TIBCO ActiveMatrix Implementation Type for TIBCO Adapters

Administration

Software Release 1.0

October 2012

Two-Second Advantage®

Important Information

Contents

Preface

TIBCO ActiveMatrix Implementation Type for TIBCO Adapters (Adapter IT) allows TIBCO Adapter configurations designed in TIBCO Designer and packaged in an Enterprise Application Archive (EAR) to be uploaded and deployed using TIBCO ActiveMatrix Administrator to the TIBCO ActiveMatrix runtime environment. Once deployed, the TIBCO Adapter is managed and monitored like any other ActiveMatrix application.

TIBCO Product Documentation

This section lists documentation resources you may find useful.

The following documents form the TIBCO ActiveMatrix Implementation Type for TIBCO Adapters documentation set:

- *Administration*: Read this manual to learn how to manage the runtime and deploy applications.
- *Installation and Configuration*: Read this manual to learn how to install TIBCO ActiveMatrix Implementation Type for TIBCO Adapters software.
- *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.

Typographical Conventions

Table 1: General Typographical Conventions

Convention	Use
TIBCO_HOME *ENV_NAME*	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments. An installation environment consists of the following properties: • **Name** - Identifies the installation environment. The name is appended to the name of Windows services created by the installer and is a component of the path to the product shortcut in the **Windows Start** > **All Programs** menu. This name is referenced in documentation as *ENV_NAME*. • **Path** - The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*.
CONFIG_HOME	The folder that stores configuration data generated by TIBCO products. Configuration data can include sample scripts, session data, configured binaries, logs, and so on. This folder is referenced in documentation as *CONFIG_HOME*.

`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example:
	• Use `MyCommand` to start the foo process.
	• Code example:
	`public class HelloWorldImpl extends AbstractHelloWorldImpl {`
	`...`
	`public HelloResponseDocument sayHello(HelloRequestDocument firstName) {`
	`...`
	`System.out.println("--> Generating Java Hello Component Response...");`
	`String name = firstName.getHelloRequest()==null||firstName.getHelloRequest(). equals("")?"Friend":firstName.getHelloRequest();`
	`HelloResponseDocument resp = HelloResponseDocument.Factory.newInstance();`
	`resp.setHelloResponse("Hi " + name + "! " + "This is the Java component.\n");`
	`System.out.println("--> Java Hello Component Response: \n\t\t" +`
	`resp.getHelloResponse());`
	`...`
	`}`
	`}`
	• *CONFIG_HOME*`/admin/`*enterpriseName*`/samples/remote_props.properties`
	• Output example:
	`C:\Program Files\tibco\amx-3\studio\3.5\eclipse>amx_eclipse_ant.exe -buildfile "C:/helloworld1/build.xml" -data "C:/hws"`
	`Buildfile: C:/helloworld1/build.xml`
	`createApplicationDAA:`
	`[sds.createDAA] Waited for 47ms for workspace refreshes after building features.`
	`all:`
	`BUILD SUCCESSFUL`
	`BUILD SUCCESSFUL`
	`Total time: 2 minutes 18 seconds`

Convention	Use
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: **`Type admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: `MyCommand [enable | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO BusinessWorks Concepts*. • To define new terms. For example: A *keystore* is a database of keys and certificates. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *pathname*.
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: **Ctrl+C**. Key names separated by a comma and space indicate keys pressed one after the other. For example: **Esc, Ctrl+Q**.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2: Syntax Typographical Conventions

Convention	Use
[ ]	An optional item in command syntax. For example: `MyCommand [optional_parameter] required_parameter`
|	A logical 'OR' that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand param1 | param2 | param3`

{ }	A logical group of items in a command. Other syntax notations may appear within each logical group.
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	`MyCommand {param1 param2} | {param3 param4}`
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	`MyCommand {param1 | param2}{param3 | param4}`

Convention	Use
	In the next example, the command can accept either two or three parameters. The first parameter must be param1. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`. `MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

After you join TIBCOmmunity, you can access the documentation here: http://docs.tibco.com.

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, contact TIBCO Support as follows:

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

  http://www.tibco.com/services/support

- If you already have a valid maintenance or support contract, visit this site:

  https://support.tibco.com

  Entry to this site requires a username and password. If you do not have a username, you can request one.

Chapter

1

TIBCO ActiveMatrix Implementation Type for TIBCO Adapters

TIBCO ActiveMatrix Implementation Type for TIBCO Adapters (Adapter IT) allows TIBCO Adapter configurations designed in TIBCO Designer and packaged in an Enterprise Application Archive (EAR) to be uploaded and deployed using TIBCO ActiveMatrix Administrator to the TIBCO ActiveMatrix runtime environment. Once deployed, the TIBCO Adapter is managed and monitored like any other ActiveMatrix application.

Topics

Overview

TIBCO ActiveMatrix Implementation Type for TIBCO Adapters (Adapter IT) allows TIBCO Adapter configurations designed in TIBCO Designer and packaged in an Enterprise Application Archive (EAR) to be uploaded and deployed using TIBCO ActiveMatrix Administrator to the TIBCO ActiveMatrix runtime environment. Once deployed, the TIBCO Adapter is managed and monitored like any other ActiveMatrix application.

An Enterprise Archive Resource (EAR) is a deployment resource for TIBCO Adapters. When an EAR file is uploaded and parsed, TIBCO Adapter configurations are identified, global variables collected and an application template containing an ActiveMatrix Implementation Type for TIBCO Adapters component is created. The components created using the EAR file are *proxies* for the TIBCO Adapter configurations in the EAR file. Once deployed in the ActiveMatrix environment, the TIBCO Adapter is monitored and managed like any other ActiveMatrix application.

A TIBCO Hawk agent runs as a separate process for each configured *CONFIG_HOME* on a machine. ActiveMatrix Implementation Type for TIBCO Adapters (AdapterIT) uses the TIBCO Hawk agent to monitor and manage the TIBCO Adapter processes.

When an ActiveMatrix Implementation Type for TIBCO Adapters component is either started or stopped using the Administrator UI or through the command-line interface, the command is communicated to the Hawk agent which in turn starts or stops the component.

Any change in the state of the ActiveMatrix Implementation Type for TIBCO Adapters component is communicated to the TIBCO Adapter process by the Hawk agent, and vice-versa.

Software Components

The ActiveMatrix Implementation Type for TIBCO Adapters consists of the following:

- Adapter Implementation Type Runtime Component

  This component is auto-deployed to the ActiveMatrixnode in the ActiveMatrix environment when an EAR file containing Adapter archives is deployed to that node.

- ProxyIT Administrator Plug-in

  This plug-in provides an interface in the ActiveMatrix Administrator UI to allow monitoring and management of TIBCO Adapter applications. The plug-in also provides a web service based implementation for the functionality needed by the ActiveMatrix Administrator UI.

  Navigate to the **Admin Configuration > Plug-ins** screen and make sure that the **TIBCO ActiveMatrix Proxy IT Plugin 1.0.0** plug-in is installed and is in the deployed state.

- TIBCO Hawk Agent Workflow

  This workflow accessed via TIBCO Configuration Tool configures the TIBCO Hawk Agent, the TIBCO Hawk Microagent, and also sets up the environment for the product.

Designtime Environment

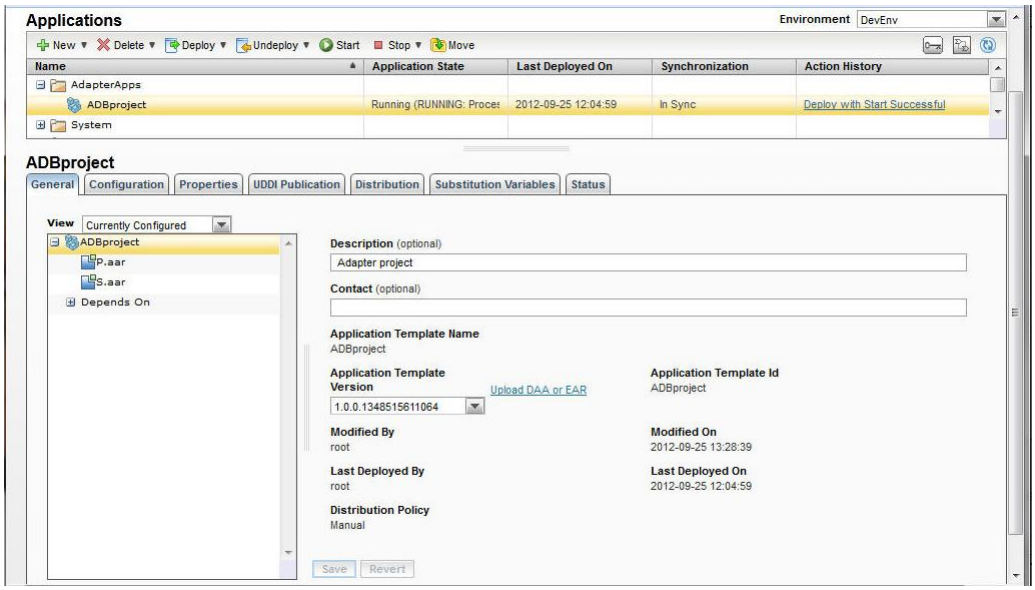
An Enterprise Archive Resource (EAR) is a deployment resource for TIBCO Adapters and TIBCO BusinessWorks. It is created using TIBCO Designer and is assembled from one or more TIBCO Adapter archives (AAR). Global and configuration specific variables set in the TIBCO Designer project are automatically added to the EAR file when the EAR is built. TIBCO Adapter SDK properties are not visible at the TIBCO Designer project level, but are properties provided by the Adapter's deployment descriptor when the TIBCO Adapter archive is built during the EAR creation process. These properties are specific to TIBCO Adapters and are responsible for controlling the runtime behavior of the TIBCO Adapter.

Hawk monitoring should be enabled in the TIBCO Adapter project at design time.

Administration

TIBCO ActiveMatrix Administrator is the utility used to create, configure, monitor, and manage objects in the TIBCO ActiveMatrix runtime. ActiveMatrix Administrator is a web application that provides a browser interface and a command-line scripting interface. The command-line interface is a set of Ant tasks that execute via an automated script.

For more information refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus.



Chapter

2

Applications

The output of the design phase of developing TIBCO Adapters applications is an EAR file. Refer to the Adapter documentation

In the administration phase, you create an application by instantiating an application template. The product installer adds product application templates to Administrator. Administrator extracts the application template when you upload an EAR file to Administrator.

Topics

Typical Workflow

Procedure

1. Create an application by uploading an EAR file.

   When an EAR file is uploaded, it is converted into the DAA format that is used by the TIBCO ActiveMatrix Administrator. A composite application is created from the EAR file and packaged into a DAA. Internally it creates a Composite application from the EAR and packages it in a DAA. The global variables of the EAR file are mapped to substitution variables for the composite application.

   Each TIBCO Adapter configuration is mapped to a Proxy Implementation Type component.

2. Configure the application.

   See Editing an Application for more information.

3. Deploy and start the application.
4. Monitor the application.

   See Invoking Hawk Methods on page 44 for more information.

Using the Command-Line Interface

Before you begin

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information about the command-line interface (CLI) and the required build and data files.

- Edit *CONFIG_HOME*`/admin/amxadmin/samples/admin-scripts-base.xml` and replace

  ```
  <path id="plugins"/>
  ```

  with

  ```
  <path id="plugins">
  <fileset dir="${plugins.dir}/com.tibco.amx.it.proxy.cli_1.0.0.001">
  <include name="*.jar" />
  </fileset>
  <pathelement path="${plugins.dir}/com.tibco.amx.extension.daa.ear_1.0.0.008.jar"/>
  <pathelement path="${plugins.dir}/com.tibco.amx.it.model.proxy_1.0.0.001.jar"/>
  <pathelement path="${plugins.dir}/com.tibco.amf.ear.artifact.model_1.0.0.002.jar"/>
  </path>
  ```

  Make sure that the version numbers for the plugins match the versions present in the plugins folder which is typically *TIBCO_HOME*`\administrator\3.2\scripts`.

- Refer to the following samples available in the folder*TIBCO_HOME*`/administrator/3.2/samples`:
  - `proxy_component_build.xml`
  - `proxy_component_data.xml`

Creating an Application

GUI

About this task

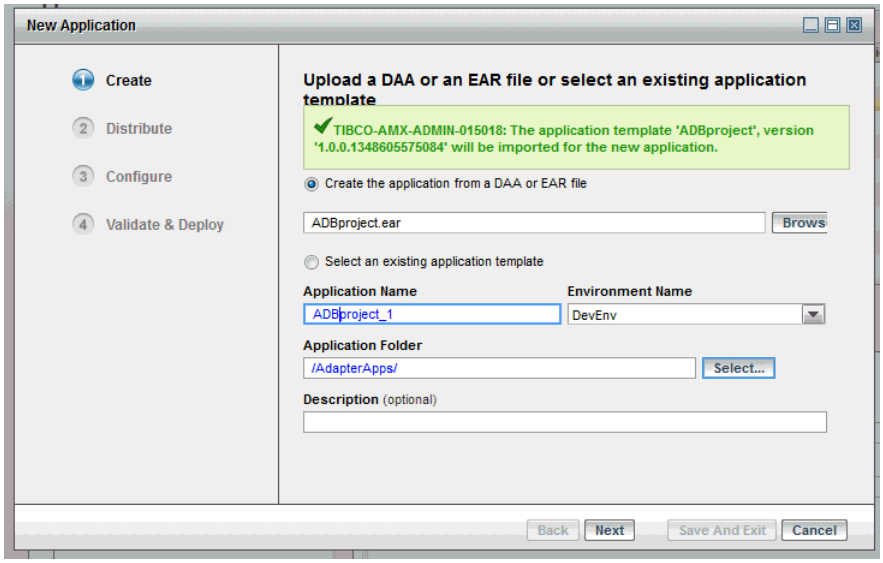
The New Application wizard allows you to create an application, specify its distribution, and configure properties and substitution variables.

Procedure

1. Click the **Applications** tab.
2. Click **New** ▾**New Application**.
   The New Application wizard displays.
3. Create an application using:
   - DAA or EAR file
     1. Click the Browse button.
     2. Navigate to a folder containing an EAR and double-click the EAR file.

     EAR files created using a newer version of the TIBCO Adapter palette should not be deployed in environments where an older version of the TIBCO Adapter is deployed.

   - Application template
     1. Select one of the displayed application template. Optionally type a string in the Search text box and click to jump to a template containing the string.



4. Specify values for the following fields:
   - Application Name - accept the default name or type an application name.

     Application names cannot contain the characters \ , / , :, *, ?, ", <, >, | , whitespace, %, #, &, (,), or comma and they cannot be the same as the node name.

   - Environment Name - from the drop-down list select an environment in which to create the application.

– Application Folder - accept the default location or click **Select...** to choose a folder for the application.

– Description - provide an optional description for the application.

Click **Next**.

5. Choose the nodes where you want to deploy the application by checking the checkbox for that node.

   The number of applications deployed on each node are displayed. This helps in distributing your applications across the available nodes.

   If you have multiple nodes running you can choose to deploy the application to more than one node by checking the checkbox for **Advanced** to fine tune the distribution.

   – **Select nodes for an item** - choose items from the composite tree on the left and drag to nodes on the right. Double-click the items to distribute it to all the nodes.

     Components or bindings in logical nodes will not appear in the item lists as they can not be distributed separately from the logical node.

     Distribution is cumulative. So if a component is explicitly distributed to node A and the application is distributed to node B, then the component will be distributed to both nodes A and B during deployment.

     Click the link to see all the nodes to which it is distributed.

   – **Select items for a node** - choose a node from the list on the left and drop on items on the right. Double-click a node to distribute all items to the selected node.

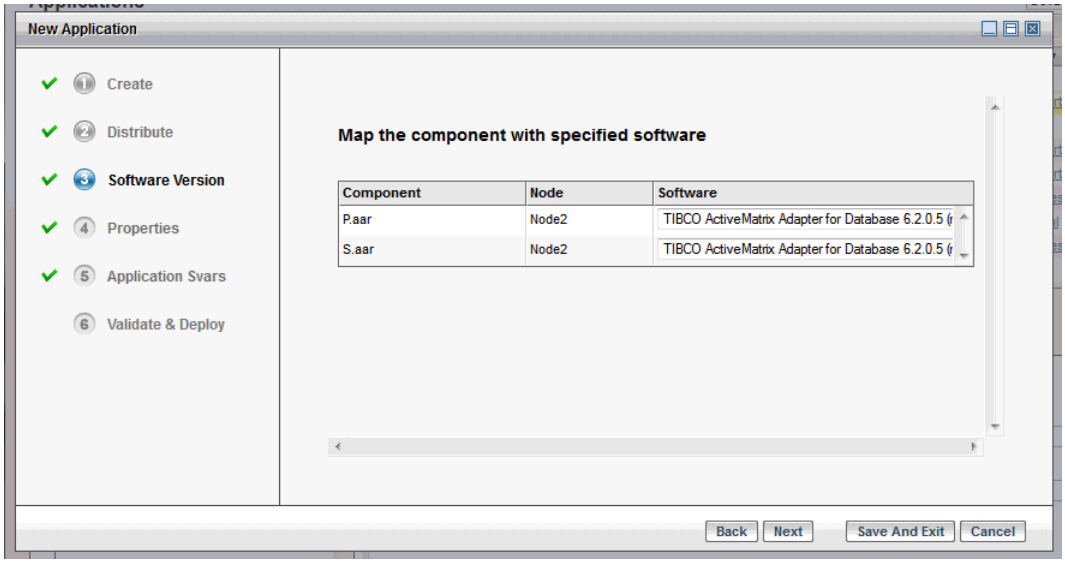
     Click the link under a node name to see all the fragments of the application distributed to the node.

   Click the **Clear distribution for the selected items** button to delete the distribution.

   Click **Next**.

6. Map the components with the specified software. By default, the highest compatible TIBCO Adapter version is selected. But you can choose a different TIBCO Adapter version if it is available. If the TIBCO Adapter archive was created in a version that is lower than the lowest software version available, then no version is available for selection.

   Choose a version from the **Software** dropdown list and click **Next**.

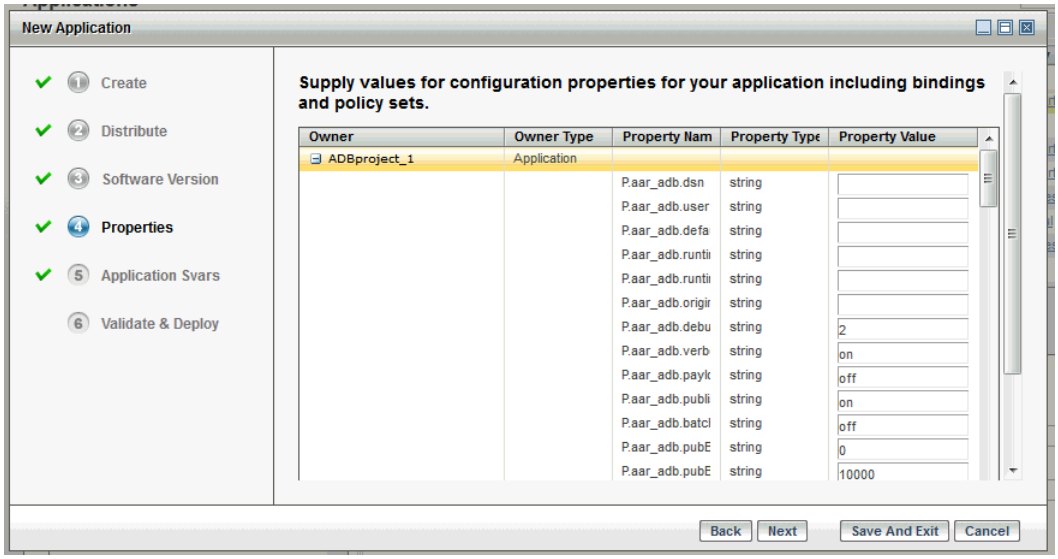


7. If your application contains configured properties, specify values for them. You can optionally specify values for the TIBCO Adapter properties defined by the those palettes. These are string values and if any are not specified you will see a warning during deployment validation. The name of the component the property applies to will be prefixed to the property name.

   Expand the tree for Owner to view and edit the displayed properties.

If Hawk monitoring is enabled for the project at design time, it is recommended that the value of the **HawkEnabled** property is set to `true`. If th value is not, the value is overridden to `true`. If a static value is used, it must be set to true.

Click **Next**.

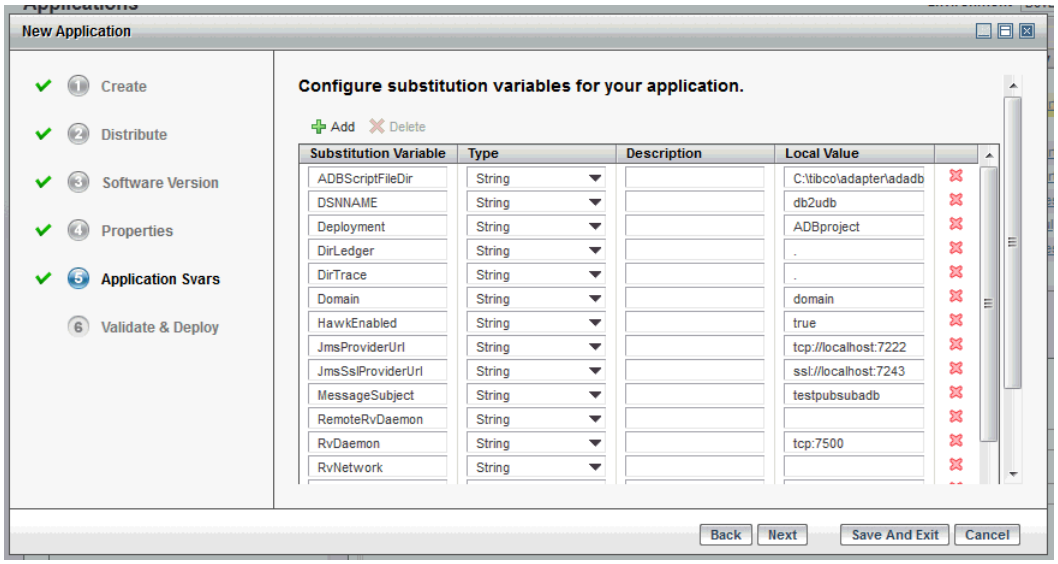


8. If your application contains substitution variables, assign values for them.

   Application level Global Variables will show up as application level substitution variables. You can override the defaults by assigning values for them.

   In the Local Value column, edit application substitution variable values. When you click on a cell in the value column, a picker icon appears that allows you to select existing resource instances of the correct type.

   Click **Next** to continue.



   It is recommended that you do not add or delete any substitution variables.

9. The validation results are displayed.
10. Choose an action based on whether you want to continue using the wizard.
    - **Back** - navigate to any previous screen in the wizard.
    - **Deploy** - deploys the newly created application.

– **Save and Exit** or **Cancel** - saves the setup information and exits the wizard.

The application is added to Applications list. Any EAR uploaded or resource templates imported in previous steps will remain.

Results

Only the TIBCO Adapter components contained in the EAR file are displayed in the Administrator UI.

CLI

Before you begin

The specified environment must already exist in the enterprise.

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information about the command-line interface (CLI) and the required build and data files.

Procedure

1. In the data file, specify an Application element in full format.

```
<Environment xsi:type="amxdata:Environment" name="EnvName" >
 <Application xsi:type="amxdata:Application" name="Adapter_EAR">
  <ApplicationTemplate xsi:type="amxdata_reference:AdapterApplicationTemplate_reference"

    name="AppTemplateName" version="1.0.0.201005040925" />
 </Application>
</Environment>
```

   To define the distribution of the components, specify the Application element as follows:

```
<Environment xsi:type="amxdata:Environment" name="EnvName" >
 <Application xsi:type="amxdata:Application" name="Adapter_EAR">
  <ApplicationTemplate xsi:type="amxdata_reference:AdapterApplicationTemplate_reference"

    name="AppTemplateName" version="1.0.0.201005040925" />
  <Component xsi:type="amxdata:Component"
name="ActiveDatabaseAdapterConfiguration-1.aar" >
      <Node name="Adapter-node-1"  environmentName="Dev" />
  <Component xsi:type="amxdata:Component"
name="ActiveDatabaseAdapterConfiguration-2.aar" >
      <Node name="Adapter-node-2"  environmentName="Dev" />
 </Application>
</Environment>
```

2. In the data file specify the location of the EAR files.

```
<DAA xsi:type="amxdata:DAA" description="Adapter EAR Files"
location="c:/Adapters/EAR_Files/PubSub.ear" />
```

3. In the build file set the `action` attribute of the `AMXAdminTask` element to **`add`** and the `objectSelector` attribute to **`Environment/Application`**.

```
<AMXAdminTask action="add" objectSelector="Environment/Application" />
```

4. Invoke the command-line interface on the build file.

Editing an Application

Procedure

1. Click **Applications**.
2. Select an application from the displayed list.
   The application details display.
3. Edit the properties and substitution variables as follows:
   Application Level
   - Edit the application properties by selecting the Properties tab.
   - Edit the application level substitution variables by selecting the Substitution Variables tab. See Substitution Variables for more information.

   Adapter Component
   - Click the General tab to choose the configuration properties. See Component General Reference on page 39 for more information.
   - Choose the version of TIBCO Adapter software to run by choosing the TIBCO Adapter component and clicking **Configuration > Software Versions**. See Software Versions for more information.
   - Substitution variables by choosing the TIBCO Adapter component and clicking **Configuration > Substitution Variables**. See #unique_27/unique_27_Connect_42_section_53AFFFBE02384813B9A15D848258AB88 for more information.
4. Click **Save**.

Selecting the Software Version

About this task
When components of the EAR file are distributed to different nodes, you have to make sure that the required TIBCO Adapter software version is running on those nodes. By default, the highest compatible TIBCO Adapter version is selected. But you can choose a different TIBCO Adapter version if it is available. If the TIBCO Adapter archive was created in a version that is lower than the lowest software version available, then no version is available for selection.

When deploying an application via the CLI if the software version is not specified, the component version from the EAR is used. If this version of software is not available, a compatible or a higher minor version is used. If the selection of compatible minor version cannot be made the deployment resulting in a **CONFIGURE_FAILED** message.

GUI

Procedure

1. Click the **Applications** button.
2. In the Applications list, click an application.
3. Select a TIBCO Adapter component.
4. Click **Configuration** > **Software Versions**.
   The list of nodes where the TIBCO Adapter component is deployed displays.
5. Choose a TIBCO Adapter version from the Software drop-down list for the node.

6. Click **Save**.

CLI

Procedure

1. In the data file along with Application element the specify the ProxyComponent element.

```
<amxdata_proxycomponent:ProxyComponent xsi:type="amxdata_proxycomponent:ProxyComponent"

 name="ActiveDatabaseAdapterConfiguration.aar">
  <softwareversion xsi:type="amxdata_proxycomponent:SoftwareVersion"
   nodename="adapter_node_1" version="6.2.0"/>
</amxdata_proxycomponent:ProxyComponent>
```

2. In the build file, set the `action` attribute of the `AMXAdminTask` element to **`setSoftwareVersion`**.

```
<AMXAdminTask action="add" objectSelector="*/Application" />
```

3. Invoke the command-line interface on the build file.

Distributing an Application

About this task

The Distribution tab is to specify the nodes on which to put the application components. The information is displayed for deployed applications, but cannot be edited. Only current configurations can be edited. To edit the distribution, select the Currently Configuration option in the View drop-down list. The allowable nodes are those in the same environment as the application.

GUI

Procedure

1. Click the **Applications** button.
2. In the Applications list, click an application.
3. Click the **Distribution** tab.
4. Select a node from the Available Nodes list and click  .
5. Click **Save**.

CLI

Procedure

1. In the data file, specify Application and Node elements in base or full format. You can distribute the entire application to a set of nodes using what is described. However, you can also distribute pieces of an application to one or more nodes. These can be components, promoted service bindings, promoted reference bindings or logical nodes. For components, they can be either composite type components or runtime components.

```
<Application xsi:type="amxdata:Application" name="testApp">
  <Node name="node1" environmentName="DevEnvironment" />
</Application>
```

2. In the build file, set the `action` attribute of the `AMXAdminTask` element to **`add`** and the `objectSelector` attribute to **`*/Application`**.

```
<AMXAdminTask action="add" objectSelector="*/Application" />
```

3. Invoke the command-line interface on the build file.

Deploying Applications

About this task

Before deploying an application containing any other type of component or binding on a node, an instance of the product application template that supports that component is automatically deployed on that node. If the Implementation Type for TIBCO Adapters application is not deployed, it will be created and deployed to the node. Make sure the **Enable Auto-Provisioning** flag has been selected for the environment. For more information refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus.

GUI

Procedure

1. Click the **Applications** button.
2. In the Applications list, click one or more applications.
3. Choose a deploy option.

Option	Procedure
Deploy Deploy with Start Dependencies on target product applications are checked.	1. Do one of the following: – Click **Deploy.** – Select **Deploy▾Deploy with Start**. 2. If the applications depend on undeployed target product applications, the Deployment Application Dependencies dialog displays. 3. Check the checkboxes next to the target applications to deploy.
Deploy without Start Dependencies on target product applications are checked.	1. Select **Deploy▾Deploy without Start** 2. If the applications depend on undeployed target product applications, the Deployment Application Dependencies dialog displays. 3. Check the checkboxes next to the target applications to deploy.
More Deploy Options	1. Select **Deploy▾More deploy options** 2. Check the checkboxes for one or more of the following options – **Start Applications** - Dependencies on target applications are checked. – **Resolve Mode** - Dependencies on target product applications are checked. Deploys the selected applications on the nodes, restarts the nodes, and causes all applications deployed on the nodes to use the latest versions of the features on which they depend. Use this operation to deploy an application with a new version of an existing feature, to force applications that reference the existing feature to use the new version, or if after clicking Deploy you get an error that says that because the node is running in stable mode, it cannot accept the deployment of the application. – **Force Deploy** - Dependencies on target product applications are not checked and validation errors ignored. May result in broken applications and should be used with caution.

Option	Procedure
	3. Click **Deploy** to deploy the application or **Cancel** to cancel the deployment.

Results

The applications are deployed and if auto-provisioning is enabled, those applications that provide implementation or binding types to the applications being deployed are also automatically deployed to the target nodes.

If you did not choose the **Deploy with Start** option, application artifacts for the deployment are not created.

CLI

Before you begin

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information about the command-line interface (CLI) and the required build and data files.

Procedure

1. In the data file, specify Environment and Application elements in base format.

```
<Environment xsi:type="amxdata:Environment" name="envName" >
 <Application xsi:type="amxdata_base:Application_base" name="app" />
</Environment>
```

2. In the build file, set the `action` attribute of the `AMXAdminTask` element set to **`deploy`** and the `objectSelector` attribute to **`Environment/Application`**. To deploy without starting the application, specify the `options` attribute and set the value to **`nostart`**.

```
<AMXAdminTask action="deploy" [options="nostart"]
objectSelector="Environment/Application" />
```

3. Specify the location of the EAR file.

```
<DAA xsi:type="amxdata:DAA" description="My DAA for
      application test" location="c:/EARFiles/PubSub.ear" />
```

4. Invoke the command-line interface on the build file.
   - The application is deployed and started.
   - If the application is a dependent application and its target application has been deployed, the application is deployed and started. If the target application is not deployed, the deployment will fail.
   - If an application is a target application, it and all its dependent applications are deployed and started.

Undeploying Applications

About this task

When you undeploy an application, the system queues the request and applies it to components as they become available.

GUI

Procedure

1. Click the **Applications** button.
2. In the Applications list, click one or more applications.
3. Choose an undeploy option.

Option	Procedure
Undeploy	1. Do one of the following: – Click **Undeploy**. – Select **Undeploy**▾**Undeploy** 2. If any of the selected applications has dependencies, the Application Dependencies to Undeploy dialog displays with target applications. 3. Check the checkboxes next to the target applications to undeploy. 4. Click **Undeploy**. The selected target applications are undeployed. If a node is stopped then the undeploy process will wait until the node starts.

Results

The application is undeployed.

if the application is undeployed when it is in the stopped state, the deployment artifacts for the application are not deleted.

CLI

Before you begin

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information about the command-line interface (CLI) and the required build and data files.

Procedure

1. In the data file, specify Environment and Application elements in base format.

```
<Environment xsi:type="amxdata:Environment" name="envName" >
 <Application xsi:type="amxdata_base:Application_base" name="app" />
</Environment>
```

2. In the build file, set the `action` attribute of the `AMXAdminTask` element to **`undeploy`** and the `objectSelector` attribute to **`Environment/Application`**. To perform a force undeploy, specify the `-force` option.

```
<AMXAdminTask action="undeploy" objectSelector="Environment/Application" [-force] />
```

3. Invoke the command-line interface on the build file.

Redeploying Applications

Any changes to the following attributes of an application changes the configuration of the affected component or the application and the state transitions to **Out of Sync** in the ActiveMatrix AdministratorUI. These components or applications are re-synchronized by re-deploying them.

- Substitution variables for an application
- Adapter SDK properties for an Adapter component
- Adapter instance variables
- Adapter component software version
- Re-distribution of components to nodes

Results
The application components have to be re-deployed.

Starting Applications

When an application is first deployed and started, deployment artifacts such as the application, repository and data files are created in the folder *CONFIG_HOME*/ext/*enterpriseName*/environments/*environmentName*/nodes/*nodeName*. Do not modify or delete any files from this folder.

GUI

Procedure

1. Click the **Applications** menu item.
2. In the Applications list, click one or more applications.
3. Click **Start**.
   If the Implementation Type for TIBCO Adapters application is not runnning, the Application Dependencies to Start dialog displays.
4. Check the checkboxes next to the Implementation Type for TIBCO Adapters application to start.
5. Click **Start**.
   The Implementation Type for TIBCO Adapters application is started. The selected applications are started after the Implementation Type for TIBCO Adapters application is running. The Runtime State of the selected applications changes to Starting.
6. Click the  until the Runtime State changes to Running.

Results

The selected applications are started.

CLI

Before you begin

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information about the command-line interface (CLI) and the required build and data files.

Procedure

1. In the data file, specify Environment and Application elements in base format.

```
<Environment xsi:type="amxdata:Environment" name="envName" >
 <Application xsi:type="amxdata_base:Application_base" name="app" />
</Environment>
```

2. In the build file set the `action` attribute of the `AMXAdminTask` element to **`start`** and the `objectSelector` attribute to **`Environment/Application`**.

```
<AMXAdminTask action="start" objectSelector="Environment/Application" />
```

3. Invoke the command-line interface on the build file.

Results

The selected applications and target applications are started.

It is likely that the status of the application and Implementation Type for TIBCO Adapters application is not immediately reflected in the Administrator UI.

Stopping Applications

GUI

Procedure

1. Click the **Applications** button.
2. In the Applications list, click one or more applications.
3. Choose a stop option.

Option	Procedure
Stop Allows applications to complete processing before shutting down. Applications may take anywhere from a few seconds to an hour to stop.	1. Do one of the following: – Click **Stop**. – Select **Stop**→**Stop** 2. Click **Stop**. Components with dependencies will be stopped only after the components they depend upon have stopped.

The selected applications are stopped.

CLI

Before you begin

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information about the command-line interface (CLI) and the required build and data files.

Procedure

1. In the data file, specify Environment and Application elements in base format.

```
<Environment xsi:type="amxdata:Environment" name="envName" >
 <Application xsi:type="amxdata_base:Application_base" name="app" />
</Environment>
```

2. In the build file, set the `action` attribute of the `AMXAdminTask` element to **`stop`**, the `options` attribute to **`immediate`** and the `objectSelector` attribute to **`Environment/Application`**.

```
<AMXAdminTask action="stop" objectSelector="Environment/Application" />
```

3. Invoke the command-line interface on the build file.

Results

The selected applications and target applications are stopped.

It is likely that the status of the application and target applications is not immediately reflected in the Administrator UI.

Logging

Refer to the *Administration* book for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus for information the logging architecture.

A logger associates a runtime object with an appender, specifies the types of events to be logged, and whether to pass messages to a parent logger. Each host, node, and application can have a logging configuration and each logging configuration has a root logger. The logging level is specified for each appender that belongs to a logger. This lets a logger to send logs to different destination with a different level. You can use the Administrator graphical and command-line interfaces to create loggers and appenders and to add appenders to existing loggers.

Logging Appenders

TIBCO ActiveMatrixruntime objects—hosts, nodes, and applications—use log4j technology to output log statements to a variety of output targets. In log4j, a target is called an *appender*. TIBCO ActiveMatrix supports the following logging appender types: clear text file, Common Base Event (CBE) format file, and JMS. Events logged to a JMS appender are stored in a database. Logging appenders can be referenced by multiple logging configurations. You can create the following types of logging appenders:

- **File** Appends events to a log file.
  - **Clear Text** - the log file is stored in clear text format.
  - **CBE** - the log file is stored in CBE format.
- **JMS** Appends events to a log service instance, which in turn stores the events to a database.

TIBCO ActiveMatrix Implementation Type for TIBCO Adapters provides an additional logging configuration for the node `com.tibco.amx.it.proxy`.

Enabling Logging

About this task

Follow these steps to enabling logging for the AdapterIT component.

Procedure

1. Navigate to a list of hosts, nodes, or applications.
2. Click 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.
5. Click **Add**.
   A row is added to the list.
6. In the Logger Name column, type a logging configuration name.
7. In the Additivity column, select an additivity.
8. Click **Set Appender**.
   A row is added to the list.
9. In the Level column, select a logging level.
10. In the Appender column, select the `com.tibco.amx.it.proxy` appender.
11. Click **Save and Apply**.

Applications Reference

Column	Description
Name	The name of an application. By default, the Applications list contains the platform applications as well as the ActiveMatrix Implementation Type for TIBCO Adapters application. The platform application contains the default binding type implementations, platform services, and data binding converters. The platform application is dependent on the TIBCO ActiveMatrix Platform feature. Administrator automatically provisions this feature when amx.platform-app is deployed. Similarly when an EAR is deployed the ActiveMatrix Implementation Type for TIBCO Adapters application is automatically provisioned.
Application State	The runtime state of the application. The application state is a rollup of the state of its constituent components and bindings. • Not Deployed - before an application is deployed. • Deployed - after the first time the application is deployed. • Partially Undeployed - while an application is being undeployed. • Partially Running - the application is deployed to more than one node and not all the nodes are running. • Starting • Start Failed - click the **Action History** link to get more information. • Running • Stopped - after the application has been started and stopped. • Stopping • Partially Stopped - the application's components and bindings are in different states. • Uninstalling • Partially Uninstalled - not all the components and bindings of the application have been uninstalled. • Waiting for Dependencies - Either a resource instance or application that this application depends upon is not running. Once all dependencies are running, the components which are waiting will automatically be started. • Preparing for Undeploy - the process of undeploying the application has started. • Partially Ready for Undeploy - Some components have completed processing and have been marked as ready for undeploy, but other components in the application have yet to complete processing. • Unknown
Last Deployed On	The date that the application was last deployed.
Synchronization	Indicates whether the runtime has the latest configuration for the application.
Action History	The outcome of the last action performed with the intent of affecting the runtime state.  The action history does not reflect the component state of ActiveMatrix Implementation Type for TIBCO Adapters .

Editable Properties Reference

The editable properties defined for an application are accessed by selecting an application and navigating to the **Properties** tab and clicking the **Editable Properties** link. These properties are specific to the TIBCO Adapter used.

Field	Read-only?	Description
Owner	Y	Name of the application or binding.
Owner Type	Y	The owner of the property.
Property Name	Y	The name of the property.
Property Type	Y	The type of the property. Either string or a resource template type.
Property Value	N	The value of the property.

When the value of a property is changed, the corresponding component will be marked as Out of Sync. When the application is redeployed, only these components will be restarted.

Application Substitution Variables Reference

The Substitution Variables tab displays the application's components, promoted services and references, and bindings in a hierarchical list. The information displayed on the right hand side matches the object selected from the hierarchical list.

The global variables of the EAR file are mapped to the application substitution variables.

Use the Add button to add variables for use in properties or logging configurations or the Delete button to remove variables so they can be resolved at another level, such as the environment.

Table 3: Substitution Variables

Property	Required?	Editable?	Description
Name	Y	Y	The name of the substitution variable.
Type	Y	Y	The type of the substitution variable. One of • String • Integer • Boolean • Password Default: String.
Description	N	Y	Description of the substitution variable.
Local Value	Y	Y	The local value or the substitution variable.

Property	Required?	Editable?	Description
Substitution Variable Name	Y	Y	The name of the substitution variable.
Node Name	Y	Y	The node on which the value for this substitution variable is defined. This property is available only on the **Application Fragment Substitution Variables** link.
Type	Y	Y	The type of the substitution variable. One of • String • Integer • Boolean • Password Default: String.
Description	N	Y	Description of the substitution variable.
Local Value	Y	Y	The local value or the substitution variable.

When the value of a substitution variable is changed, the corresponding component will be marked as **Out of Sync**. When the application is redeployed, only these components are restarted and the **Out of Sync** cleared.

Component General Reference

Details

Property	Required?	Editable?	Description
Description	N	N	The name of the component.
Component Type	N	N	The type of the component.
Component Version	N	N	The version of the component.
Synchronization	N	N	Indicates whether the runtime has the latest configuration for this component.
Node Name		N	The node on which an instance of the component is running.
Component State		N	The state of the component.
Component Action History		N	The outcome of the last action.

Properties

The TIBCO Adapter SDK properties are displayed. The property values can be modified from the **Application** > **Properties** tab.

The application should be redeployed after modifying a property.

For more information refer to the *Administration* guide for TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus.

Hawk Methods

To view the Hawk methods for an Adapter IT component, the component should be in the running state and the application of which the component is a part of enabled for Hawk monitoring where the `HawkEnabled` substitution variable is set to `true`. Additionally, the TIBCO Hawk agent should be running.

Property	Required?	Editable?	Description
Node	Y	N	The name of the node where the component is deployed.
Method	Y	N	The name of the Hawk method. Click **Show All** for the list of available Hawk methods. Clicking on a Hawk method displays information about the method as well as the parameters required when invoking the method.
Name	Y	N	The name of the Hawk method.
Description	N	N	The description of the Hawk method.

Config Details

This tab displays configuration information for the installed TIBCO Adapter software used by the component in the selected node.

Tracing

The TIBCO Hawk agent has to be running to view the details on this tab.

Property	Description
Node	The node where the component is deployed.
Search	
File Name	The log file name. Choose a log file from the drop down list. Alternatively choose **custom** from the drop down list and specify the log file.
Lines to fetch	The number of lines to fetch from the log file.
Add Search Condition Click the **Add Search Condition** link to specify the search criteria. Choose a criteria from the drop down list and then specify values for the chosen criteria. If more than one search criteria is specified, choose to display entries where any or all conditions apply. Click **Search** to display the log entries.	
Configure Tracing	Click this button to configure the tracing information. In the dialog box that displays when you click this button, **1.** Check the checkboxes for the tracing roles you want displayed **2.** Click **Save**.
Export	To export trace entries, select the entries and click **Export**.

Component Configuration Reference

Software Versions

Property	Description
Node	The node where the component is deployed.
Software	Choose a software version from the drop-down list. If you choose a different software version, redeploy the application.
Synchronization	Indicates whether the node runtime matches the node's configuration in the Administrator database.

Service Substitution Variables Reference

This tab displays service level substitution variables. Instance specific variables are like global variables except that they can be set for each TIBCO Adapter instance. Variables can be made instance specific by marking them as *service settable* in the TIBCO Designer global variables tab.

When the value of a substitution variable is changed, the corresponding component will be marked as Out of Sync. When the application is redeployed, only these components will be restarted.

Property	Description
Name	The name of the variable.
Type	The type of the component.
Node	The node on which an instance of the component is running.
Description	The description of the variable.
Value	The value of the variable.

Chapter

3

Monitoring Applications

The Proxy IT framework uses TIBCO Hawk Agents and TIBCO Hawk Microagents (HMA) to monitor and manage the TIBCO Adapter processes launched by the Adapter IT components. A TIBCO Hawk Agent is a process that monitors activity on a particular machine using microagents which represent managed objects such as operating system subsystems, agent components, log files, event logs or applications which have embedded microagent interfaces. Each microagent exposes a set of methods to the agent that the agent uses to collect information and take action. TIBCO Hawk is part of TIBCO Runtime Agent (TRA).

Topics

Invoking Hawk Methods

About this task

Hawk microagent methods are used to collect information from and perform tasks on the ActiveMatrix Implementation Type for TIBCO Adapters component.

Procedure

1. Click **Applications**.
2. Select an application from the displayed list.
   The application details display.
3. Click **General** > **Hawk Methods**.
4. Select a node from the drop-down list.
5. Choose a Hawk Method.

   Click **Show All** for the list of available Hawk methods or search for the Hawk method name.
6. Click on the Hawk method.
   Details of the Hawk method display.
7. Specify values for the required parameters and click Invoke.
   The results are displayed in the **Method invocation result** dialog.

   Review the invocation results and click **Close**.

Lifecycle Management

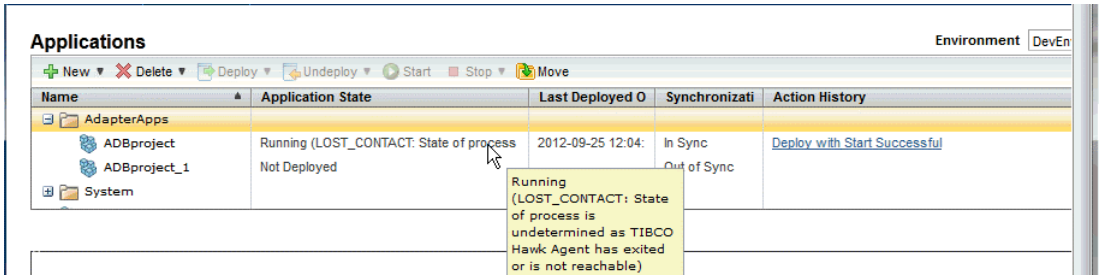
When a TIBCO Adapter application is deployed, the deployment artifacts are created in the *TIBCO_HOME* folder. The default Deploy with Start option deploys the EAR and start the TIBCO Adapter components.

The life-cycle functionality comprises of start and stop operations. These operations can be initiated on the TIBCO Adapter application or on individual components.

When a TIBCO Adapter application is stopped, all components that comprise the application are also stopped.

Extended State Management

The Implementation Type for TIBCO Adapters component extends the ActiveMatrix state to reflect the current state of the TIBCO Adapter processes. The Proxy IT framework provides a granular ActiveMatrix component state. This enhanced state is displayed in parenthesis along with the ActiveMatrix state as shown in the following figure.



See Extended States and Messages on page 46 for the completed list of extended messages.

Extended States and Messages

ActiveMatrix State	Extended State	Extended Message	Description
Running	RUNNING	Process is Running	A normal state transition to the RUNNING state. The TIBCO Adapter was started successfully.
	STARTED	Process is Started	An intermediate transition state when the component just starts and changes to a RUNNING state after a successful start.
	RUNNING_WITH_ERROR	Error details about the condition	The runtime problems encountered with the TIBCO Adapter while it is running. For example, the connectivity to the backend application is lost. The extended message shows complete details
	STOP_FAILED	Error details about the reason for not being able to stop	This is an user initiated stop request failed if the stop action did not make the runtime TIBCO Adapter process to exit
Stopped	STOPPED	Process is stopped	A normal state transition to the STOPPED state. The TIBCO Adapter was stopped successfully.
	EXITED	Process has exited	A normal state transition to the EXITED state if the TIBCO Adapter exits under normal circumstances.
	STOPPED_WITH_ERROR	Error details about the condition	The TIBCO Adapter stops due an error or this was the last error before the TIBCO Adapter stopped..
	START_FAILED	Error details about the reason for not	A user initiated start request fails.

ActiveMatrix State	Extended State	Extended Message	Description
		being able to start	
Running or Stopped	LOST_CONTACT	Details about its loosing contact with the running Hawk Agent	The TIBCO Adapter infrastructure has lost contact with the TIBCO Hawk Agent, possibly because TIBCO Hawk Agent has exited or is not reachable. The correct state will be applied when the connection with TIBCO Hawk Agent is restored

Appendix

A

Workflow using TIBCO ActiveMatrix Adapter Binding Type

About this task

The sections lists the high level steps involved in using TIBCO ActiveMatrix Adapter Binding Type to integrate the TIBCO Adapter services with that of the TIBCO ActiveMatrix Adapter Binding Type services and reference bindings.

Messages sent by the TIBCO Adapter publisher are received by the composite application via the service binding of TIBCO ActiveMatrix Adapter Binding Type and routed back to the TIBCO Adapter subscriber via the reference binding of the TIBCO ActiveMatrix Adapter Binding Type. The messages are transformed from the published message schema format to the subscribed message schema format using Mediation component flows so they can be processed by the TIBCO Adapter subscriber.

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO Adapter being used
- TIBCO ActiveMatrix Adapter Binding Type
- TIBCO ActiveMatrix Service Grid or TIBCO ActiveMatrix Service Bus

  Specifically the *Composite Development* and *Administration* books.

Procedure

1. Using TIBCO Designer, create a TIBCO Adapter project and configure two TIBCO Adapter instances.
2. Configure subjects for the TIBCO Adapter services.
3. Create an EAR file for the TIBCO Adapter configurations.
4. Using TIBCO Business Studio import the TIBCO Designer project into a SOA Project and generate a WSDL for it.
5. For the SOA project created in the previous step create a composite application and configure it with the TIBCO ActiveMatrix Adapter Binding Type service and reference bindings.

   Make sure you match subjects for the service and references to the corresponding subjects in the configurations specified for the project in TIBCO Designer.
6. Configure the Mediation flow and mapping.
7. Create the DAA.
8. Using ActiveMatrix Administrator create an application using the EAR file created in Step 3. Deploy and start this application.
9. Using ActiveMatrix Administrator create an application using the DAA created in the previous step.
10. Deploy and start the application.