTP Connector Resource Instance

### Procedure


1. Select **Infrastructure > Hosts**.
2. In the Hosts list, click **SystemHost**.  
The host details display below the list.
3. Click the **Resource Instances** tab.  
The All Instances list displays.
4. Click **New**.  
The New Resource Instances dialog displays.
5. In the resource templates list at the top, click the **hello1Connector** row. The Instance Name field is set to hello1Connector.
6. In the Available Nodes list at the bottom, click **DevNode** and click .  
The node moves to the Selected Nodes list.
7. Click **Save and Install**.  
(OR)  
Click **Save and Close**. In the **Resource Instances** tab, select the newly created resource instance and click **Install**.
8. Click  until the status changes to Running.

## Distributing the Hello World Application

If the application is created using the ActiveMatrix Administrator GUI, it is mapped to selected Nodes during creation time itself. If you do not want to change the mapping or distribution of the application, the following steps are not required.



If the application is created using the ActiveMatrix Administrator CLI, change the mapping or distribution as shown below. This mapping or distribution can also be changed using the CLI.

### Procedure

1. Click **Applications**.
2. In the Applications list, click **helloworld1**.
3. Click the **Distribution** tab.
4. Click **helloworld1**.
5. In the **Available Nodes** list, click **DevNode** and click .  
The node moves to the **Selected Nodes** list.
6. Click **Save**.

## Deploying and Starting the Hello World Application

### Procedure

1. Click **Deploy**.  
The application is deployed and started. If you have completed [Create a UDDI Server in Administrator](#), the WSDL file is published in the UDDI server.
2. Click  to refresh the display.  
The Runtime State changes to Starting and the Action History changes to In Progress (Deploy).
3. Click  to refresh the display.  
The Runtime State changes to Running and the Action History changes to Deploy with Start Successful.

## Generating the Hello World WSDL File

### Procedure

1. Click the **Configuration** tab.
2. Expand the **jb.helloworld1.soa** node.
3. Click the **HelloWorldPT** service.  
The service details display on the right.
4. Click the **Bindings** link.  
The service's binding details display.
5. In the list of bindings, expand **HelloWorld1SOAP**.  
The list of nodes on which the binding is deployed displays below.
6. In the node list, click **DevNode**.  
The Generate WSDL button activates.
7. Click **Generate WSDL**.  
The WSDL document representing the deployed service displays.
8. Save the WSDL file with the name **helloworld1.wsdl** in the ConcreteWSDLs project in TIBCO Business Studio.

### Result

The WSDL file is created.

By default HTTP connectors are created with the machine name 0.0.0.0. For example,

```
<wsdl:port name="HelloWorldSOAP" binding="tns:HelloWorldSOAP">
  <soap:address location="http://0.0.0.0:9095/helloWorldPT/" />
</wsdl:port>
```

The machine name has to be updated before using the WSDL file.

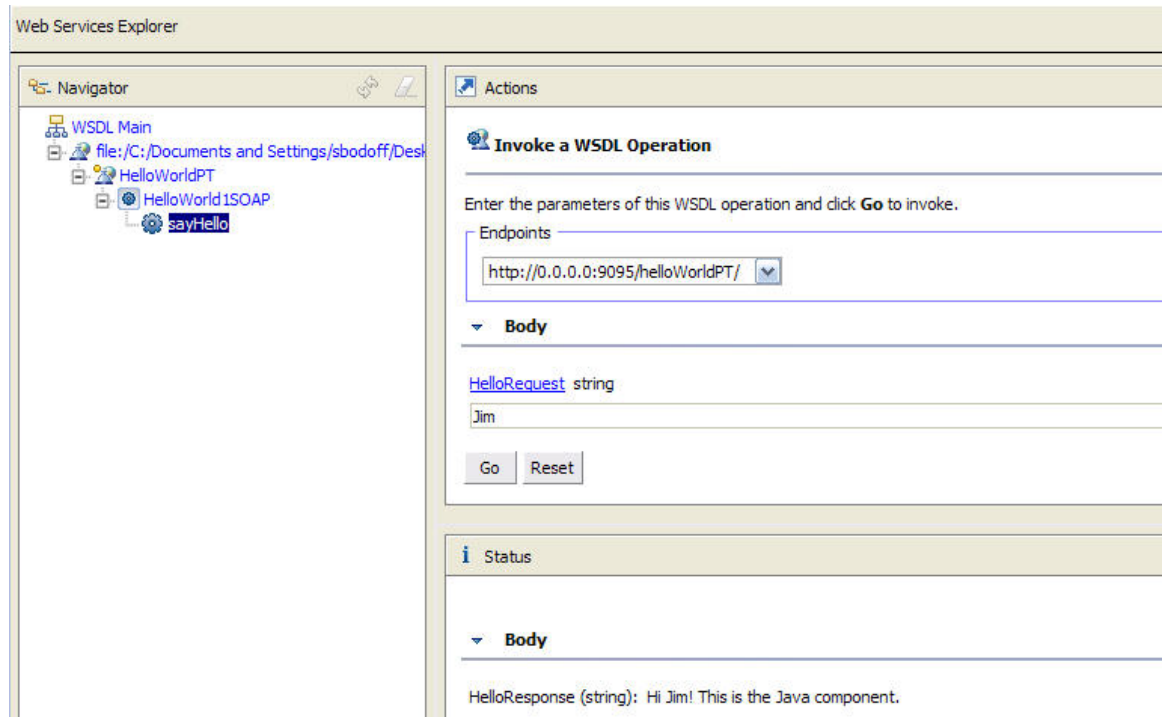
## Invoking the Hello World Service

### Procedure

1. In TIBCO Business Studio, right-click **Concrete WSDLs/helloworld1.wsdl** and select **Web Services > Test with Web Services Explorer**.

The WSDL file opens in the Web Services Explorer.

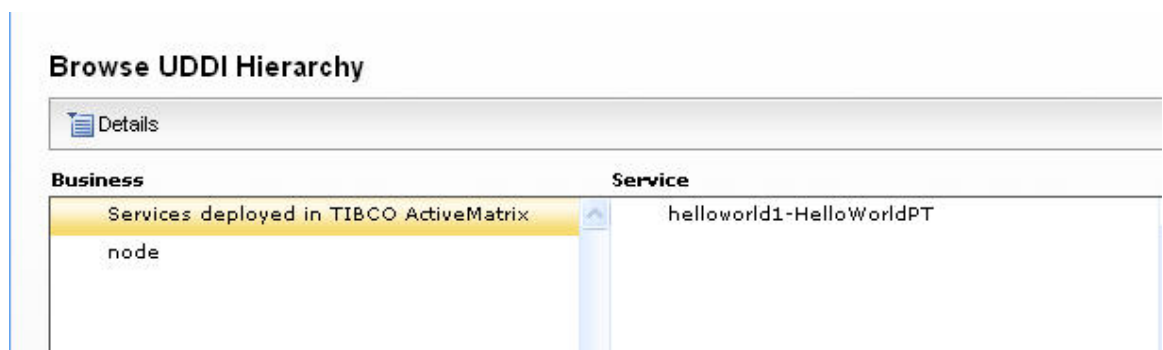
2. In the Navigator pane, expand the **HelloWorld1SOAP** node.
3. Click the **sayHello** node.  
The Invoke a WSDL Operation displays on the right with the newly created endpoint selected in the Endpoints drop-down list.
4. In the Actions pane, type **Jim** in the HelloRequest field.
5. Click **Go**.



## Viewing the Hello World Service in the UDDI Registry Server

### Procedure

1. Start the TIBCO ActiveMatrix UDDI Service Console GUI.  
The Services Deployed in TIBCO ActiveMatrix business displays in the Service Console.



2. Click **Details**.  
The details of the services deployed in TIBCO ActiveMatrix business will display.

3. In tree on the left, expand the **helloworld1-HelloWorldPT** node and click the binding template key under the node.  
In the right pane, the General Information tab of the Binding Template displays the service access point `http://0.0.0.0:9095/helloWorldPT/`.

#### Services deployed in TIBCO ActiveMatrix

The screenshot shows the TIBCO ActiveMatrix console interface. On the left, a tree view displays the hierarchy: 'Services deployed in TIBCO ActiveMa' (partially visible), 'helloworld1-HelloWorldPT', and a selected binding template 'uddi:mydomain.com:afe0100a-d8b0-4350-b6be-dbc8fa0087d6'. The right pane shows the 'Binding Template - uddi:mydomain.com:afe0100a-d8b0-4350-b6be-dbc8fa0087d6' with the 'General Information' tab selected. The tab displays the following details:

- Binding Template Key:** `uddi:mydomain.com:afe0100a-d8b0-4350-b6be-dbc8fa0087d6`
- Description:** (empty)
- Service Key:** `uddi:mydomain.com:60740c04-69cd-4522-920c-8504c64ae5d0`
- Service Name:** `helloworld1-HelloWorldPT`
- Access Point:** `http://0.0.0.0:9095/helloWorldPT/`
- Access Point Type:** `endpoint`

Below this information, a section titled 'tModel Instances contained in Binding Template' contains a table with two columns: 'tModel Instance Key' and 'Description'.

tModel Instance Key	Description
<a href="#">uddi:mydomain.com:bb74ef59-6446-4442-98aa-003fc8d3</a>	
<a href="#">uddi:uddi.org:transport:http</a>	

# How to Deploy and Run the Enhanced Hello World Application

This how-to describes how to deploy and run the enhanced Hello World application in TIBCO ActiveMatrix Administrator.

## Completing Enhanced Hello World Prerequisites

### Procedure

1. In TIBCO Business Studio, create a distributed application archive (DAA) by following the instructions in the How To Enhance the Hello World Application in TIBCO Business Studio cheat sheet, or extract `jv.helloworld2.soa/Deployment Artifacts/jv.helloworld2.soa.daa` from `helloworld2.zip`.
2. Extract `datemgr_build.xml`, `datemgr_data.xml`, and `jv.datemanager.soa.daa` from `jv.datemanager.soa/Deployment Artifacts/` from `helloworld2.zip`.
3. Edit `datemgr_build.xml` and replace all instances of `propsFile` with `CONFIG_HOME/admin/amxadmin/samples/remote_props.properties`. Set the file attribute of the import element to `TIBCO_HOME/administrator/<version>/bin/taskdef.xml`.

## Deploying and Starting the Date Manager Application from the CLI

### Procedure

- In a TIBCO ActiveMatrix terminal window, run `ant -f datemgr_build.xml`.

```
Buildfile: datemgr_build.xml
upload.daa:
[AMXAdminTask] INFO - Processing 1 objects
[AMXAdminTask] INFO - Uploading DAA...
[AMXAdminTask] INFO - DAA location: jv.datemanager.soa.daa
[AMXAdminTask] INFO - Successfully added Application Template
'jv.datemanager.soa' (version: '1.0.0.201005041043') to the Administrator
Staging Area
[AMXAdminTask] INFO - Successfully added Feature
'jv.datemanager.soa.customfeature.id' (version: '1.0.0.201005041043') to the
Administrator Staging Area
create.app:
[AMXAdminTask] INFO - Processing 1 objects
[AMXAdminTask] INFO - Adding application...
[AMXAdminTask] INFO - Successfully added Application 'datemanager'
map.app.to.node:
[AMXAdminTask] INFO - Processing 1 objects
[AMXAdminTask] INFO - Mapping application to nodes...
[AMXAdminTask] INFO - Successfully mapped application 'datemanager' to node
'DevNode'
create.rt:
[AMXAdminTask] INFO - Processing 1 objects
[AMXAdminTask] INFO - HttpConnector Resource template created with name
DateMgrConnectorTemplate and Id 29
create.ri:
[AMXAdminTask] INFO - Processing 1 objects
[AMXAdminTask] INFO - Adding Resource Instances...
[AMXAdminTask] INFO - Resource Instance datemgrConnector created on Node DevNode
install.ri:
[AMXAdminTask] INFO - Processing 1 objects
[AMXAdminTask] INFO - Start to install Resource Instance 'datemgrConnector'
[AMXAdminTask] INFO - Resource Instance Install finished successfully
deploy.app:
[AMXAdminTask] INFO - Processing 1 objects
```

```
[AMXAdminTask] INFO - Deploying application...
[AMXAdminTask] INFO - Successfully deployed Application 'datemanager':
Deployment succeeded for application 'datemanager' at 05/03/10 4:04 PM
[AMXAdminTask] INFO - .
[AMXAdminTask] INFO - Application Deploy with Start finished successfully

all:

BUILD SUCCESSFUL
```

## Creating the Hello World Application

### Procedure

1. Click the **Applications** tab.  
The Applications list displays. The datemanager application appears in the list with Runtime State Running.
2. Click the **New, New Application**.  
The New Application wizard displays.
3. Select the radio button for **Create the application from a DAA or EAR file** and click the **Browse** button.
4. Navigate to the folder containing the file `fv.helloworld2.soa.daa`, click the DAA, and click **Open**.
5. In the Application Name field, replace the default name with `helloworld2`.
6. In the Environment Name drop-down list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Choose to import the listed features and click **Next**.
10. Check the checkbox for the node where you want to deploy your application and click **Next**.
11. In the promoted references screen, click **Next**.
12. Choose to import to listed resource templates and click **Next**.
13. Accept default values for the configuration properties and click **Next**.
14. Accept default values for the substitution variables and click **Next**.
15. Click **Save And Exit**.  
The helloworld2 application is added to the Applications list with Runtime State Not Deployed. The specified resource templates are imported into the Administrator database.

## Creating an HTTP Connector Resource Template

### Procedure

- Follow the steps in [Creating an HTTP Connector Resource Template](#). In the Name field, type `hello2Connector`. In the Port field, type `9096`.

## Creating and Installing the HTTP Connector Resource Instance

### Procedure

- Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#). Use the template `hello2Connector`.



## Creating and Installing the HTTP Client Resource Instance

### Procedure

- Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#). Use the `HttpClient_DateManagerSOAP` template.

## Deploying and Starting the Enhanced Hello World Application

### Procedure

- Click **Deploy**.  
The application is deployed and started. If you have completed [Create a UDDI Server in Administrator](#), the WSDL file is published in the UDDI server.
- Click  to refresh the display.  
The Runtime State changes to Starting and the Action History changes to In Progress (Deploy).
- Click  to refresh the display.  
The Runtime State changes to Running and the Action History changes to Deploy with Start Successful.

## Distributing the Enhanced Hello World Application

### Procedure

- Follow the steps in [Distributing the Hello World Application](#). Click the **helloworld2** application and the **java.helloworld2.soa** component.

## Generating the Hello World WSDL File

### Procedure

- Follow the steps in [Generating the Hello World WSDL File](#). Click the **HelloWorld2SOAP** binding and name the WSDL file `helloworld2.wsdl`.

## Invoking the Hello World Service

### Procedure

- Apply the steps in [Invoking the Hello World Service](#) to `helloworld2.wsdl`.  
The Status pane displays a response of the form:  
`HelloResponse (string): Hi Jim! This is the Java component.`  
`The current time is 2009-11-19 16:32:57.337.`

## How to Deploy and Run the Java8 Hello World Application

---

Refer to the sample available in `TIBCO_HOME\amx\<version>\samples\java\helloworld5.zip`. This sample introduces you to the Java 8 features. Steps to deploy and run the Java8 Hello World Application are same as mentioned in [How to Deploy and Run the Hello World Application](#)



# How to Deploy and Run REST Binding Samples

---

The TIBCO ActiveMatrix installation includes REST Binding sample programs which demonstrates use of the HTTP operations and Mediation component. These samples are located in the `TIBCO_HOME\amx\<version>\samples\rest` directory. For step-by-step instructions on these samples, refer to the "Sample Projects" section in the *REST Binding Development Guide*.

## Service Side

The REST Binding includes the following sample programs for the service side:

- **Bookstore sample**  
Implemented in Java. Exposes a potential interaction of a bookstore administrator with the bookstore inventory. The sample includes two HTTP GET operations, `getBookList` and `getBookByTitle`, and one HTTP POST operation, `addBook`.
- **Multiplecomplextypes sample**  
Demonstrates the use of a Mediation component to expose a WSDL operation with multiple parts of complex type. Your application might need to perform such mediation because the REST Binding only supports WSDL operations with a single part of complex type.

## Reference Side

The REST Binding includes the following sample programs for the reference side:

- **Bookstore client sample**  
This sample is configured with a REST Binding on a reference with XML media-type. It consumes the bookstore sample service shipped with TIBCO ActiveMatrix.
- **Facebook client sample**  
This sample is configured with a REST Binding on a reference with Standard JSON as the media-type. It invokes an external FaceBook service.
- **Pass-Through Mode Sample**  
This sample is configured with REST Binding on a reference to demonstrate the Pass Through Mode.

## Common to Service and Reference Side

- **REST Context Sample**  
This sample demonstrates the usage of context parameters and headers in a REST Binding.
- **REST Extended JSON Conversion Sample**  
This sample demonstrates the XML to JSON conversion of the "string" XSD element on the Reference side (Inbound/Request) and Service side (Outbound/Response).

# How to Deploy and Run the Phonebook Application

This how-to describes how to deploy and run the Phonebook web application in TIBCO ActiveMatrix Administrator.

## Completing Phonebook Prerequisites


### Procedure

1. In TIBCO Business Studio, create a distributed application archive (DAA) by following the instructions in the How to Create and Package the Phonebook Application using the TIBCO Business Studio cheat sheet, or extract `jv.phonebook.soa/Deployment Artifacts/jv.phonebook.soa.daa` from `phonebook.zip`.
2. Download HyperSQL Database Engine version 1.8.1.2 from [http://sourceforge.net/projects/hsqldb/files/hsqldb/hsqldb\\_1\\_8\\_1/hsqldb\\_1\\_8\\_1\\_2.zip/download](http://sourceforge.net/projects/hsqldb/files/hsqldb/hsqldb_1_8_1/hsqldb_1_8_1_2.zip/download).
3. Unpack `hsqldb-1.8.1.2.zip`.
4. Change to the server directory: `cd hsqldb/bin`.
5. Start the HyperSQL database server: `java -cp ../lib/hsqldb.jar org.hsqldb.Server`.  
The server outputs:

```
> java -cp ../lib/hsqldb.jar org.hsqldb.Server
[Server@83cc67]: [Thread[main,5,main]]: checkRunning(false) entered
[Server@83cc67]: [Thread[main,5,main]]: checkRunning(false) exited
[Server@83cc67]: Startup sequence initiated from main() method
[Server@83cc67]: Loaded properties from [C:\Documents and Settings\sbodoff\
\Desktop\hsqldb\hsqldb\bin\server.properties]
[Server@83cc67]: Initiating startup sequence...
[Server@83cc67]: Server socket opened successfully in 78 ms.
[Server@83cc67]: Database [index=0, id=0, db=file:test, alias=] opened
sucessfully in 468 ms.
[Server@83cc67]: Startup sequence completed in 546 ms.
[Server@83cc67]: 2010-05-04 14:06:10.923 HSQldb server 1.8.1 is online
[Server@83cc67]: To close normally, connect and execute SHUTDOWN SQL
[Server@83cc67]: From command line, use [Ctrl]+[C] to abort abruptly
```

## Creating the Phonebook Application

### Procedure

1. Click **Applications**.  
The Applications list displays.
2. Click the **New**  **New Application**.  
The Application Setup wizard displays.
3. Select the radio button for **Create the application from a DAA or EAR file** and click the **Browse** button.
4. Navigate to the folder containing the file `jv.phonebook.soa.daa`, click the DAA, and click **Open**.
5. In the Application Name field, type `phonebook`.
6. In the Environment Name drop-down list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Choose to import the listed features and click **Next**.

10. Check the checkbox for the node where you want to deploy your application and click **Next**.
11. Choose to import to listed resource templates and click **Next**.
12. Accept default values for the configuration properties and click **Next**.
13. Click **Save And Exit**.  
The phonebook application is added to the Applications list with Runtime State Not Deployed. The specified resource templates are imported into the Administrator database.

## Creating an HTTP Connector Resource Template

### Procedure

- Follow the steps in [Creating an HTTP Connector Resource Template](#). In the Name field, type `phonebookConnector`. In the Port field, type 9098.

## Creating and Installing the HTTP Connector Resource Instance

### Procedure

- Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#). Use the template `phonebookConnector`.

## Creating and Installing the JDBC Shared Resource Instance

### Procedure

- Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#). Use the template `hsqldb` and specify the instance name `hsqldb`.



## Distributing the Phonebook Application

### Procedure

- Follow the steps in [Distributing the Hello World Application](#). Click the **phonebook** application and the **jv.phonebook.soa** component.

## Deploying and Starting the Phonebook Application

### Procedure

1. Click **Deploy**.  
The application is deployed and started. If you have completed [Create a UDDI Server in Administrator](#), the WSDL file is published in the UDDI server.
2. Click  to refresh the display.  
The Runtime State changes to Starting and the Action History changes to In Progress (Deploy).
3. Click  to refresh the display.  
The Runtime State changes to Running and the Action History changes to Deploy with Start Successful.

## Generating the Phonebook WSDL File

### Procedure

- Follow the steps in [Generating the Hello World WSDL File](#). Click the **PhonebookPT** service, click the **PhonebookSOAP** binding, and name the WSDL file `phonebook.wsdl`.

## Invoking the Phonebook Service

### Procedure

- Apply the steps in [Invoking the Hello World Service](#) to `phonebook.wsdl`.
- Click the **addPhone** node. In the Actions pane, enter values in the fields `firstName: Joe`, `lastName: Doe`, and `phone: 111-1111`.  
The Status pane displays:
- Click the **getPhone** node.
- In the Invoke a WSDL Operation pane on the right, click the **Add** link next to the `firstName` field.  
A new row is added below the field.
- Type `Joe` in the empty row.
- Click **Go**.

The Status pane displays:

```
GetPhoneResponse
out
entryId (string):  0

firstName (string):  Joe
lastName (string):  Doe
phone (string):  111-1111
```

# How to Deploy and Run the Hello World Web Application

---

This how-to describes how to deploy and run the Hello World web application in TIBCO ActiveMatrix Administrator.

## Completing Hello World Web Application Prerequisites

### Procedure

- In TIBCO Business Studio, create a distributed application archive (DAA) by following the instructions in the [How To Create and Package a Hello World Web Application](#) in the TIBCO Business Studio cheat sheet, or extract `webapp.helloworld.soa/Deployment Artifacts/webapp.helloworld.soa.daa` from `helloworld.zip`.

## Creating the Hello World Web Application

### Procedure

1. Click **Applications**.  
The Applications list displays.
2. Click the **New** **New Application**.  
The New Application wizard displays.
3. Select the radio button for **Create the application from a DAA or EAR file** and click the **Browse** button.
4. Navigate to the folder containing the file `webapp.helloworld.soa.daa`, click the DAA, and click **Open**.
5. In the Application Name field, replace the default name with `helloworldwebapp`.
6. In the Environment Name drop-down list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Check the checkbox for the node where you want to deploy your application and click **Next**.
10. Click **Save and Exit**.  
`helloworldwebapp` is added to the Applications list with Runtime State Not Deployed.

## Creating an HTTP Connector Resource Template

### Procedure

- Follow the steps in [Creating an HTTP Connector Resource Template](#). In the Name field, type `WebAppHttpConnectorTemplate`. In the Port field, type `9099`.

## Creating and Installing the HTTP Connector Resource Instance

### Procedure

- Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#). Use the template `WebAppHttpConnectorTemplate` and specify the instance name `webAppHttpConnector`.

## Distributing the Hello World Web Application

### Procedure

- Follow the steps in [Distributing the Hello World Application](#). Click the **helloworldwebapp** application and the **webapp.helloworld.soa** component.



## Invoking the Hello World Web Application

### Procedure

- In a browser, open the location: `http://localhost:9099/helloworld/sayHello?firstName=Jim`. The response is: Hi Jim! This is the Web App component.

## Deploying and Starting the Hello World Web Application

### Procedure

1. Click **Deploy**.  
The application is deployed and started. If you have completed [Create a UDDI Server in Administrator](#), the WSDL file is published in the UDDI server.
2. Click  to refresh the display.  
The Runtime State changes to Starting and the Action History changes to In Progress (Deploy).
3. Click  to refresh the display.  
The Runtime State changes to Running and the Action History changes to Deploy with Start Successful.

# How to Deploy and Run the Content-Based Routing Mediation Application

---

This how-to describes how to deploy and run the content-based routing mediation application in TIBCO ActiveMatrix Administrator.


## Creating the Routing and Target Service Mediation Applications

### Procedure

1. Click **Applications**.
2. Click the **New** **New Application**.  
The New Application wizard displays.
3. Select the radio button for **Create the application from a DAA or EAR file** and click the **Browse** button.
4. Navigate to the folder `TIBCO_HOME\amx_it_mediation\<version>\samples\ContentBasedRouting\mediation.querygds.route.soa\Deployment Artifacts\` and select click **mediation.querygds.route.daa**, and click **Open**.
5. In the Application Name field, type `querygds`.
6. In the Environment Name drop-down list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Check the checkbox for the node where you want to deploy your application and click **Next**.
10. Click **Next**.
11. Choose to import to listed resource templates and click **Next**.
12. Accept default values for the configuration properties and click **Next**.
13. Click **Save And Exit**.  
The `querygds` application is added to the Applications list with Runtime State Not Deployed and the resource templates used by the application are added to the database.
14. Repeat steps 2 through 13 for the target service application. Name the application `targetservice` and use the DAA `TIBCO_HOME\amx_it_mediation\<version>\samples\TargetService\targetservice.soa\Deployment Artifacts\targetservice.mediationsamples.daa`.  
The `targetservice` application is added to the Applications list with Runtime State Not Deployed.

## Creating HTTP Connector Resource Templates




### Procedure

1. Select **Shared Objects > Resource Templates**.
2. Click  **New**.
3. In the Name field, type `httpConnector`.
4. In the Type drop-down list, select **HTTP Connector**.
5. Accept the default Machine Name.
6. In the Port field, type `8777`.

7. Click **Save**.
8. Repeat the preceding steps for a template named `httpConnectorTargetService` with port 8666.


## Creating and Installing Resource Instances

### Procedure

1. Select **Infrastructure > Hosts** .  
The Hosts list displays.
2. Click the **Resource Instances** tab.  
The All Instances list displays.
3. Click **New**.  
The New Resource Instances dialog displays.
4. In the Resource Templates list, click the **httpConnector** row.  
The Instance Name field is set to `httpConnector`.
5. In the Available Nodes list at the bottom, click **DevNode** and click  .  
The node moves to the Selected Nodes list.
6. Click **Save and Close**.  
The `httpConnector` resource instance is added to the All Instances table with status `NOT_INSTALLED`.
7. Click the row containing **httpConnector** and click  **Install** .  
The Action History changes to In Progress (Install).
8. Click  until the status changes to Running.
9. Repeat the preceding steps to create and install resource instances from the following resource templates:
  - `httpConnectorTargetService`
  - `HttpClient_querygds.soap.binding`
  - `HttpClient_querygdsasia.soap.binding`
  - `HttpClient_querygdseurope.soap.binding`
  - `HttpClient_querygdsus.soap.binding`

## Distributing the Routing and Target Service Applications



### Procedure

1. Click **Applications**.
2. In the Applications list, click **querygds**.
3. Click the **Distribution** tab.
4. In the Available Nodes list, click **DevNode** and click  .  
The node moves to the Selected Nodes list.
5. Click **Save**.
6. Repeat the preceding steps for the `targetservice` application.



## Deploying and Starting the Routing and Target Service Applications

### Procedure

1. In the Applications list, select **querygds** and **targetservice**.
2. Click **Deploy**.  
The applications are deployed and started.
3. Click  to refresh the display.  
The Runtime State changes to Starting and the Action History changes to In Progress (Deploy).
4. Click  to refresh the display.  
The Runtime State changes to Running and the Action History changes to Deploy with Start Successful.

## Generating the Routing WSDL File

### Procedure

1. Click the **Configuration** tab.
2. Expand the **mediation.querygds.route** node.
3. Click the **QueryGDS** service.  
The service details display on the right.
4. Click the **Bindings** link.  
The service's binding details display.
5. In the list of bindings, expand **SOAPService\_Binding1**.  
The list of nodes on which the binding is deployed displays below.
6. In the node list, click **DevNode**.  
The Generate WSDL button activates.
7. Click **Generate WSDL**.  
The WSDL document representing the deployed service displays.
8. Save the WSDL file with the name `QueryGDS_gen.wsdl` in the ConcreteWSDLs project in TIBCO Business Studio.