



# **TIBCO ActiveMatrix® Service Grid**

## **Administration Tutorials**

Version 3.4.3 | February 2025

# Contents

---

<b>Contents</b>	<b>2</b>
<b>Overview</b>	<b>5</b>
<b>Tutorial Prerequisites</b>	<b>6</b>
Setting Up the Runtime Environment	6
Creating an UDDI Server in the Administrator	6
Configuring Administrator Command-Line Properties	7
Setting Up the Service Invocation Environment	8
<b>How to Configure Logging</b>	<b>9</b>
Creating a Logging Appender	9
Navigating to a Logging Configurations List	11
Creating a Logging Configuration for a Host or a Node	11
Applying a Logging Configuration	13
<b>How to Deploy and Run the Hello World Application</b>	<b>15</b>
Completing Hello World Prerequisites	15
Creating the Hello World Application	15
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	19
Viewing the Hello World Service in the UDDI Registry Server	20
<b>How to Deploy and Run the Enhanced Hello World Application</b>	<b>22</b>
Completing Enhanced Hello World Prerequisites	22

Deploying and Starting the Date Manager Application from the CLI .....	22
Creating the Hello World Application .....	24
Creating an HTTP Connector Resource Template .....	25
Creating and Installing the HTTP Connector Resource Instance .....	25
Creating and Installing the HTTP Client Resource Instance .....	25
Deploying and Starting the Enhanced Hello World Application .....	26
Distributing the Enhanced Hello World Application .....	26
Generating the Enhanced Hello World WSDL File .....	26
Invoking the Enhanced Hello World Service .....	27
<b>How to Deploy and Run the Java 8 Hello World Application .....</b>	<b>28</b>
<b>How to Deploy and Run REST Binding Samples .....</b>	<b>29</b>
<b>How to Deploy and Run the Phonebook Application .....</b>	<b>31</b>
Completing Phonebook Prerequisites .....	31
Creating the Phonebook Application .....	32
Creating an HTTP Connector Resource Template for the Phonebook Application .....	33
Creating and Installing the HTTP Connector Resource Instance .....	33
Creating and Installing the JDBC Shared Resource Instance .....	33
Distributing the Phonebook Application .....	34
Deploying and Starting the Phonebook Application .....	34
Generating the Phonebook WSDL File .....	34
Invoking the Phonebook Service .....	35
<b>How to Deploy and Run the Hello World Web Application .....</b>	<b>36</b>
Completing Hello World Web Application Prerequisites .....	36
Creating the Hello World Web Application .....	36
Creating an HTTP Connector Resource Template .....	37
Creating and Installing the HTTP Connector Resource Instance .....	37
Distributing the Hello World Web Application .....	38
Invoking the Hello World Web Application .....	38

Deploying and Starting the Hello World Web Application .....	38
<b>How to Deploy and Run the Content-Based Routing Mediation Application</b>	<b>39</b>
Creating the Routing and Target Service Mediation Applications .....	39
Creating HTTP Connector Resource Templates .....	40
Creating and Installing Resource Instances .....	41
Distributing the Routing and Target Service Applications .....	42
Deploying and Starting the Routing and Target Service Applications .....	42
Generating the Routing WSDL File .....	43
<b>TIBCO Documentation and Support Services .....</b>	<b>44</b>
<b>Legal and Third-Party Notices .....</b>	<b>46</b>

# Overview

These tutorials demonstrate how to deploy and run SOA applications developed in TIBCO Business Studio™ - BPM Edition in TIBCO ActiveMatrix Administrator and send SOAP requests from the TIBCO Business Studio - BPM Edition 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.


## Setting 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, the 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.

## Creating an UDDI Server in the 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, select **UDDI**.

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 respectively.
9. Click **Test Connection**.
10. In the **Publication Business** dropdown list, type Services deployed in TIBCO ActiveMatrix.
11. Select the **Automatic Publication** checkbox.
12. Click **Save**.

## Result

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

# Configuring 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 that you specified when you created the Administrator server.
2. Save the properties file.

# Setting Up the Service Invocation Environment

Create a project in TIBCO Business Studio - BPM Edition 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 - BPM Edition Web Services Explorer and generate SOAP requests.

## Procedure

1. Start TIBCO Business Studio - BPM Edition.
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

---

Instructions on configuring logging are 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.

### GUI

#### Procedure

1. Select **Shared Objects > Logging Appenders**.

2. Click **New**.

The New Logging Appender dialog displays with the dropdown list of the logging appender type expanded.

3. Select an appender type from the Type list.

- Clear Text File - Appends events to a file in the 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 Appenders table and is selected.

#### Result

**New Logging Appender**

**Name**  
TestLog

**Type**  
Clear Text File Appender

**Description (optional)**

**File Path**  
C:\reports\log.txt

**Pattern Layout**  
%d{dd MMM yyyy HH:mm:ss,SSS} [%t] [%-5p] [%R[\_cl.logicalCompld.matrix.application]] %c - %m%n

**Max File Size (MB)**  
10

**Max Backup Index**  
5

**Save** **Revert**

## CLI

### Procedure

1. In the `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/cfgmgmt/tibcohost/Admin-amxadmin-
instanceOne/
```

```
nodes/DevNode/logs/HelloWorld.log"
maxSize="10000" maxBackupNum="5"/>
```

2. In the `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
```

For information on how to retrieve log entries from the destination queue of a JMS appender, see *TIBCO ActiveMatrix® Service Grid Composite Development*.

## 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. 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
Basic	<ol style="list-style-type: none"><li>a. Click <b>Add</b>. A row is added to the list.</li><li>b. In the Logger Name column, type a logging configuration name.</li><li>c. Select the FileAppender log level.</li><li>d. Click <b>Save And Apply</b>, or <b>Save</b>, or <b>Revert</b>.</li></ol>
Advanced	<ol style="list-style-type: none"><li>a. Click <b>Add</b>. A row is added to the list.</li><li>b. In the Logger Name column, type a logging configuration name or select from the list.</li><li>c. In the Additivity column, select an additivity.</li><li>d. Click <b>Set Appender</b>. A row is added to the list.</li><li>e. In the Appender column, select an appender from the list.</li><li>f. In the Level column, select a logging level.</li><li>g. 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>h. 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 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 naming 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 or 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 the 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

In TIBCO Business Studio - BPM Edition, create a distributed application archive (DAA) by following the instructions in the [How To Create and Package a Hello World Application](#) TIBCO Business Studio - BPM Edition cheat sheet or from `helloworld1.zip` extract -

`jv.helloworld1.soa/Deployment Artifacts/jv.helloworld1.soa.daa`.

## Creating the Hello World Application

### Procedure

1. Click the **Applications** tab.
2. Click **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 **Browse**.
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 dropdown list, select **DevEnvironment**.
7. Choose an application folder.

8. Click **Next**.
9. Choose to import the listed feature and click **Next**.
10. Select the checkbox for the node where you want to deploy your application and click **Next**.
11. Accept the 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** dropdown list, select **HTTP Connector**.



The HTTP connector configuration fields are displayed.

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 are displayed 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 [Creating an UDDI Server in the Administrator](#), the WSDL file is published in the UDDIserver.

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 **ju.helloworld1.soa** node.

3. Click the **HelloWorldPT** service.

The service details are displayed 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 is displayed 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 - BPM Edition.

## 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

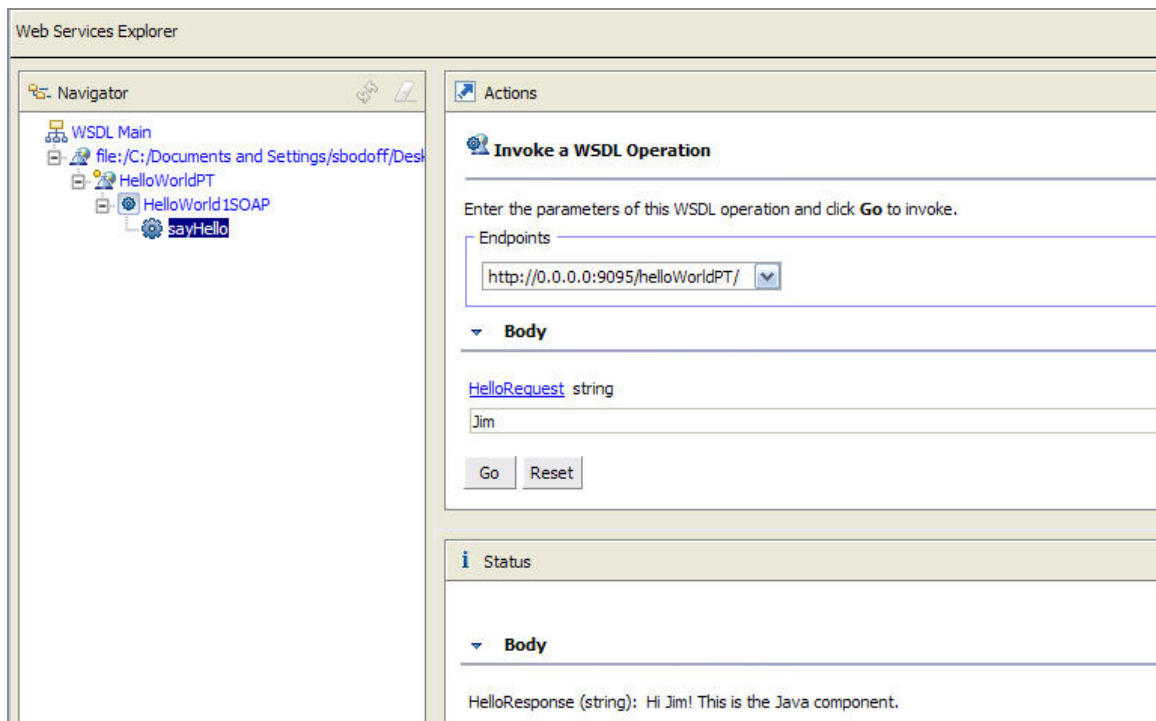
1. In TIBCO Business Studio - BPM Edition, right-click Concrete WSDLs/helloworld1.wsdl and select **Web Services > Test with Web Services Explorer**.

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 pane displays on the right with the newly created endpoint selected in the **Endpoints** dropdown list.

4. In the Actions pane, type Jim in the **HelloRequest** field.
5. Click **Go**.



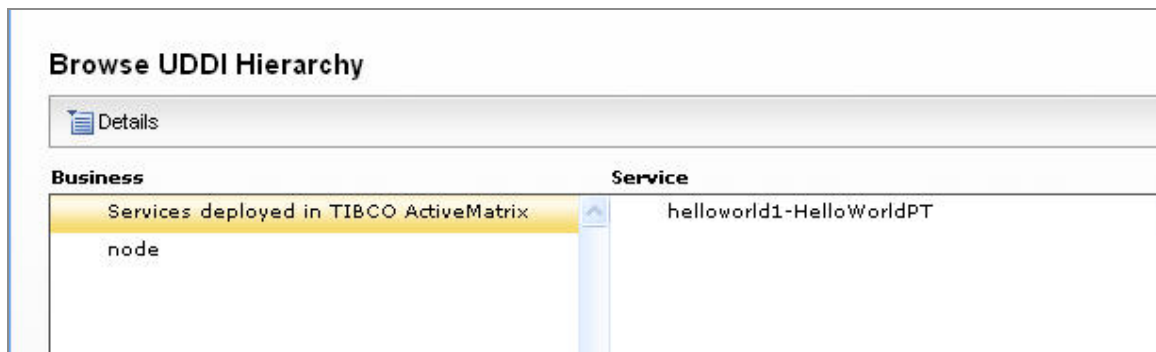
## Viewing the Hello World Service in the UDDI Registry Server

### Procedure

1. Start the TIBCO ActiveMatrix UDDI Service Console GUI.

The services deployed in ActiveMatrix Service Grid are displayed in the Service

Console.

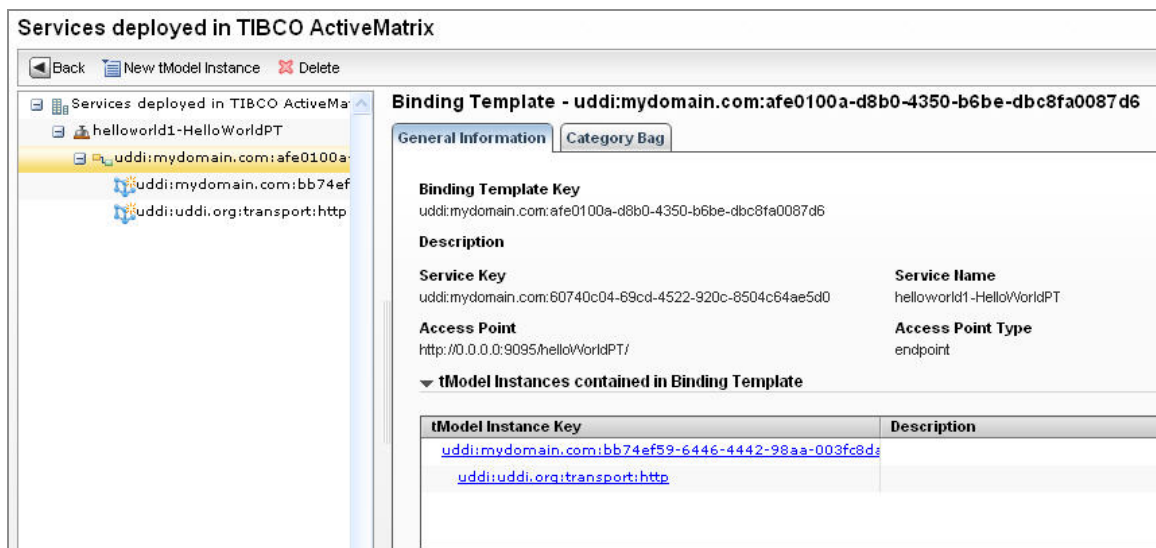


2. Click **Details**.

The details of the services deployed in ActiveMatrix Service Grid are displayed.

3. In the 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/`.



# 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 ActiveMatrix Service Grid Administrator.

## Completing Enhanced Hello World Prerequisites

### Procedure

1. In TIBCO Business Studio - BPM Edition, create a distributed application archive (DAA) by following the instructions in the How To Enhance the Hello World Application in TIBCO Business Studio - BPM Edition 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

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 datamanager application appears in the list with Runtime State Running.

2. Click **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 **Browse**.

4. Navigate to the folder containing the file `jav.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 dropdown list, select **DevEnvironment**.

7. Choose an application folder.

8. Click **Next**.

9. Choose to import the listed features and click **Next**.

10. Select 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 the default values for the configuration properties and click **Next**.

14. Accept the 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

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

## Creating and Installing the HTTP Connector Resource Instance

### Procedure

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



## Creating and Installing the HTTP Client Resource Instance

### Procedure

1. Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#).
2. Use the HttpClient\_DateManagerSOAP template.

# Deploying and Starting the Enhanced Hello World Application

## Procedure

1. Click **Deploy**.  
The application is deployed and started. If you have completed [Creating an UDDI Server in the 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.

# Distributing the Enhanced Hello World Application

## Procedure

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

# Generating the Enhanced Hello World WSDL File

## Procedure

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

## Invoking the Enhanced Hello World Service

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 Java 8 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. The steps to deploy and run the Java 8 Hello World Application are the same as mentioned in [How to Deploy and Run the Hello World Application](#).

# How to Deploy and Run REST Binding Samples

---

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

## Service Side

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

- **Bookstore sample**  
Implemented in Java. It 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 the 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 a 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 included in TIBCO ActiveMatrix® Service Grid.
- **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 - BPM Edition, 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 - BPM Edition cheat sheet, or from phonebook.zip extract `jv.phonebook.soa/Deployment Artifacts/jv.phonebook.soa.daa`.
2. Download HyperSQL Database Engine version 1.8.1.2 from <http://sourceforge.net/>.
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 successfully 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 is displayed.
2. Click **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 **Browse**.
4. Navigate to the folder containing the file `jav.phonebook.soa.daa`, click the DAA, and click **Open**.
5. In the **Application Name** field, type `phonebook`.
6. In the **Environment Name** dropdown list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Choose to import the listed features and click **Next**.
10. Select 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 the 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 for the Phonebook Application

### Procedure

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

## Creating and Installing the HTTP Connector Resource Instance

### Procedure

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

## Creating and Installing the JDBC Shared Resource Instance

### Procedure

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



## Distributing the Phonebook Application

### Procedure

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

## Deploying and Starting the Phonebook Application

### Procedure

1. Click **Deploy**.  
The application is deployed and started. If you have completed [Creating an UDDI Server in the 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

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

# Invoking the Phonebook Service

## Procedure

1. Apply the steps in [Invoking the Hello World Service](#) to `phonebook.wsdl`.
2. Click the **addPhone** node. In the Actions pane, enter values in the following fields:

**firstName:** Joe

**lastName:** Doe

**phone:** 111-1111.

The Status pane displays:

```
AddPhoneResponse
entryId (string):  0
```

3. Click the **getPhone** node.
4. 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.

5. Type Joe in the empty row.
6. 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

In TIBCO Business Studio - BPM Edition, 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 - BPM Edition cheat sheet, or from `helloworld.zip` extract `webapp.helloworld.soa/Deployment Artifacts/webapp.helloworld.soa.daa`.

## Creating the Hello World Web Application

### Procedure

1. Click **Applications**.  
The Applications list is displayed.
2. Click **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 **Browse**.
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 dropdown list, select **DevEnvironment**.
7. Choose an application folder.
8. Click **Next**.
9. Select 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

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

## Creating and Installing the HTTP Connector Resource Instance

### Procedure

1. Follow the steps in [Creating and Installing the HTTP Connector Resource Instance](#).
2. Use the template WebAppHttpConnectorTemplate.
3. Specify the instance name webAppHttpConnector.

# Distributing the Hello World Web Application

## Procedure

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

# Invoking the Hello World Web Application



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 [Creating an UDDI Server in the 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 **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 **Browse**.
4. Navigate to the following folder:  
`TIBCO_HOME\amx_it_  
mediation\  
<version>  
\samples\ContentBasedRouting\mediation.querygds.route.soa\Deployment  
Artifacts\`
5. Click **mediation.querygds.route.daa** and click **Open**.
6. In the **Application Name** field, type querygds.
7. In the **Environment Name** dropdown list, select **DevEnvironment**.
8. Choose an application folder.
9. Click **Next**.

10. Select the checkbox for the node where you want to deploy your application and click **Next**.
11. Click **Next**.
12. Choose to import to listed resource templates and click **Next**.
13. Accept the default values for the configuration properties and click **Next**.
14. Click **Save And Exit**.


The querygds application is added to **Applications list with Runtime State Not Deployed** and the resource templates used by the application are added to the database.

15. Repeat steps 2- 13 for the target service application. Name the application targetservice.
16. Use the DAA `TIBCO_HOME\amx_it_mediation\<version>\samples\TargetService\targetservice.soa\Deployment Artifacts\targetservice.mediationsamples.daa`.

The targetservice application is added to **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** dropdown 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 the 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 are displayed 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 is displayed 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 - BPM Edition.

# 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 documentation for this product is available on the [TIBCO ActiveMatrix® Service Grid Product Documentation](#) page.

## How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our [product Support website](#).
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the [product Support website](#). If you do not have a username, you can request one by clicking **Register** on the website.

## How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature

requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

# Legal and Third-Party Notices

---

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

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

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

TIBCO, the TIBCO logo, the TIBCO O logo, ActiveMatrix, Business Studio, Enterprise Message Service, and Hawk are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

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

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

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

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

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

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

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

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at <https://www.tibco.com/patents>.

Copyright © 2010-2025. Cloud Software Group, Inc. All Rights Reserved.