TIBCO ActiveMatrix BusinessWorks™

Administration

Version 5.16.1 | February 2025

Document Updated: March 2025

Contents

TIBCO ActiveMatrix BusinessWorksTM Administration Introduction

This section describes administration of ActiveMatrix BusinessWorks™ and how to start and stop TIBCO Administrator.

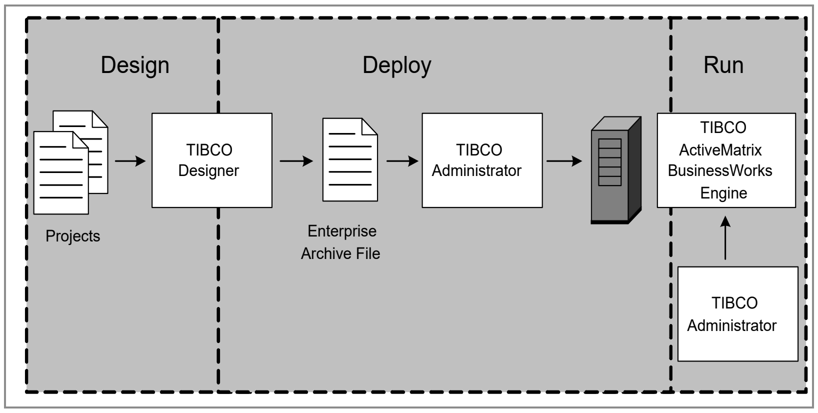
Overview of Administration

A ActiveMatrix BusinessWorks project has the following lifecycle:

- Design
- Deploy
- Run

Project lifecycle illustrates the lifecycle of a ActiveMatrix BusinessWorks project.

Figure 1: Project lifecycle



The design phase involves creating and testing the project using TIBCO Designer™. During the deployment phase, you use TIBCO Designer to create an Enterprise Archive file (EAR file) that contains the required resources for running the project. Then, you use TIBCO Administrator to deploy the project to the machine(s) where it will run. In the run time phase, the ActiveMatrix BusinessWorks process engine executes the process instances of the deployed project and TIBCO Administrator can be used to monitor and manage the process engine.

This manual describes the deployment and run time phases of the project lifecycle. The design phase is described in *TIBCO ActiveMatrix BusinessWorks™ Process Design Guide*.

TIBCO Administrator

TIBCO Administrator is a central administration server for TIBCO products. You create, deploy, and manage applications in TIBCO Administrator using a web-browser-based interface. There is also a set of command-line utilities available for creating EAR files and deploying applications. For more information about TIBCO Administrator, see *TIBCO Administrator User's Guide*.

Starting TIBCO Administrator on Microsoft Windows

To launch TIBCO Administrator, you first start the administration server.

Starting the Administration Server

Two Microsoft Windows Services must be running for the server to be available. After creating an administration domain, the services are installed and set to start automatically. To start the services the first time, navigate to the Services dialog, find the TIBCO administration server for your domain and click the **Start** button. Repeat for the TIBCO Hawk Agent service.

Note: If the TIBCO Hawk Agent is started as a service, mapped drives on the machine are not recognized by deployed services. The workaround is to start the TIBCO Hawk Agent from the command line.

Alternatively, to start on the command line:

Procedure

1. Start the administration server by typing the following into a command-line prompt:

```
% cd <install-path>\tibco\administrator\domain\<domain-name>\bin
% tibcoadmin_<domain-name>.exe
```

2. Start the TIBCO Hawk Agent, which performs the TIBCO Administrator monitoring functions, by typing:

```
% cd <install-path>\tibco\tra\domain\<domain-name>
% hawkagent_<domain-name>.exe
```

The following message appears when starting the administration server from the command line:

```
log4j:WARN No appenders could be found for logger
(org.apache.commons.digester.Digester).
```

The `log4j` logging class is included in the Tomcat web server that is bundled with TIBCO Administrator. TIBCO Administrator provides a logging mechanism and does not use the logging facility provided by `log4j`. The message can be ignored.

Starting the TIBCO Administrator GUI

You can launch the TIBCO Administrator GUI by entering the appropriate URL into your browser, or you can use the **Start** menu.

Starting from a Web Browser

Procedure

1. Open a web browser and connect to this URL for the TIBCO Administrator GUI:

```
http://<host-name>:<port>/administrator/servlet/tibco_administrator
```

— *host-name* is the name of the machine on which the administration server has been installed. If this is the same machine you are currently on, you can use `localhost` as the machine name.

— *port* is 8080 by default. If you have used the TIBCO Domain Utility to assign a different port, use that port number instead. If you created multiple domains on one machine, the port is incremented by 10 for each domain. For example, the second domain will use 8090.

Tip You can enter `http://`*<host-name>*`:8080`. This displays a list of domains, the port each domain is using, and the TIBCO software available on that port. You can pick a domain from this list to go to the login screen.

2. Log in. For the first login, this must be the user specified as the domain administrator user with the Domain Utility.

   You can then assign other users privileges to log in.

Starting from the Start Menu

To start TIBCO Administrator from the Start menu, your default browser must be set to one of the following:

- Google Chrome 32.0.x
- Microsoft Internet Explorer 9.0.x, 10.0.x, 11.0.x
- Mozilla Firefox 26.0.x, 28.0.x

Follow these steps:

Procedure

1. Select **Start > All Programs>TIBCO>TIBCO Administrator <***version***> > TIBCO Administrator**.
2. Log in. For the first login, this must be the user specified as the domain administrator user with the Domain Utility. That user can then assign other users privileges to log in.

Starting TIBCO Administrator on UNIX

To start TIBCO Administrator, you first start the administration server.

Starting the Administration Server

Procedure

1. First start the server by typing the following into a command-line prompt:

   ```
   % cd <install-path>/tibco/administrator/domain/<domain-name>/bin
   % tibcoadmin_<domain-name>
   ```

2. Then start the TIBCO Hawk Agent, which performs the TIBCO Administrator monitoring functions, by typing:

   ```
   % cd <install-path>/tibco/tra/domain/<domain_name>
   % hawkagent_<domain-name>
   ```

The following message appears when starting the administration server from the command line:

```
log4j:WARN No appenders could be found for logger
(org.apache.commons.digester.Digester).
```

The `log4j` logging class is included in the Tomcat web server that is bundled with TIBCO Administrator. TIBCO Administrator provides a logging mechanism and does not use the logging facility provided by `log4j`. The message can be ignored.

Starting the TIBCO Administrator GUI

Procedure

1. Open your web browser and connect to the following URL:

   ```
   http://<host-name>:<port>/administrator/servlet/tibco_administrator
   ```

   — *<host-name>* is the name of the machine on which the administration server has been installed. If this is the same machine you are currently on, you can use `localhost` as the machine name.

   — *<port>* is 8080 by default. If you have used the TIBCO Domain Utility to assign a different port, use that port number instead. If you created multiple domains on one machine, the port is incremented by 10 for each domain. For example, the second domain will use 8090.

✓ **Tip**

You can enter http://<*host-name*>:8080 to get a list of domains, the ports they are using, and the TIBCO software available on that port, and then pick a domain from this list to go to the login screen.

2. Log in as the domain administrator user. This user was specified using the Domain Utility. That user can then assign other users privileges to log in.

Stopping the Administration Server

You can stop the administrator server in several ways:

- On all platforms, from TIBCO Administrator, choose **Application Management > All Services Instances**. Select <*machine-name*> `- TIBCO Administrator` and click **Stop Selected**.
- If you started the administration server from a command line, you can use Control-C on any platform to stop the server.
- On Microsoft Windows, navigate to the `Services` panel. Select the administrator server service, and then click the **Stop** button.
- On UNIX, use the appropriate `kill` command for your system to stop the administrator server.

Using TIBCO Hawk

TIBCO Administrator is the preferred monitoring and management application for ActiveMatrix BusinessWorks. However, the process engine is instrumented with a TIBCO Hawk microagent that can be used to perform most administrative functions. For more information about using TIBCO Hawk to manage ActiveMatrix BusinessWorks process engines, see Performance Tuning.

Using TIBCO Enterprise Management Advisor

TIBCO Enterprise Management Advisor^{TM} (EMA) extends TIBCO Administrator and allows you to automate the management of resources in your enterprise. For example, your environment may include databases, application servers, ActiveMatrix BusinessWorks processes, and so on. Each resource in your enterprise may depend upon other components for continued operation. TIBCO EMA allows managed resources to automatically communicate their availability with each other and react accordingly.

ActiveMatrix BusinessWorks can receive notifications from TIBCO EMA about the status of resources such as databases, JMS servers, and other external resources a business process can depend upon. When a resource becomes unavailable, ActiveMatrix BusinessWorks suspends execution of processes that rely on the resource until the resource is once again available.

Using TIBCO EMA to manage resource dependencies with ActiveMatrix BusinessWorks processes can simplify exception handling in process definitions. Because TIBCO EMA handles resource availability issues, handling resource exceptions in process definitions can be minimized.

ActiveMatrix BusinessWorks automatically builds a list of resources that a process definition depends upon and communicates the list to TIBCO EMA. Communication between ActiveMatrix BusinessWorks and TIBCO EMA relies on TIBCO Hawk microagent methods.

To enable TIBCO EMA integration with ActiveMatrix BusinessWorks, both TIBCO Hawk and TIBCO EMA must be enabled with custom engine properties. The properties `bw.engine.emaEnabled` and `Hawk.enabled` must both be set to true. For more information about custom engine properties, see Custom Engine Properties.

You can use TIBCO Administrator to resume processes that have been suspended due to blocked resources. For more information, see Blocked Resources.

For more information about automating enterprise resource management, see *TIBCO Enterprise Management Advisor User’s Guide*.

Administration Tutorial

This section provides a short tutorial that shows how to use TIBCO Administrator to deploy and start an application that contains a <MadCap:variable name="productvar.productName" /> process definition.

Note: This section is meant to provide an introduction to the functionality, not comprehensive step-by-step instructions.

Tutorial Overview

This section walks through a simple example of how to do the following:

Procedure

1. Create the Enterprise Archive File
2. Create and Deploy the Application
3. Start the Application
4. Stop the Application

Prerequisites

This tutorial relies on the project created in *TIBCO ActiveMatrix BusinessWorks™ Getting Started*. Create the project described in that tutorial before continuing with the tutorial in this section.

Also, to perform the tasks in this tutorial, you must have installed and configured ActiveMatrix BusinessWorks and TIBCO Administrator software properly.

Procedure

1. Install all components of TIBCO Runtime Agent™ on your system.
2. Install all components of ActiveMatrix BusinessWorks on your system.

3. Install all components of TIBCO Administrator on your system.

Overview of Example Process

The project monitors a directory for a specific file. When the file changes, a new file is created that contains the contents of the original file plus the time the change was made to the original file. The new file is named after the change that occurred (`create.txt`, `modify.txt`, or `remove.txt`). If you modify the file multiple times, the new file overwrites the existing `modify.txt`.

Create the Enterprise Archive File

Before you can deploy a project, you must create an enterprise archive file in TIBCO Designer. Follow these steps:

Procedure

1. If your project is not currently open, start TIBCO Designer and choose **Open Existing Project** in the startup screen. Then, select the project you wish to deploy. If you worked with the project recently, you can also choose **Reopen Project**.
2. Select the top-level (`tutorial`) folder and drag an Enterprise Archive resource from the General palette into the design panel.

   Notice that the name is the same as the project name.
3. In File Location field, use the default value or click the **Browse** button and select a location and filename.
4. With the tutorial archive selected in the project panel, drag a Process Archive from the Process palette into the design panel.
5. In the Configuration tab:

   Change the name to `FileActivityTest` and click **Apply**.

   Select the Processes tab and click the browse (process starter) icon.

   Select the FileTest process you created earlier, click **OK**, and then click **Apply**.
6. Select the tutorial archive and click the **Build Archive** button in the left bottom corner.
7. Click **Project > Save**.

8. Click **Project > Exit**.

Create and Deploy the Application

This section explains how to use TIBCO Administrator to import an enterprise archive file and create a corresponding application.

Follow these steps:

Procedure

1. Start TIBCO Administrator and log into the administration domain in which you wish to deploy the application.
2. Click the **Application Management** module, and then click the **New Folder** tab.
3. In Name, type `File Test Application`.
4. Click **Save**.
5. Click the File Test Application folder, and then click the **New Application** button.
6. Click the **Browse** button to select the enterprise archive file you created in Create the Enterprise Archive File.
7. Click **OK** to load the EAR file.
8. In the dialog that appears, select the Deploy on Save check box, and then click the **Save** button.

   If you choose Deploy on Save, TIBCO Administrator uses the parameters specified in the project file and the default machine that registered the software in TIBCO Administrator.

   This example does not require further customization. For other cases, you may decide not to choose Deploy on Save so you can first configure the application.

The next dialog displays the application. In the left panel, expand the application and click Configuration. This displays the Configuration Builder and Deployed Configuration panels with the consoles created for the deployment. The `Configuration Builder` panel on the left allows you to customize the application configuration. The `Deployed Configuration` panel shows the deployed applications.

Start the Application

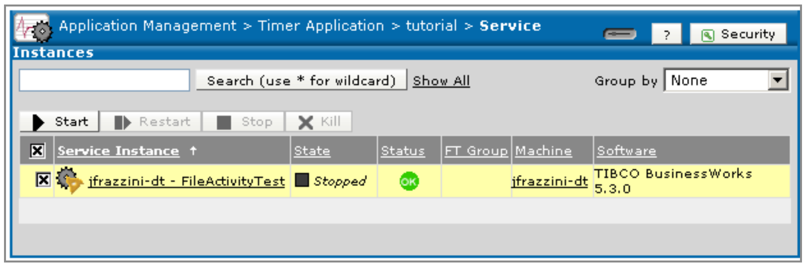
This section gives an overview of starting an application.

If you deployed using Deploy on Save, the tutorial application is actually started by default.

To start service instances, follow these steps:

Procedure

1. In TIBCO Administrator, select the application in the left panel, and then click Service Instances.


Application Management > Timer Application > tutorial > Service Instances
?
Security
Search (use * for wildcard)
Show All
Group by
None
Start
Restart
Stop
Kill

☒	Service Instance ↑	State	Status	FT Group	Machine	Software
☒	jfrazzini-dt - FileActivityTest	Stopped	OK		jfrazzini-dt	TIBCO BusinessWorks 5.3.0

2. In the Service Instances console, select the check box next to the Service Instance (named after the machine and the process archive) and click **Start**

   The State column changes to first show **Starting Up**, and then **Running**.

Monitor the Application

Monitoring an application can be done in two ways, discussed in this section:

- Viewing Default Monitoring Information
- Specifying a Custom Alert

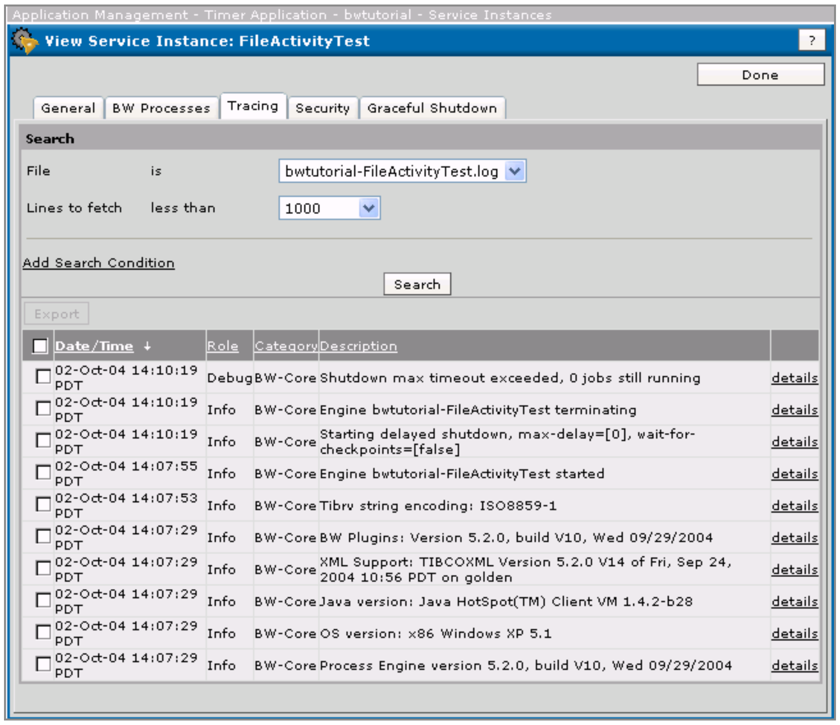
Viewing Default Monitoring Information

Some monitoring information for the application is available by default. To view the log from TIBCO Administrator, follow these steps:

Procedure

1. Choose **Application Management> Timer Application > tutorial > Service Instances** and select the <*machine*>-FileActivitiesTest service instance, and then the Tracing tab.
2. In the Search box, make sure that the log for your application (in this case `tutorial-FileActivityTest.log`) is selected, and then click **Search**.

   TIBCO Administrator displays information about the instance, which includes starting, termination, and any errors that occurred.



Application Management - Timer Application - bwtutorial - Service Instances

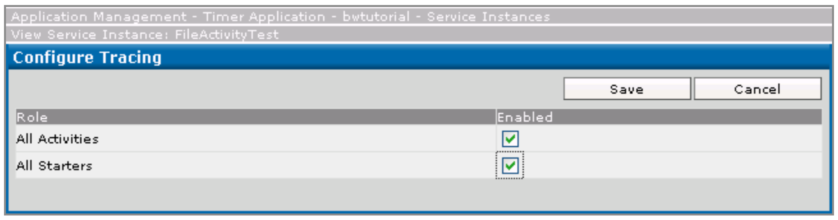
View Service Instance: FileActivityTest

Search

File	is	bwtutorial-FileActivityTest.log
Lines to fetch	less than	1000

Date/Time	Role	Category	Description	
02-Oct-04 14:10:19 PDT	Debug	BW-Core	Shutdown max timeout exceeded, 0 jobs still running	details
02-Oct-04 14:10:19 PDT	Info	BW-Core	Engine bwtutorial-FileActivityTest terminating	details
02-Oct-04 14:10:19 PDT	Info	BW-Core	Starting delayed shutdown, max-delay=[0], wait-for-checkpoints=[false]	details
02-Oct-04 14:07:55 PDT	Info	BW-Core	Engine bwtutorial-FileActivityTest started	details
02-Oct-04 14:07:53 PDT	Info	BW-Core	Tibrv string encoding: ISO8859-1	details
02-Oct-04 14:07:29 PDT	Info	BW-Core	BW Plugins: Version 5.2.0, build V10, Wed 09/29/2004	details
02-Oct-04 14:07:29 PDT	Info	BW-Core	XML Support: TIBCOXML Version 5.2.0 V14 of Fri, Sep 24, 2004 10:56 PDT on golden	details
02-Oct-04 14:07:29 PDT	Info	BW-Core	Java version: Java HotSpot(TM) Client VM 1.4.2-b28	details
02-Oct-04 14:07:29 PDT	Info	BW-Core	OS version: x86 Windows XP 5.1	details
02-Oct-04 14:07:29 PDT	Info	BW-Core	Process Engine version 5.2.0, build V10, Wed 09/29/2004	details

3. To make the default monitoring information more detailed, click the **Configure Tracing** button. The service instance must be running or the button is disabled.
4. In the window that appears, click All Activities to have execution of each activity included in the log; click All Starters to have execution of all starters included, and then click **Save**.



Application Management - Timer Application - bwtutorial - Service Instances

View Service Instance: FileActivityTest

Configure Tracing

Role	Enabled
All Activities	✓
All Starters	✓

5. When you return to the log, you will find that information about the individual activities and the starter are now included.

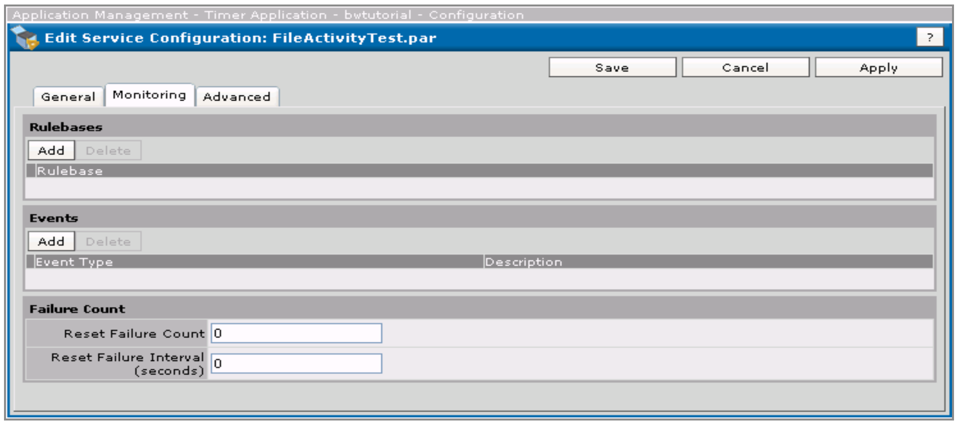
Specifying a Custom Alert

In addition to tracing, TIBCO Administrator allows you to specify that you wish to be alerted if certain conditions are met. This section gives one example. A detailed discussion of tracing is included in Adding a Rulebase to a Process or Service.

To specify a custom alert, follow these steps:

Procedure

1. Choose **Application Management> Timer Application > tutorial > Configuration.**
2. In the Configuration Builder panel, expand the tutorial application and select the `FileActivityTest.par`, and then choose the Monitoring tab.


Application Management - Timer Application - bwtutorial - Configuration
Edit Service Configuration: FileActivityTest.par
?
Save
Cancel
Apply
General
Monitoring
Advanced
Rulebases
Add
Delete
Rulebase
Events
Add
Delete
Event Type
Description
Failure Count

Reset Failure Count: 0
Reset Failure Interval (seconds): 0

3. In the `Events` pane, click **Add**.
4. In the dialog that appears, make the following changes (shown in the next figure).

   In the General pane, select First Component Failure.

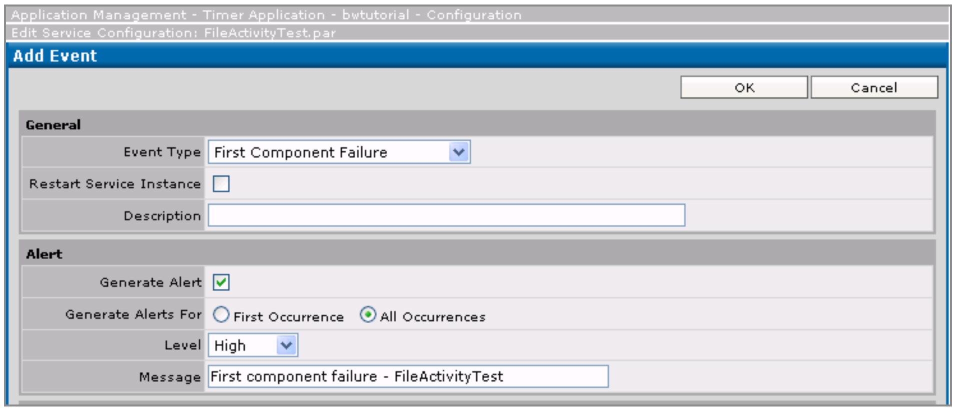
   In the `Alert` pane, select the check box next to Generate Alert.

   Select the All Occurences radio button and choose the level `High`.

   In the `Message` field, type `First component failure - FileActivityTest`.

   Click **OK**, and then click **Save**.

   TIBCO Administrator now displays a high-level alert with the message upon first failure of this process. If you wanted, you could also have an email sent in the event of component failure.


Application Management - Timer Application - bwtutorial - Configuration
Edit Service Configuration: FileActivityTest.par
Add Event
OK
Cancel

General

Event Type	First Component Failure
Restart Service Instance	☐
Description	

Alert

Generate Alert	✓
Generate Alerts For	First Occurrence [✓ All Occurrences]
Level	High
Message	First component failure - FileActivityTest

5. When you are returned to the Configuration Builder notice that it indicates that services require deployment. Click **Deploy** and the changes take effect.

Stop the Application

To stop the application, follow these steps:

Procedure

1. In the left panel of TIBCO Administrator, select the `Application Management` module.

2. Select either the All Service Instances panel, or choose **Application Management> Timer Application > tutorial > Service Instances**.

3. Select the check box next to the process engine you started earlier, and then click **Stop**.

Creating an Archive for Deployment

When you are ready to deploy your project, you must generate an Enterprise Archive, which contains the configuration for the process definitions you wish to deploy. You can upload the archive in TIBCO Administrator to deploy the associated application on the machine of your choice.

Overview

During development, you save your design to a project repository as needed. When you are ready to deploy your project to a machine, you must generate an Enterprise archive file (EAR file) using TIBCO Designer™.

The EAR file contains information on the resources you wish to deploy. This includes one or more ActiveMatrix BusinessWorks process definitions and the associated shared resources required by the process definitions.

Building an archive creates the EAR file that you can then deploy from TIBCO Administrator. If you make changes to the business processes or shared resources included in the archive, you must rebuild and redeploy the archive. Saving the project does not affect the archive.

You can define multiple Enterprise Archive resources for a single TIBCO Designer project. For each archive, you can choose the processes to include.

Enterprise Archive File Size

An EAR file can contain local project resources, `LibraryBuilder` resources, and files as specified in `AliasLibrary` resources. In addition, the TIBCO Designer classpath may include references to other files that are included in the EAR file. The EAR file size may become an issue when you build it in TIBCO Designer, load it into TIBCO Administrator and deploy it to remote machines.

An EAR file should only include resources that are required to deploy the project. Large archive files can have a negative effect at deployment. If each application in your project uses different resources, different `AliasLibrary` resources should be used by each application (rather than one large `AliasLibrary`).

- You should insure that the machine on which the EAR file is loaded and deployed has sufficient disk space.
- TIBCO Designer displays a warning when you add a directory to an EAR file, or alias to an `AliasLibrary` that references a directory. The EAR file size is typically one fifth of the warning, due to file compression. The warning reminds you that when referencing a directory, all files and sub directories in the directory are archived.

  You can modify the following property in `designer.tra` so that the warning appears only when files of the given size are loaded. The value is in megabytes.

  ```
  designer.ear.watermark.size=16
  ```

  This property specifies when the warning message should appear. For example, if you change the value to 32, the warning only appears if you are loading a files that are greater than 32 MB.

Warning: If the warning appears, you must understand the consequence of loading the files. You may need to increase the heap size value in `designer.tra` so Designer has enough memory to manage the project. At deployment, TIBCO Administrator copies the EAR file and deployment files to remote machines. If the EAR file is large, copying files may take extra time.

It is good practice to load only the files you need for your project. For example, if you require only one jar file of 500KB that exists in a directory that contains 20 MB of jar files, you should create an alias that references only the required jar rather than the entire directory.

Creating an Enterprise Archive

To create an Enterprise Archive, perform the following procedure:

Procedure

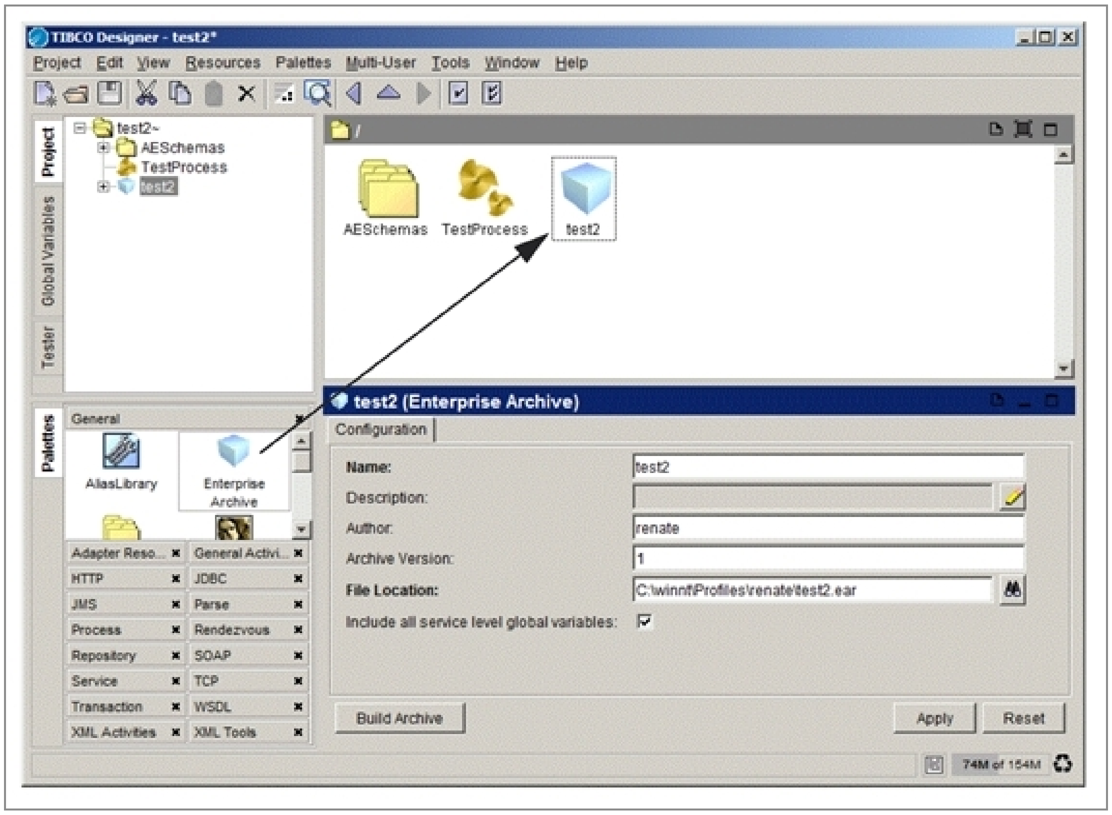
1. In TIBCO Designer, select a folder and find the Enterprise Archive resource.

   In palette mode, this resource is in the General palette.

2. Drag an Enterprise Archive into the design panel.

   The archive is displayed in the design panel, and the configuration panel allows you to supply information about the archive.

Figure 2: Adding an Enterprise Archive to your project



Name: test2
Description:
Author: renate
Archive Version: 1
File Location: C:\winnt\Profiles\renate\test2.ear
Include all service level global variables: ✓

3. Provide the following information:

Field	Description
Name	Name of the Enterprise Archive you are creating.
Description	Description of the archive content.
Author	Person creating the archive.
Archive Version	Version of the archive. **Note:** The user assigns this number. TIBCO Designer does not use this number.

Field	Description
File Location	Location where this archive will be saved. Click **Browse** to select a different file location than the default.
Include all service level global variables	Includes all global variables for which you clicked the Service check box. **Note:** This check box is provided for some adapters that do not properly report all their properties. TIBCO Designer cannot tell if that adapter is using a service level global variables or not. This check box explicitly forces inclusion of all service-level variables.

4. Create one or more Process Archives and add them to the Enterprise archive. This is described in Creating a Process Archive.
5. Build the archive by clicking the **Build Archive** button. When you click the button, TIBCO Designer creates an enterprise archive (`.ear`) file that you can then deploy from TIBCO Administrator.

Creating a Process Archive

Store the ActiveMatrix BusinessWorks process definitions that you wish to include in your application in a Process Archive resource. Process Archives are then added to Enterprise Archives.

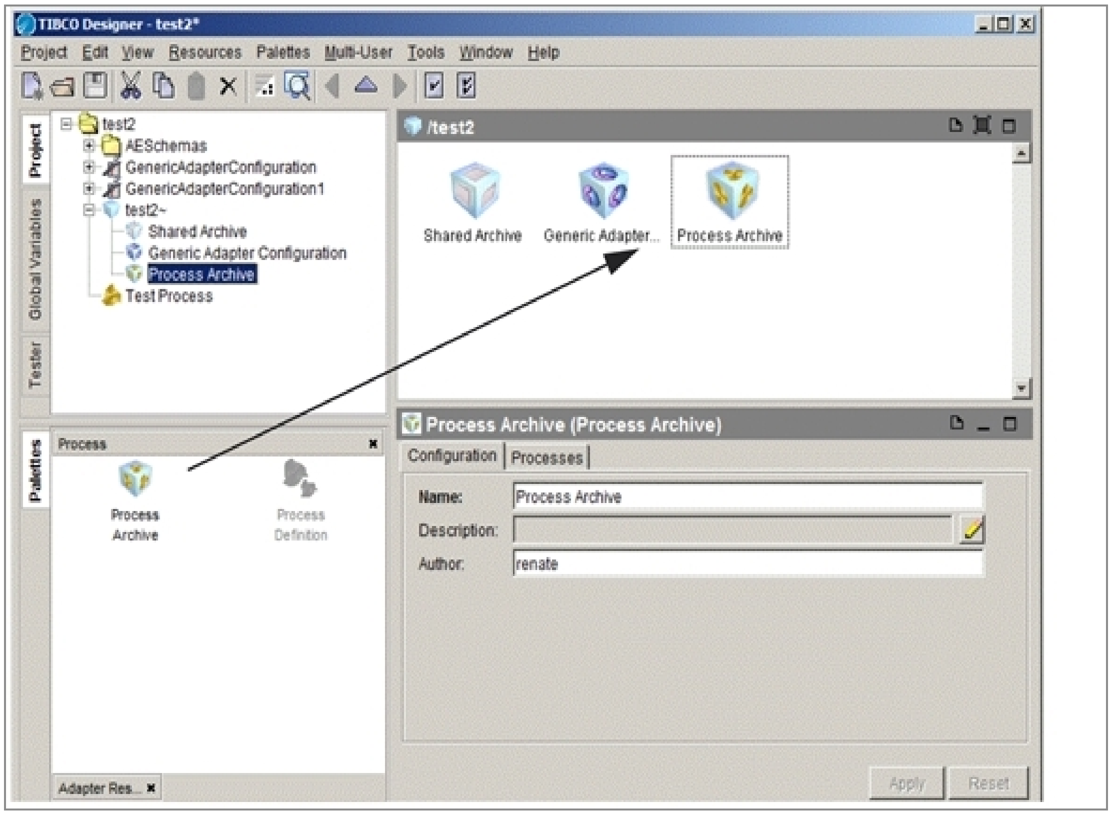
To create a Process Archive, perform the following procedure:

Procedure

1. Create and configure one or more process definitions that have process starters. For more information about creating and testing process definitions, see *TIBCO ActiveMatrix BusinessWorks™ Process Design*.
2. Select the Enterprise Archive resource in the project panel. If there are any processes that have process starters in your project, a Process Archive resource becomes available in the palette panel. If you're working in palette mode, it is located in the Process palette.

3. Drag the Process Archive into the design panel.

Figure 3: Adding a Process Archive to your Enterprise Archive



Name: Process Archive
Description:
Author: renate

4. Specify information in the `Configuration` tab, then click **Apply**:

Name	Name of the `Process Archive`.
Description	Optional description of the `Process Archive`.
Author	Optional author of the `Process Archive`.

5. Click the Processes tab to specify the process definitions to include. To include processes:

   Click the browse button (the binoculars).

   Select the process definitions you wish to add to the archive. Any process definitions called by the selected process (unless they are dynamically called) are automatically included in the archive. Explicitly add any dynamically called sub-processes.

Note: Do not explicitly add sub-processes that are not dynamically called. For more information about dynamically called sub-processes, see *TIBCO ActiveMatrix BusinessWorks™ Process Design*.

Repeat the procedure to add multiple processes.

Tip: You can select multiple process definitions by using shift-click or control-click to make contiguous or discontiguous selections)

6. Click **Apply**.
7. To build the archive, select the Enterprise Archive (which is one level higher in the project tree) and click the **Build Archive** button.

Making Changes to a Shared Archive

When you create an Enterprise Archive, a Shared Archive is automatically included in the archive and becomes visible when you select the Enterprise Archive. By default, TIBCO Designer adds any resources that are referenced by process definitions for which you created archives. This may include, for example, custom schema resources, TIBCO Rendezvous or JMS Connection configurations, JDBC Connection configurations, and so on.

Note: All JDBC connections are included in the Shared Archive automatically when the EAR includes a process archive.

In some cases, you may wish to add resources that are not automatically included.

To change a Shared Archive, perform the following procedure:

Procedure

1. Select the Shared Archive and provide a name and description if desired.
2. Click the Shared Items tab.
   — Click the **Browse** button next to the Include from Local Project field to include items from the currently loaded project (this should not usually be necessary).

- Click the **Browse** button next to the Include from Filesystem field to include other items.

3. Click **Apply**.
4. To build the archive, select the Enterprise Archive (which is one level higher in the project tree) and click the **Build Archive** button.

Creating and Deploying Applications

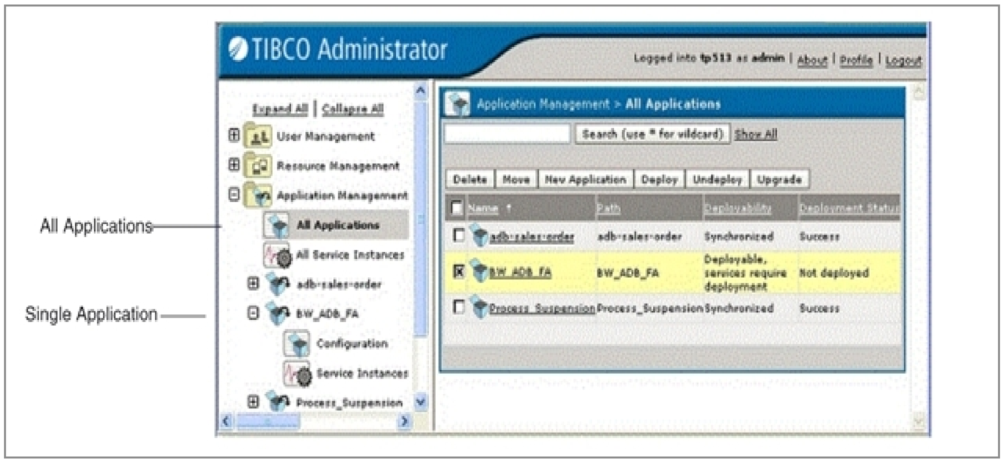
The TIBCO Administrator Application Management module allows you to create and deploy applications using the corresponding archive files.

Application Management Overview

The Application Management module allows you to create and deploy applications, and then start, stop, and monitor them.

This module is only available if you purchased the TIBCO Administrator Enterprise Edition and enabled the user and resource management option. You can enable the option using the TIBCO Domain Utility when creating an administration domain. For more details, see the TIBCO Administrator documentation .

The module contains the applications you have imported into TIBCO Administrator. You can view all applications in the All Applications dialog or all process engines in the All Services dialog. Alternatively, you can configure and manage the process engines for an application under the application's dialog.



Name	Path	Deployability	Deployment Status
adb-sales-order	adb-sales-order	Synchronized	Success
BW_ADB_FA	BW_ADB_FA	Deployable, services require deployment	Not deployed
Process_Suspension	Process_Suspension	Synchronized	Success

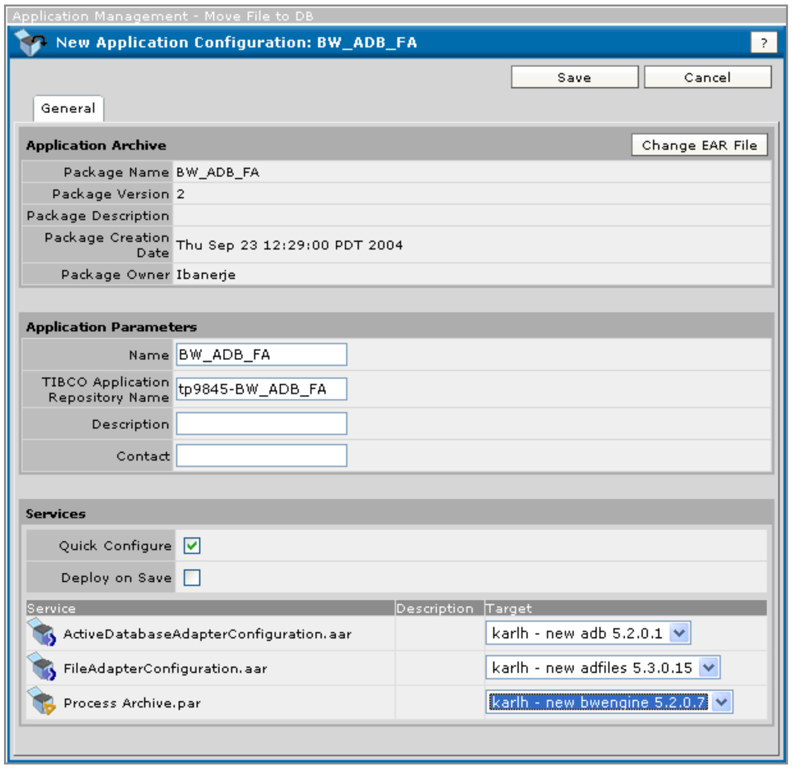
Creating an Application

To create an application in TIBCO Administrator, you must import an enterprise archive file created in TIBCO Designer. For more information, see Creating an Archive for Deployment.

To create an application, perform the following procedure:

Procedure

1. Select either **Application Management**, a previously created folder or **All Applications**.
2. Click **New Application**.
3. Click **Browse** and select an enterprise archive file, and then click **OK**. For example, the next diagram shows a new application, ready for deployment.



Application Management - Move File to DB

New Application Configuration: BW_ADB_FA

Application Archive

Package Name	BW_ADB_FA
Package Version	2
Package Description	
Package Creation Date	Thu Sep 23 12:29:00 PDT 2004
Package Owner	Ibanerje

Application Parameters

Name	BW_ADB_FA
TIBCO Application Repository Name	tp9845-BW_ADB_FA
Description	
Contact	

Services

Quick Configure	✓
Deploy on Save	☐

Service	Description	Target
ActiveDatabaseAdapterConfiguration.aar		karlh - new adb 5.2.0.1
FileAdapterConfiguration.aar		karlh - new adfiles 5.3.0.15
Process Archive.par		karlh - new bwengine 5.2.0.7

4. Click **Save**.

Once an application is created, you specify deployment configuration information, such as which machines should run which process engines in the application. Then the application can be deployed and the process engines can be started.

You can use the same enterprise archive file to create multiple applications and configure and deploy each application separately with different deployment options. This allows you to balance the load of the application across multiple machines.

You can also modify a deployed application and redeploy it. Also, you can revert to an earlier deployment if the changes you made do not have the desired result.

You have the following choices when creating an application:

- You can verify application information and make choices in the fields that allow input. If you wish, you can select a different archive file by clicking **Change EAR File**.
- If **Quick Configure** is selected, the services are bound to the targets selected in the target field.
- If **Quick Configure** is selected, **Deploy on Save** can be selected. When you click **Save**, the application is created and immediately deployed to the target machines specified in the Services pane, Target column. All variables, logging and other configuration values will use defaults defined in the archive file. The next screen displays the deployment status.

  If **Deploy on Save** is not selected, the application must be explicitly deployed using the application’s Configuration Console that displays upon save. This allows you to change settings, such as global variable settings and runtime options before deploying.

Deleting an Application

When you delete an application, all files associated with that application are removed and it becomes unavailable in TIBCO Administrator. You must upload the related enterprise archive file again to recreate the application.

A deployed application must be undeployed before it can be deleted.

To delete an application, perform the following procedure:

Procedure

1. Click **Application Management** or **All Applications**.
2. Select the application to delete.
3. Click **Delete**.

4. Click **OK** in the confirmation dialog.

Deploying an Application

When you deploy an application, TIBCO Administrator copies any required configuration information to the target machine and creates services to run the deployed process engines. You can automatically start any deployed services after the application is successfully deployed.

You can make changes to a deployed application, and then deploy the changed application. The currently deployed application can continue to run while you make changes. When you deploy the updated application, the current application is automatically undeployed. You can revert to a previously deployed application, if the changes you made need be rolled back.

You may want to change:

- The machines where the processes or services are deployed. See Enabling a Process or Service to Run on Other Machines.
- The monitoring behavior. See Adding a Rulebase to a Process or Service.
- Runtime variables. See Changing Runtime Variables for a Process or Service.
- Checkpointing behavior for ActiveMatrix BusinessWorks. See Changing the Checkpoint Data Repository for a Process.
- How ActiveMatrix BusinessWorks processes are executed. See Controlling Execution of TIBCO ActiveMatrix BusinessWorks Services.

To deploy an application, perform the following procedure:

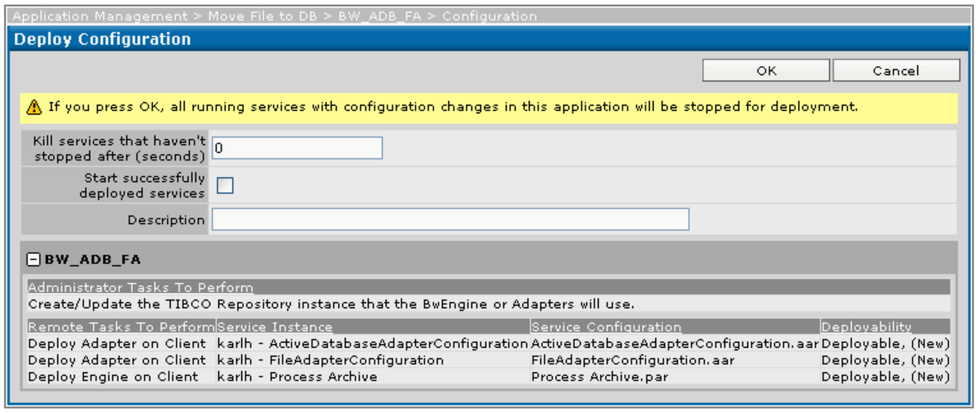
Procedure

1. Click **Application Management**.
2. Click *ApplicationName* **> Configuration**, where *ApplicationName* is the application created when you loaded the enterprise archive file. See Creating an Application.
3. Before deploying, you can:
   — Click the process archive (`.par`) and make changes, as outlined under Changing Application Global Variables and Repository Properties.

- Click the service instance (<*machine-name*>`-Process Archive`), to change server settings. See Edit Service Instance Dialog.

4. Click **Deploy**.

   The dialog similar to the following displays and informs you that all running processes with configuration changes in this application will be stopped when you click **OK** to deploy. If processes deploy successfully, they are restarted automatically if the corresponding check box was selected.



Application Management > Move File to DB > BW_ADB_FA > Configuration

Deploy Configuration

⚠ If you press OK, all running services with configuration changes in this application will be stopped for deployment.

Kill services that haven't stopped after (seconds)	0
Start successfully deployed services	☐
Description	

BW_ADB_FA

Administrator Tasks To Perform

Create/Update the TIBCO Repository instance that the BwEngine or Adapters will use.

Remote Tasks To Perform	Service Instance	Service Configuration	Deployability
Deploy Adapter on Client	karlh - ActiveDatabaseAdapterConfiguration	ActiveDatabaseAdapterConfiguration.aar	Deployable, (New)
Deploy Adapter on Client	karlh - FileAdapterConfiguration	FileAdapterConfiguration.aar	Deployable, (New)
Deploy Engine on Client	karlh - Process Archive	Process Archive.par	Deployable, (New)

   The dialog allows you to add a description and displays information about the application and each service.

5. Click **OK** to deploy the application, or click **Cancel** to choose an advanced configuration, different archive file, or make other changes.

Deploying Applications Manually

Scripting tools are used to build an EAR file for an application configured in TIBCO Designer. The application can then be loaded into one or more TIBCO Administrator administration domains.

Deployment options can be specified in a deployment configuration file that is created using the `AppManage` utility.

The AppManage utility, creates an XML based deployment configuration file in which the deployment options can be defined. The utility also uploads the deployment file and EAR file into the TIBCO Administrator administration domain.

For instructions about using the AppManage scripting utility to deploy applications, see *TIBCO Runtime Agent™ Scripting Deployment User’s Guide*.

Reverting to a Previously Deployed Application

When you revert an application, you select a different version of the currently deployed application to deploy. When you deploy, service instances and process engines are stopped, updated, and restarted. Any component that is removed from a machine as a result of the revert operation is undeployed from that machine.

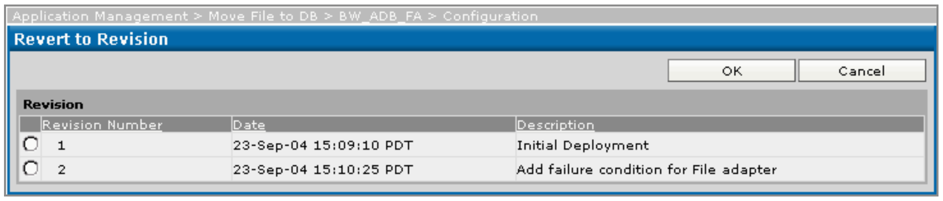
Note: Reverting an application is only possible if you have deployed an application more than once.

To revert to a previously deployed application, perform the following procedure:

Procedure

1. Click **Application Management**.
2. Click *ApplicationName* **> Configuration**, where *ApplicationName* is the application created when you loaded the archive. See Creating an Application.
3. In the Configuration Builder panel, click **Revert**.

   The deployed revisions and the time at which each was deployed display.



Application Management > Move File to DB > BW_ADB_FA > Configuration

Revert to Revision

Revision

	Revision Number	Date	Description
○	1	23-Sep-04 15:09:10 PDT	Initial Deployment
○	2	23-Sep-04 15:10:25 PDT	Add failure condition for File adapter

4. Click the button next to the revision you wish to use.
5. Click **OK**.

   The application is now shown as ready to deploy in the Configuration Builder.
6. Click **Deploy**.

Undeploying a Deployed Application

When you undeploy a deployed application, TIBCO Administrator stops all running process engines and removes them from the list of services that can be started. In effect, it completely removes all traces of the deployment (with the exception of the logs).

Note: To use an earlier version of the deployment configuration, choose Revert, not Undeploy. See Reverting to a Previously Deployed Application .

To undeploy an application, perform the following procedure:

Procedure

1. Click **Application Management**.
2. Click *ApplicationName* **> Configuration**, where *ApplicationName* is the application created when you loaded the archive.
3. In the Configuration panel, click **Undeploy**.
4. Click **OK** to undeploy, or **Cancel** to stop the operation.

Undeploy Dialog

Kill services that haven’t stopped after (seconds)

Specify the amount of time to wait before stopping process engines that have not stopped. The default is zero, meaning no time is allowed for a graceful shutdown, if previously set for a process engine.

Administrator Tasks To Perform

Lists the tasks that TIBCO Administrator will perform for this server if you choose to deploy by selecting the OK button.

Remote Tasks To Perform

Lists the tasks to perform on the selected machine (which could actually be the local machine) in the following fields:

- Service Instance.
- Service Configuration.
- Deployability — Shows whether the application has been deployed before.
- Task — Actions the deployment process performs on the target machine(s).

Viewing Application Deployment History

You can view a history of each time an application has been deployed.

To view deployment history, perform the following procedure:

Procedure

1. Click **Application Management**.
2. Select an application.
3. Click **Configuration**.
4. Click **History**.
5. Click **Details** for more information.

Managing Folders

If the structure of the applications you expect to manage using TIBCO Administrator is complex, you can organize the applications into folders. After creating a folder, you can create other folders, or add applications to the folder.

When you delete a folder, the folder contents are also deleted.

Note: To move a folder, you must have Administer permissions on the source folder (including its contents) and the destination folder.

To create a folder, perform the following procedure:

Procedure

1. Select either **Application Management**, or a previously created folder.
2. Click **New Folder**.
3. Provide a folder name and, optionally, a description and contact.
4. Click **Save**.

To delete a folder, perform the following procedure:

Procedure

1. Select the folder’s parent, either **Application Management**, or a previously created folder.
2. Select the folder to delete.
3. Click **Delete**.
4. Click **OK** in the confirmation dialog.

To move a folder, perform the following procedure:

Procedure

1. Select the folder’s parent, either **Application Management**, or a previously created folder.
2. Select the folder to move.
3. Click **Move**.

Moving an Application to a Folder

If you wish to organize your applications into folders, or need to move your application for other reasons, you can do so from the `Application Management` console.

Note: To move an application, you must have Administer permissions on the application and the destination folder.

To move an application to a folder, perform the following procedure:

Procedure

1. Click **Application Management** or **All Applications**.
2. Create folders if desired.
3. Select the application you wish to move.
4. Click **Move.**
5. You are prompted for the desired location of the application.
6. Click **Save** to make the change.

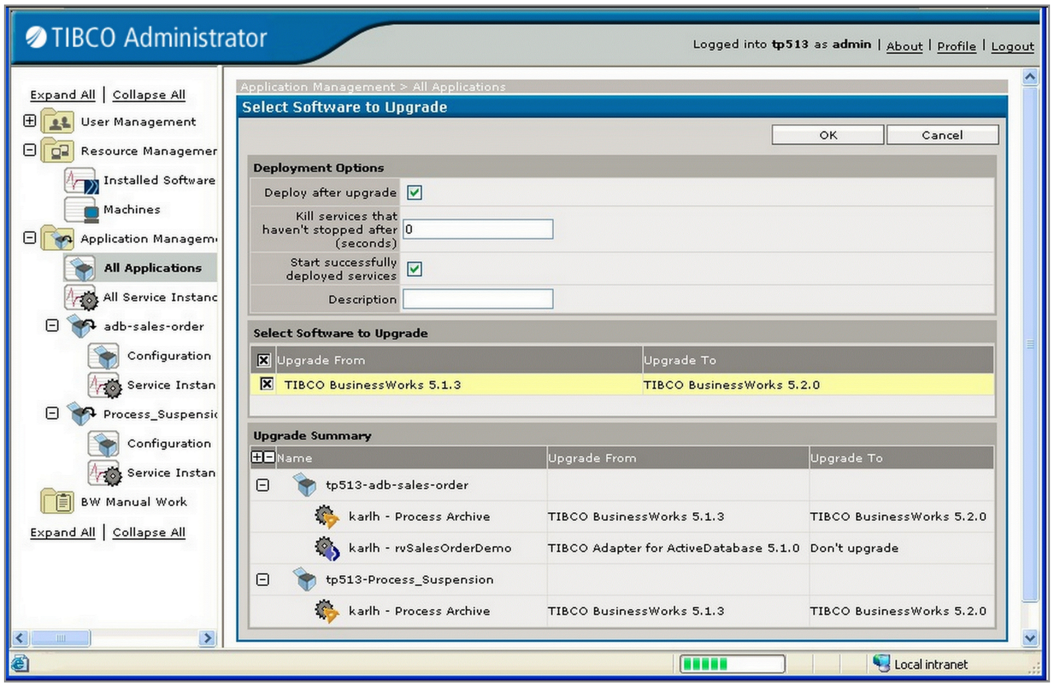
Upgrading an Application

If you have installed a new version of TIBCO software on a machine that is part of your administration domain, and the software is used in one or multiple applications, you can use the Upgrade feature to enable the applications to use the upgraded software.

The Upgrade feature remaps properties in the property files of the process engines to use the new software targets. The upgrade operation is not reversible. That is, you cannot revert to using the previous software version after upgrading.

You must redeploy your applications after upgrading.

The next diagram shows the dialog that displays when upgrading software.



Select Software to Upgrade

Deployment Options

Deploy after upgrade	✓
Kill services that haven't stopped after (seconds)	0
Start successfully deployed services	✓
Description	

Select Software to Upgrade

☒	Upgrade From	Upgrade To
☒	TIBCO BusinessWorks 5.1.3	TIBCO BusinessWorks 5.2.0

Upgrade Summary

Name	Upgrade From	Upgrade To
tp513-adb-sales-order		
karlh - Process Archive	TIBCO BusinessWorks 5.1.3	TIBCO BusinessWorks 5.2.0
karlh - rvSalesOrderDemo	TIBCO Adapter for ActiveDatabase 5.1.0	Don't upgrade
tp513-Process_Suspension		
karlh - Process Archive	TIBCO BusinessWorks 5.1.3	TIBCO BusinessWorks 5.2.0

To upgrade an application, perform the following procedure:

Procedure

1. Click **Application Management**.
2. Click *ApplicationName* **> Configuration**, where *ApplicationName* is the application you wish to upgrade
3. In the Configuration panel, click **Upgrade**.
4. Select **Deploy after upgrade** to redeploy your application as part of the upgrade. You can redeploy later, if necessary.
5. Select Start successfully deployed services to deploy and start your service instances and process engines. If Deploy after upgrade is not selected, this option is not available.
6. Select the software to upgrade.
7. Review the upgrade summary.
8. Click **OK**.

> **Note:** Validate your project and rebuild the EAR file before upgrading to a newer version of BusinessWorks.

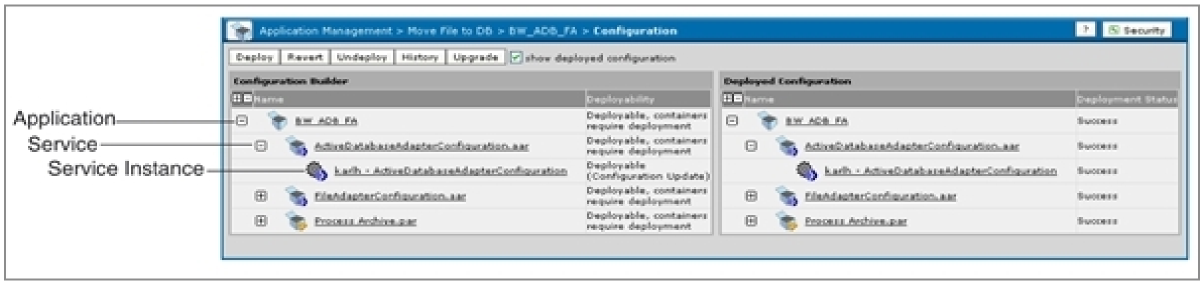
Setting Deployment Options

This section explains how to use the configuration builder to manage deployment options.

Configuration Console Overview

When you create an application, the enterprise archive file you import has values defined for global variables. The process engines in the archive have configuration options set as well. When you deploy the application, you can use the options set in the archive, or change options in TIBCO Administrator.

The Configuration console consists of two panes, Configuration Builder and Deployed Configuration. Each pane contains applications, service configurations, and service instances as shown in the next diagram.


Application
Service
Service Instance
Application Management > Move File to DB > BW_ADB_FA > Configuration
?
Security
Deploy
Revert
Undeploy
History
Upgrade
show deployed configuration

Configuration Builder

Name	Deployability
BW_ADB_FA	Deployable, containers require deployment
ActiveDatabaseAdapterConfiguration.aar	Deployable, containers require deployment
karlh - ActiveDatabaseAdapterConfiguration	Deployable (Configuration Update)
FileAdapterConfiguration.aar	Deployable, containers require deployment
Process Archive.par	Deployable, containers require deployment

Deployed Configuration

Name	Deployment Status
BW_ADB_FA	Success
ActiveDatabaseAdapterConfiguration.aar	Success
karlh - ActiveDatabaseAdapterConfiguration	Success
FileAdapterConfiguration.aar	Success
Process Archive.par	Success

The Configuration Builder pane on the left allows you to deploy or update applications and to revert a deployment, that is, choose an earlier deployment configuration if there was one. You can deploy the same application multiple times, for example, to try out different machine configurations. However, only one deployment configuration can be running at any time. If you later wish to return to a previous deployment configuration, you can do so by choosing **Revert**.

You can also view the deployment history, the current deployment if there is one, and completely undeploy the application.

For more information, see Undeploying a Deployed Application.

If you have installed new TIBCO software on a machine that is running process engines, you can upgrade them to use the new software by clicking **Upgrade**.

For more information, see Upgrading an Application .

When you select an application, service or service instance in the Configuration Builder panel, the displayed dialog allows you to change parameters for the deployment. When you select an application, service or service instance in the Deployed Configuration panel, the displayed dialog is read-only, providing a description of the properties.

Changing Application Global Variables and Repository Properties

When TIBCO Administrator deploys an application, it creates an application repository that contains information about the application configuration. You can view and change certain aspects of the application repository. For example, you can change the deployment repository instance to use the HTTP transport, if the Rendezvous transport was set in the archive file.

If your domain was configured to push the application repository to the local machines where the application is run, the default choice is local. If this choice is used, each local machine must have TRA™ 5.9.0 (or later) installed.

The defaults set in the enterprise archive file for the application name and for global variables can also be changed.

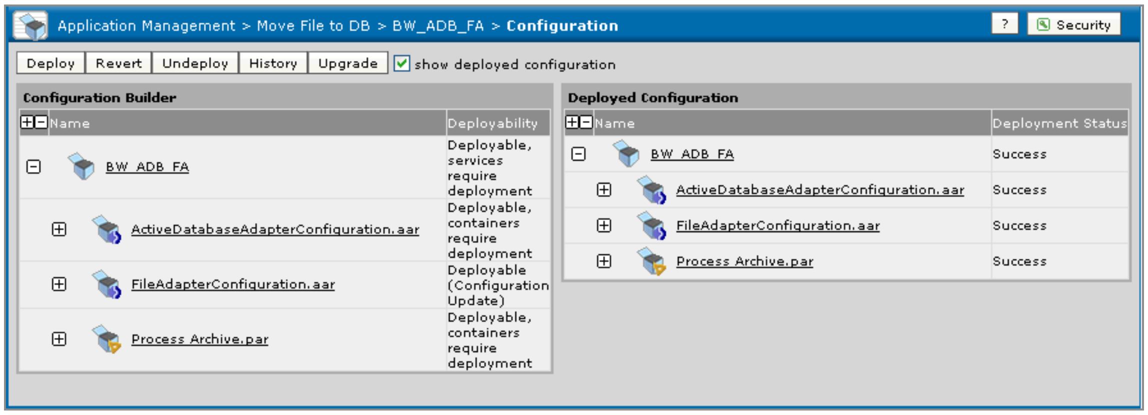
The **Reset to Defaults** button restores all properties to the values defined in the enterprise archive file.

To change application properties, perform the following procedure:

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Click *Application* **> Configuration**, where *Application* is the application created when you loaded the enterprise archive file. See Creating an Application.
3. In the Configuration Builder pane, select the *Application* name.
4. Click the **General** tab to change the application name, description or contact information.

5. Click the **Advanced** tab to change global variables or deployment repository instance properties. For more information about global variables, see Global Variables.
6. Click **Save**.
7. After you have made your changes, the Configuration Builder indicates that the deployment is out of date. For more information, see Edit Application Configuration Dialog .


Application Management > Move File to DB > BW_ADB_FA > Configuration
?
Security
Deploy
Revert
Undeploy
History
Upgrade
show deployed configuration

Configuration Builder

Name	Deployability
BW_ADB_FA	Deployable, services require deployment
ActiveDatabaseAdapterConfiguration.aar	Deployable, containers require deployment
FileAdapterConfiguration.aar	Deployable (Configuration Update)
Process Archive.par	Deployable, containers require deployment

Deployed Configuration

Name	Deployment Status
BW_ADB_FA	Success
ActiveDatabaseAdapterConfiguration.aar	Success
FileAdapterConfiguration.aar	Success
Process Archive.par	Success

8. Click **Deploy**.

Enabling a Process or Service to Run on Other Machines

You can assign a process to run on any machine that is part of your administration domain. For information about adding a machine to a domain, see *TIBCO Runtime Agent™ Domain Utility User’s Guide* .

Tip: Adding a process to additional machines is useful for fault tolerance. As a rule, it therefore does not make sense to run the same process on the same machine twice.

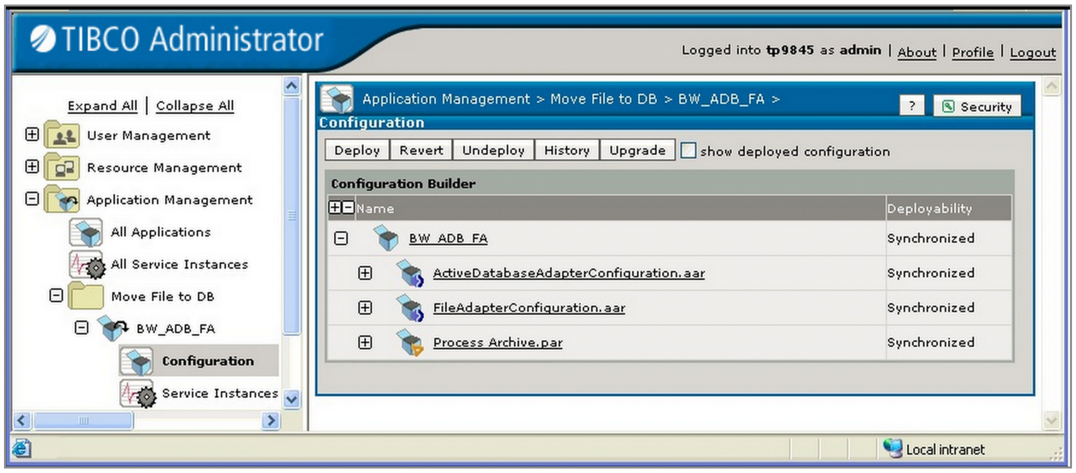
A service can be enabled or disabled. Only enabled services are deployed. When you disable a service, it is no longer deployed the next time you deploy the application, while all other services in the application are deployed as before. This can be useful, for example when you wish to deploy an application that includes a service for which you don't have the required software.

Only machines that have the software required by the process or service are visible when selecting the machine.

To enable a process to run on other machines, perform the following procedure:

Procedure

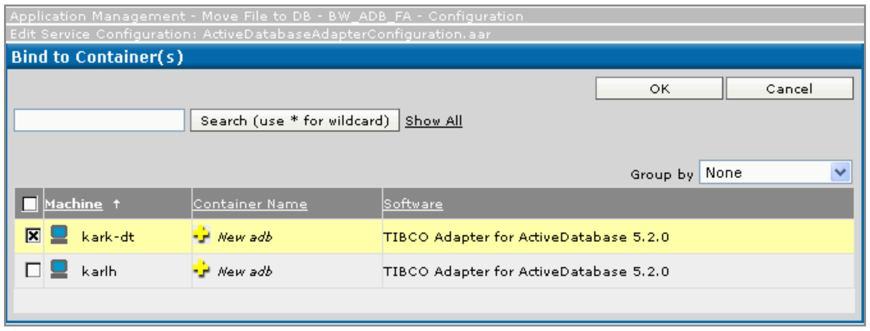
1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it.



Configuration Builder

Name	Deployability
BW_ADB_FA	Synchronized
ActiveDatabaseAdapterConfiguration.aar	Synchronized
FileAdapterConfiguration.aar	Synchronized
Process Archive.par	Synchronized

3. In the Configuration Builder pane, click a service or process name. A service is named with a `.arr` suffix. A process is named with a `.par` suffix.
4. In the General pane enable or disable the process or service by selecting or clearing the `Enable Service` check box.
5. In the Target Machines pane, click **Add to Additional Machines** to add a selected process or service to another machine.
6. A dialog appears, similar to the following, displaying all machines in the domain on which the software required by the process or service is available. Select a machine, and then click **OK**.



Application Management - Move File to DB - BW_ADB_FA - Configuration
Edit Service Configuration: ActiveDatabaseAdapterConfiguration.aar
Bind to Container(s)

	Machine ↑	Container Name	Software
☒	kark-dt	New adb	TIBCO Adapter for ActiveDatabase 5.2.0
☐	karlh	New adb	TIBCO Adapter for ActiveDatabase 5.2.0

7. Click **Save**.

See Also

See Edit Service Configuration Dialog for more information.

See Configuring Fault Tolerant Process Engines.

Adding a Rulebase to a Process or Service

The TIBCO Hawk agent monitors managed objects by processing rulebases, which are named collections of rules that contain management logic. Using TIBCO Hawk Display, you can create rulebases with specialized rules. (TIBCO Hawk Display is not included in TRA). Hawk allows you to specify a very large number of alert conditions and alert results. You must have purchased the full TIBCO Hawk product to create TIBCO Hawk rulebases.

The same rulebase can be loaded on a single service, or multiple services.

Multiple rules defined in the same rulebase can monitor a particular application or system function. For example, an application rulebase could include one rule for issuing a medium-level alert if disk space or CPU usage exceeds certain thresholds. Another rule could issue a high-level alert and send a pager message to the system administrator if the application process terminates.

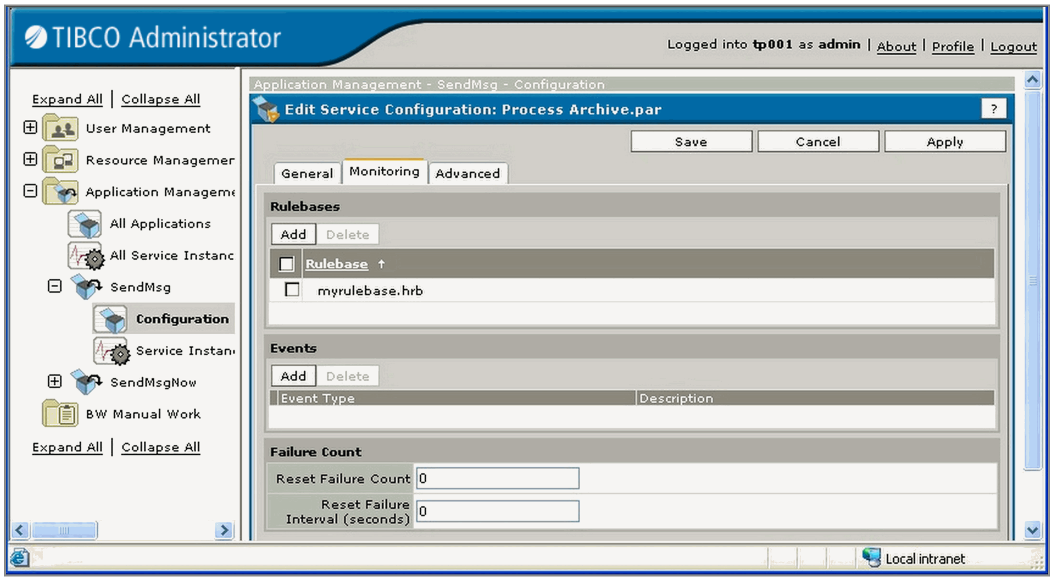
Adding a TIBCO Hawk Rulebase to an Application

This section provides information about adding a rulebase for a service or process. Information about building the rulebase expression is not provided. For more information about creating rulebases, see the TIBCO Hawk Administrator's Guide . The guide is part of the TIBCO Hawk documentation set.

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select the application for which the rulebase has been defined, and expand it.
3. In the `Configuration Builder` pane, click the service or process name for which the rulebase has been defined. A service is named with a `.arr` suffix. A process is named with a `.par` suffix.
4. Click the **Monitoring** tab.
5. In the Rulebases panel, click **Add**.
6. Click **Browse** and in the window that appears, navigate to the directory where the rulebase is stored and select the rulebase. Click **OK**.
7. Click **Save**.

For example, the next diagram shows the rulebase section for a process archive.



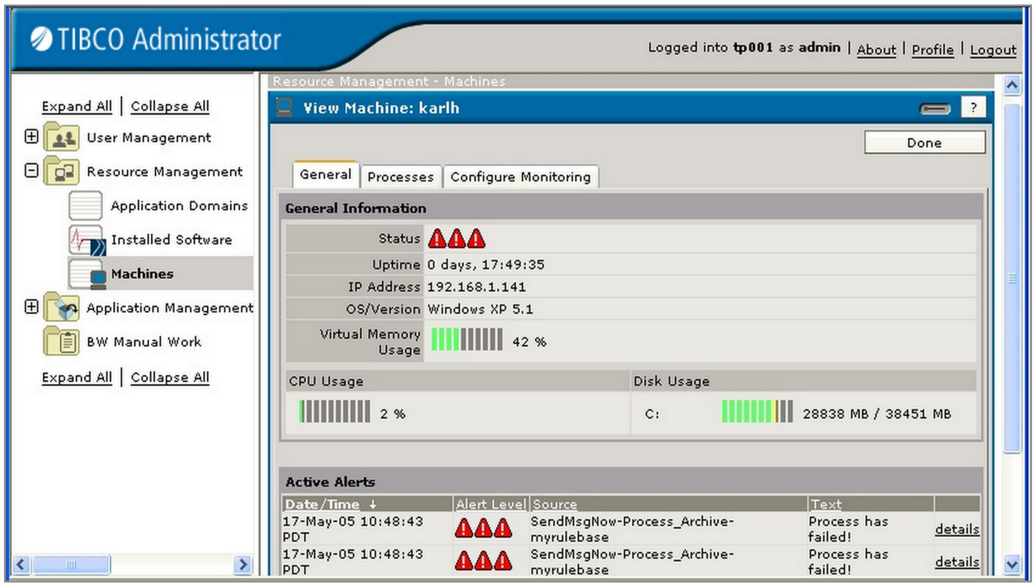
Rulebase ↑
myrulebase.hrb

Event Type	Description

Reset Failure Count	0
Reset Failure Interval (seconds)	0

When you deploy the service, TIBCO Hawk Agent saves the rulebase file in the *<install-path>*`\tibco\tra\domain\`*<domain-name>*`\rulebase` folder. The original rulebase can be safely removed, because the rulebase has been loaded into the domain. If you change the original rulebase, it must be reloaded into the service and the application must be redeployed.

When the conditions specified in the rulebase occur, the results display in the `Resource Management > Machines View Machine` panel. For example, the next screen shows several alerts that have been generated based on a rulebase.



General Information

Status	⚠⚠⚠
Uptime	0 days, 17:49:35
IP Address	192.168.1.141
OS/Version	Windows XP 5.1
Virtual Memory Usage	42 %

CPU Usage	Disk Usage	
2 %	C:	28838 MB / 38451 MB

Active Alerts

Date/Time ↓	Alert Level	Source	Text	
17-May-05 10:48:43 PDT	⚠⚠⚠	SendMsgNow-Process_Archive-myrulebase	Process has failed!	details
17-May-05 10:48:43 PDT	⚠⚠⚠	SendMsgNow-Process_Archive-myrulebase	Process has failed!	details

How to Create a Custom Rulebase

You can create rulebases using the TIBCO Hawk Display rulebase editor. The rulebase file name can be saved without using a naming convention (as was necessary in releases prior to 5.9.0). TIBCO Hawk Agent creates the appropriate rulebase name and file when the service instance to which the rulebase is assigned is deployed.

Note: The rulebase file name should not contain the space character.

For example, if two rulebase files are created and named:

- shared_custom1.hrb
- shared_custom2.hrb

And the above rulebase files are assigned to two service instances (as described in the previous section).

- D1-Process_Archive
- E1-Process_Archive

When the D1-Process_Archive service is deployed, TIBCO Hawk Agent creates the following rulebase files for the service.

- D1-Process_Archive-shared_custom1.hrb

- D1-Process_Archive-shared_custom2.hrb.

Similarly, when the D1-Process_Archive-1 service is deployed, TIBCO Hawk Agent creates the following rulebase files for the service:

- E1-Process_Archive-shared_custom1.hrb
- E1-Process_Archive-shared_custom2.hrb.

The rulebase file names for each instance are stored in an external property file so the TIBCO Hawk Agent knows where to re-load the rulebase files if it is restarted.

- Rulebase file names are stored in the *<install-path>*`\tibco\tra\domain\`*<application-name>*`\startup\`*<application-name>*`.properties` file in the `rbList` property.
- TIBCO Hawk Agent assumes that the input rulebase file name provided when uploading a rulebase file uses the `.hrb` extension. If there is no file extension, Hawk Agent appends `.hrb` to the rulebase file during deployment.

Variable Substitution

You can assign certain variables to a rulebase and TIBCO Hawk Agent will substitute values for the variables when the application is deployed. Variable substitution is typically used in a rulebase to change the data source from pointing to a specific service instance to point a generic service instance.

The following variables are supported by TIBCO Hawk Agent:

- %%TIBCO_DEPLOYMENT%%— When encountered, the Hawk Agent substitutes the application’s deployment name.
- %%TIBCO_COMPONENT_INSTANCE%% — When encountered, Hawk Agent substitutes the service instance name.
- %%TIBCO_DOMAIN%%— When encountered, Hawk Agent substitutes the administration domain name.
- %%TIBCO_COMPONENT_TYPE%%— When encountered, Hawk Agent substitutes the component’s type.

Adding an Event to a Service

You can define an event type to respond to a service instance failure, or to be triggered when a match occurs for some condition that is reported in the service instance log file.

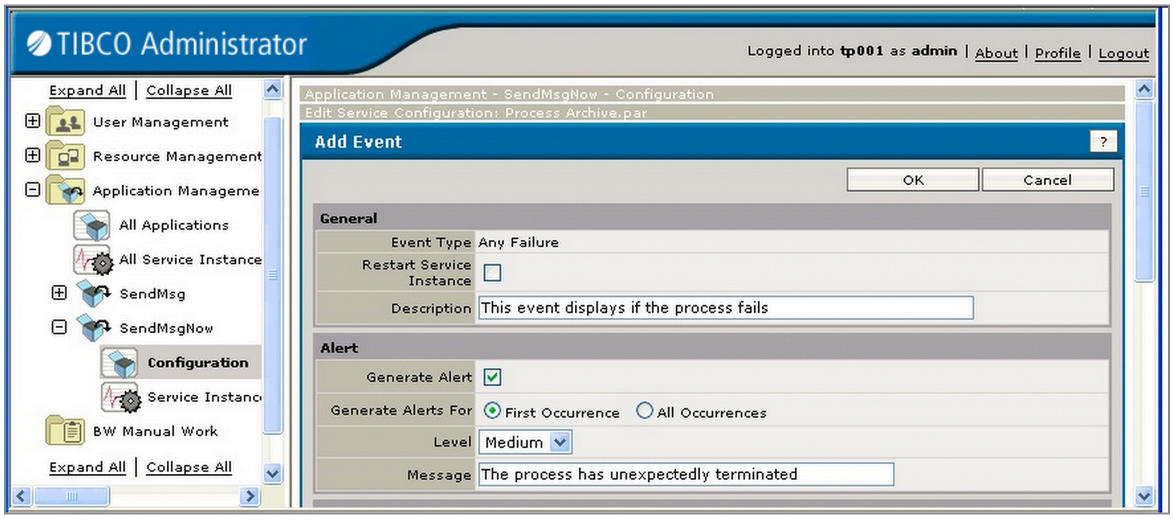
Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it.
3. In the `Configuration Builder` pane, click a service or process name. A service is named with a `.arr` suffix. A process is named with a `.par` suffix.
4. Click the **Monitoring** tab.
5. Click **Add** in the `Events` panel.
6. Specify the conditions and the event.

   First choose a condition in the `General` panel.

   In case the condition is met, you can choose to send an alert, send an email, or execute a command, or a combination of those.
7. When you've configured both condition and event, click **OK**.
8. Click **Save**.

For example, the next diagram shows the Add Event panel for a process archive.


TIBCO Administrator
Logged into tp001 as admin | About | Profile | Logout
Expand All | Collapse All
User Management
Resource Management
Application Manageme
All Applications
All Service Instance
SendMsg
SendMsgNow
Configuration
Service Instanc
BW Manual Work
Expand All | Collapse All
Application Management - SendMsgNow - Configuration
Edit Service Configuration: Process Archive.par
Add Event
?
OK
Cancel

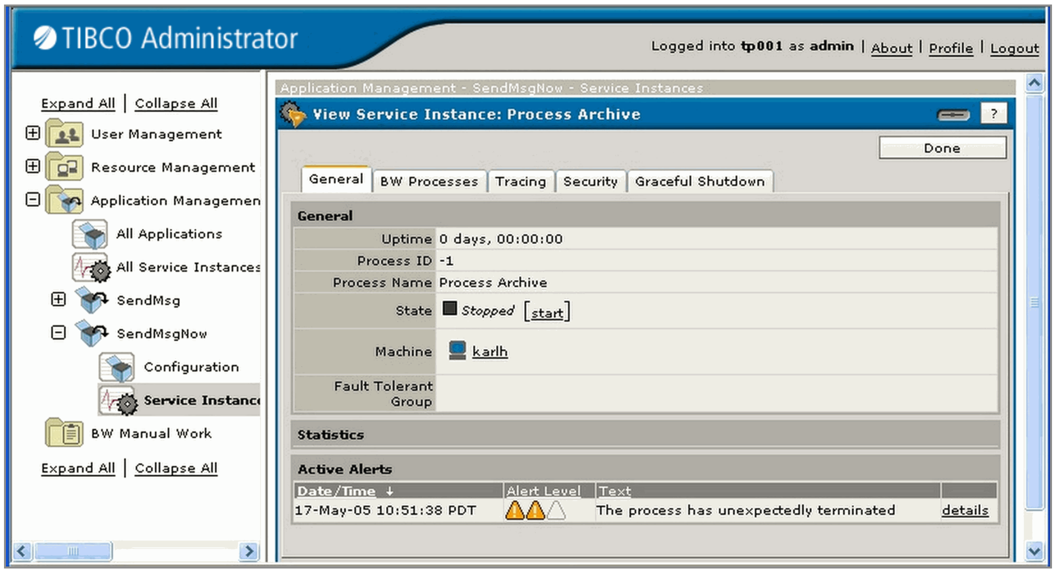
General

Event Type	Any Failure
Restart Service Instance	☐
Description	This event displays if the process fails

Alert

Generate Alert	✓
Generate Alerts For	[✓ First Occurrence] All Occurrences
Level	Medium
Message	The process has unexpectedly terminated

The event can be sent as an alert, by email or can trigger an operating system command. If the event is sent as an alert, it appears in the `View Service Instance` dialog under the `Active Alerts` pane. For example:



General

Uptime	0 days, 00:00:00
Process ID	-1
Process Name	Process Archive
State	■ *Stopped* [start]
Machine	karlh
Fault Tolerant Group	

Active Alerts

Date/Time ↓	Alert Level	Text	
17-May-05 10:51:38 PDT	⚠⚠△	The process has unexpectedly terminated	details

For an example event configuration, see Specifying a Custom Alert

Configuring Storage for ActiveMatrix BusinessWorks Processes

You can use TIBCO Administrator to configure the location where ActiveMatrix BusinessWorks process engines store internal information.

Most of the information a process engine stores is information about each service’s state when a checkpoint is taken. There is, however, some other internal information stored by the engine. You can specify that this information is stored in the file system (the default) or in a database.

For some systems, using a file system for storage may be sufficient. However, some functionality is only available when you use a database for storing information about service state:

- When configured properly, shared variables can be used to pass information across multiple process engines when a database is used for storage.
- Duplicate detection of messages across multiple engines after a recovery from a checkpoint requires a database for process engine state storage.
- Using critical section groups across multiple engines requires a database for storage.

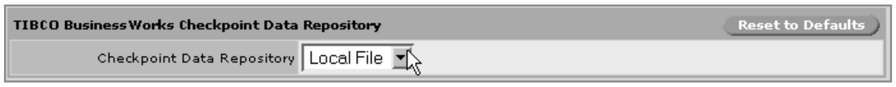
- With a database for storage, Wait/Notify activities can be used to pass data between services running on different machines.

Specifying a Database for Storage

To configure a database for storage, follow these steps:

Procedure

1. In TIBCO Designer, make sure to specify a `JDBC Connection` resource for the database you wish to use, and then build the EAR file.
2. After you have uploaded the EAR file and created the application in the TIBCO Administrator GUI, select `Application Management` then select the application in the `Configuration Builder` pane of the `Configuration` console.
3. Select the service (.par) in the `Configuration Builder` pane and choose the **Advanced** tab.



TIBCO BusinessWorks Checkpoint Data Repository

Checkpoint Data Repository: Local File

4. In the `TIBCO ActiveMatrix BusinessWorks Checkpoint Data Repository` pane, select the `Database` pane.

Database Table Names

When you specify a database for ActiveMatrix BusinessWorks storage, tables are created in your database. The administration domain name and deployment ID (assigned by ActiveMatrix BusinessWorks) are used to name the tables to ensure uniqueness of the tables for each domain and each deployment.

Because some databases limit the number and kinds of characters for table names, the domain name can altered before being used in the table name. The first eight characters and the last eight characters of the domain name are taken and any non-legal characters (such as spaces or dashes) are converted to underscores. This creates a sixteen-character unique ID for each domain, provided that the first and last eight characters of all of your domain names are unique.

For example, the following illustrates conversion of domain names. Notice the second and third domain names convert to the same ID. You should avoid this by creating domain names so that the combination of the first and last eight characters are unique.

Domain Name	Converts To Domain ID
TIBCO_domain_Accounting	TIBCO_docounting
TIBCO_domain_Marketing	TIBCO_doarketing
TIBCO_domain_Direct_Marketing	TIBCO_doarketing

All table names created by ActiveMatrix BusinessWorks begin with `T_`*<domain-id>*`_`*<deploymentID>*`_`. You can alter the storage parameters for these tables if you desire, but the table names and column definitions must remain the same.

Manually Creating Database Tables

The process engine creates database tables used to store process engine information automatically. Some database administrators do not permit applications to automatically create tables. If you wish to manually create the database tables, ActiveMatrix BusinessWorks provides template scripts for the supported databases in the `<TIBCO_BW_HOME>/bin` directory.

In these scripts `<TABLE_NAME_PREFIX>` and `<ENGINE_NAME_MAX_LENGTH>` are placed in the SQL code as placeholders. The `<TABLE_NAME_PREFIX>` is determined by default at deployment time.

For a description about how the table name prefix is determined, see Database Table Names

You can obtain this prefix by locating the `Database.Tablename.Prefix` property in the deployment configuration file and substituting its value where required in the template SQL script. `<ENGINE_NAME_MAX_LENGTH>` is 128, so supply that value instead of the placeholder in the SQL script.

To manually create the tables, perform the following procedure:

Procedure

1. Create a deployment configuration that specifies a database for process engine storage. See Specifying a Database for Storage.
2. Before starting the process engine, examine the deployment configuration file (the <*processEngine*>`.tra` file) and locate the property `Database.Tablename.Prefix`.
3. Edit the appropriate SQL script template for the database you are using and replace `<TABLE_NAME_PREFIX>` with the value of the `Database.Tablename.Prefix` property.
4. Change `<ENGINE_NAME_MAX_LENGTH>` to 128.
5. Save the changes to the SQL script.
6. Run the SQL script against the database you wish to use.
7. Start the process engine.

Controlling Execution of ActiveMatrix BusinessWorks Services

Process starters create process instances to handle incoming events. Process instances consume memory and CPU resources on your system. Depending on the available machine resources, you may only be able to run a limited number of process instances concurrently.

Process instances typically remain in memory as long as they are executing an activity. If the process instance is waiting for an incoming event (for example, a Wait for Adapter Message activity), the process instance can be paged out to disk and resumed later after the event arrives. New process instances are paged out to disk until there is available memory and resources to accommodate them.

You can use TIBCO Administrator to control the execution of ActiveMatrix BusinessWorks process instances. This is useful if your system has limited memory or resources, or if you want to restrict process instances to run sequentially.

The TIBCO ActiveMatrix BusinessWorks Process Configurations dialog allows you to specify the following:

- Max Jobs — Specifies the maximum number of process instances that can concurrently be loaded into memory.
- Use Activation Limit — Specifies that once a process instance is loaded, it must remain in memory until it completes.

- Flow Limit — Specifies the maximum number of currently running process instance to start before suspending the process starter.

To change process configuration properties, perform the following procedure:

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it.
3. Click Configuration.
4. In the Configuration Builder pane, click a process name. A process is named with a `.par` suffix.
5. Click the **Advanced** tab.
6. Change properties as required. The remaining topics in this section provide information about the properties you can set.
7. Click **Save**.

Specifying the Maximum Number of Concurrently Active Processes

Incoming events may not be evenly distributed over time. That is, there may be periods where a large number of incoming events occur and other periods where relatively few events occur. To prevent your system from being overwhelmed by incoming events, the **Flow Limit** field limits the number of process instances created by a process starter. This allows you to control the flow of processing so that incoming events are no longer accepted when the limit is reached.

Controlling the flow of processing is especially useful when you are using protocols that can store unsent messages on the server until the receiver is ready to process them. For example, if your process definition polls an email server for new messages (that is, Receive Mail is the process starter), and then you can set Flow Limit to control the number of process instances created for each new email. Email that has not been processed remains on the email server until the process engine is ready to create more process instances. Other protocols where this approach are useful are TIBCO Rendezvous Certified Messaging (RVCM), JMS durable topic subscriptions, and JMS queues.

When a process engine reaches the specified Flow Limit, it is placed in a FLOW_ CONTROLLED state. In this state, the process engine can continue executing existing process instances, but new process instances are not allowed. Incoming messages can then be directed to another process engine. A process engine will resume creating new process instances once a sufficient number of its current process instances have completed. Typically a process engine comes out of the FLOW_CONTROLLED state when the number of process instances completed is approximately half of the value specified for the Flow Limit property.

The HTTP Receiver process starter uses a different mechanism for controlling the flow of incoming requests. When Flow Limit is set on a process definition containing this process starter, the maximum number of incoming requests is limited to `<`*flowLimitValue*`> -1`. It is recommended that you use the minProcessors and maxProcessors properties to control the flow of incoming HTTP requests instead of using the Flow Limit property.

For more information about flow control of the HTTP Receiver process starter, see the description of the HTTP Connection resource in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

The **Max Jobs** field in the Process Configurations dialog allows you to specify the maximum number of concurrent process instances that can be stored in memory. For example, if you set Max Jobs to 5, the process engine can only keep five process instances in memory. Any process instances created once the maximum is reached must be paged out to disk.

Specifying a value for `Max Jobs` causes the process engine to incur some overhead for managing the paging of process instances to and from disk. If you have sufficient system resources and do not expect incoming events to exceed the limits of your system, consider specifying `Max Jobs` as 0. This allows the process engine to create an unbounded number of process instances and eliminates the overhead of paging.

Keeping Services in Memory

The Use Activation Limit field specifies that once a process instance is loaded into memory, it should not be paged out to disk until it completes. This option is useful if you wish to specify sequential processing of incoming events, or if you want to enforce limited concurrent execution of process instances.

Effects of Setting the Configuration Fields

The Max Jobs and Use Activation Limit options work together to provide different concurrency limits. The Flow Limit field also affects the concurrency limit. The next table describes the effects of various combinations of these options.

Effects of various configuration settings

Max Jobs	Use Activation Limit	Flow Limit	Description
0	Cleared or selected	0	An unlimited number of process instances can be created and concurrently loaded into memory. Use Activation Limit is ignored when Max Jobs is set to 0.
0	Cleared or selected	N	No paging of process instances. Allows up to N process instances before placing process starter in flow controlled stated. Use Activation Limit is ignored when Max Jobs is set to 0.
1	Selected	N	One process instance is loaded into memory at a time and kept there until it completes its execution. This guarantees incoming events are processed in the order in which they occur. Up to N process instances are paged to disk, and then the process starter is placed into flow controlled state. **Note:** If your goal is to sequentially process incoming events, use the Sequencing Key field on the Misc tab of the process starter. Using Max Jobs and Use Activation Limit incurs overhead as process instances are paged to disk and retrieved from disk.
1	Selected	0	Once process instance is loaded into memory at a time and kept there until it completes its execution. This guarantees incoming events are processed in the order in which they occur. There is no limit on the number of

Max Jobs	Use Activation Limit	Flow Limit	Description
			process instances that can be created and paged to disk. **Note:** If your goal is to sequentially process incoming events, use the Sequencing Key field on the Misc tab of the process starter. Using Max Jobs and Use Activation Limit incurs overhead as process instances are paged to disk and retrieved from disk.
1	Cleared	N	One process instance is loaded into memory at a time, but up to N process instances are created. Incoming events can be processed in any order because process instances are not kept in memory until they complete execution.
M	Selected	0	An unlimited number of process instances can be created, but only M are loaded into memory and processed concurrently. This setting ensures a limited amount of concurrent processing. This situation is useful if you have limited resources, such as database connections. You can set Max Jobs to a relatively small number and the Use Activation Limit option keeps each service in memory until the service completes. Each loaded process uses a machine resource until the service completes. Once a service releases the resource, a new process can be loaded into memory and the corresponding service can use the resource.
		N	Same as above, except only N process instances are created before the process engine is placed in the flow controlled state.
M	Cleared	0	An unlimited number of process instances can be created, but only M are loaded into memory and processed concurrently. After M process instances are

Max Jobs	Use Activation Limit	Flow Limit	Description
			created, new process instances are paged to disk. There is no guarantee of the order in which process instances are executed.
		N	Same as above, except only N process instances are created before the process engine is placed in the flow controlled state.

Changing Server Settings

You can change the following properties for a process engine. You can also modify Java properties, such as changing the classpath and managing the heap size. In addition, you can set whether the instance should run as a Windows Service and define startup options.

- Start on Boot
- Enable Verbose Tracing
- Max Log File Size
- Max Log File Count
- Thread Count

To change server settings for a process engine, perform the following procedure:

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it to view process engines.
3. Click a process engine or service instance.
4. Click the **Server Settings** tab. For more information about Server Settings tab, see Server Settings Tab.
5. Change options as required.

6. Click **Save**.

Setting Graceful Shutdown Properties for a Process Engine

The graceful shutdown command causes the process engine to deactivate all process starters and wait (up to the maximum timeout) for all current jobs to either finish or take a checkpoint, before shutting down the engine. If Wait For Checkpoints is selected, the engine will wait up to the Kill Jobs Timeout for all jobs to finish even if they take a checkpoint.

You can only set the graceful shutdown properties on an undeployed process engine.

To set graceful shutdown properties, perform the following procedure:

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it to view a process engine.
3. Click a process engine.
4. Click the **Graceful Shutdown** tab. For more information, see Graceful Shutdown Tab.
5. Change options as required.
6. Click **Save**.

Configuring Fault Tolerant Process Engines

The ActiveMatrix BusinessWorks process engine can be configured to be fault-tolerant. You can start up several engines. In the event of a failure, other engines restart process starters and the corresponding services.

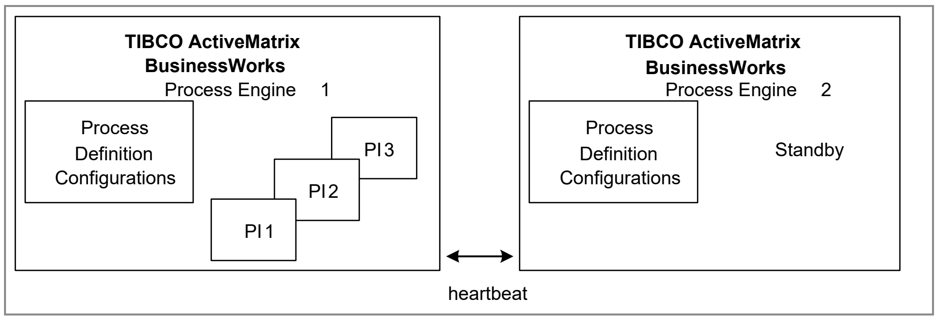
If you use a database to store process engine information, a process instance is re-instantiated to the state of its last checkpoint. In the event of a failure, any processing done after a checkpoint is lost when the process instance is restarted by another engine.

For more information about Checkpoint activities, see *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

For more information about configuring process engine storage, see Changing the Checkpoint Data Repository for a Process.

Figure 2 illustrates normal operation of a fault-tolerant configuration. One engine is configured as the master, and it creates and executes services. The second engine is a secondary engine, and it stands by in case of failure of the master. The engines send heartbeats to notify each other they are operating normally.

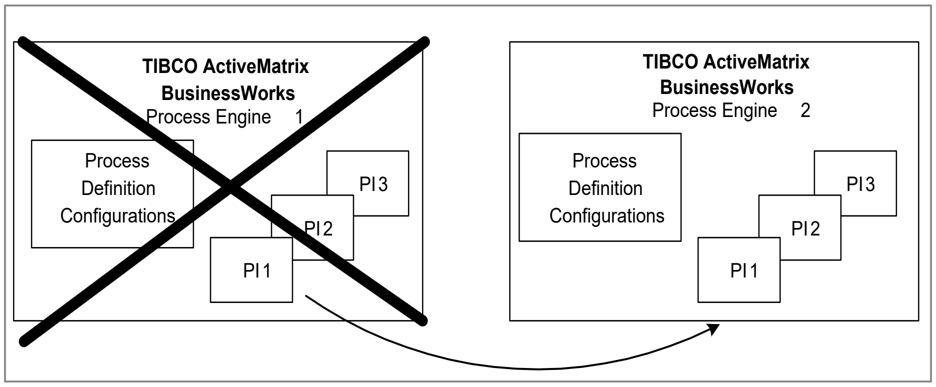
Figure 4: Normal operation: master processing while secondary stands by


TIBCO ActiveMatrix BusinessWorks
Process Engine 1
Process Definition Configurations
PI 3
PI 2
PI 1
TIBCO ActiveMatrix BusinessWorks
Process Engine 2
Process Definition Configurations
Standby
heartbeat

In the event the master process engine fails, the secondary engine detects the stop in the master’s heartbeat and resumes operation in place of the master. All process starters are restarted on the secondary, and services are restarted to the state of their last checkpoint.

Figure 3 illustrates a failure and the secondary restarting the service.

Figure 5: Fault-tolerant failover


TIBCO ActiveMatrix BusinessWorks
Process Engine 1
Process Definition Configurations
PI 3
PI 2
PI 1
TIBCO ActiveMatrix BusinessWorks
Process Engine 2
Process Definition Configurations
PI 3
PI 2
PI 1

The expected deployment is for master and secondary engines to reside on separate machines. You can have multiple secondary engines, if desired, and you can specify a weight for each engine. The weight determines the type of relationship between the fault-tolerant engines. For more information about relationships between fault-tolerant engines, see Peer or Master and Secondary Relationships.

A master and its secondary engines is known as a *fault-tolerant group*. The group can be configured with several advanced configuration options, such as the heartbeat interval and the weight of each group member. For a complete description of configuration options for fault tolerance, see *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

Peer or Master and Secondary Relationships

Members of a fault-tolerant group can be configured as peers or as master and secondary engines. If all engines are peers, when the machine containing the currently active process engine fails, another peer process engine resumes processing for the first engine, and continues processing until its machine fails.

If the engines are configured as master and secondary, the secondary engine resumes processing when the master fails. The secondary engine continues processing until the master recovers. Once the master recovers, the secondary engine stands by and the master takes over processing again.

The **Fault Tolerance** tab of the Process Engine deployment resource allows you to specify the member weight of each member of a fault-tolerant group. The member with the highest weight is the master. You can select "Peer" in the first field on the tab to configure all engines as peers (that is, they all have the same weight). You can select Primary or Secondary to configure the engines as master and secondary. You can also select Custom to specify your own values for the weight of each member of the group.

Process Starters and Fault-Tolerance

When a master process engine fails, its process starters are restarted on the secondary engine. This may not be possible with all process starters. For example, the HTTP Receiver process starter listens for HTTP requests on a specified port on the machine where the process engine resides. If a secondary engine resumes operation for a master engine, the new machine is now listening for HTTP requests on the specified port. HTTP requests always specify the machine name, so incoming HTTP requests will not automatically be redirected to the new machine.

Each process starter has different configuration requirements, and not all process starters may gracefully resume on a different machine. You may have to provide additional hardware or software to redirect the incoming events to the appropriate place in the event of a failure.

Also, your servers may not have all of the necessary software for restarting all of instances. For example, your database may reside on the same machine as your master process engine. If that server goes down, any JDBC activities will not be able to execute. Therefore, you may not wish to load process definitions that use JDBC activities in your secondary process engine.

You can specify that your secondary process engine loads different process definitions than the master. You may only want to load the process definitions that can gracefully migrate to a new server during a failure.

Setting Fault Tolerant Options

The FT Group Settings panel displays only if the ActiveMatrix BusinessWorks process you have selected has been added to at least two (different) machines. If your domain includes components that were deployed as part of a fault-tolerant group, the display includes the information about the group.

You can start one or more process engines in the group. If more than one engine has started, only one is displayed as `Running` and all other engines are displayed as `Standing By` (or, initially, as `Starting Up`).

When you change the status of a component that has been deployed as part of a FT group, the status change affects all other members of the group.

- After you have deployed the process engines, it is most efficient to select all process engines by clicking the check boxes, and then choosing **Start**. After the primary and secondary engines have communicated, the master will display as `Running` and all other engines as `Standby`. If you start only the primary, it will first go to `Standby` mode as it checks the status of the other engines. It then changes to `Running`.
- If you shutdown a process engine, the appropriate secondary engine starts automatically.

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it.
3. In the `Configuration Builder` pane, click process name. A process is named with a `.par` suffix.

4. Click the **General** tab.
5. Select **Run Fault Tolerant**. Change other options as required. For field descriptions, see FT Group Settings .
6. Click **Save**.

Changing the Checkpoint Data Repository for a Process

A checkpoint saves the current state of a running process instance. For a secondary process engine to resume running process instances from their last checkpoint, the secondary process engine must have access to the saved state of the process instances from the master process engine.

Features that allow communication across process engines (for example, wait/notify, critical sections, shared variables, and so on) require a database for storage of process engine state.

If you are running process engines that do not communicate with each other, then the file system can be used for process engine storage. In this case, if you configure primary and secondary engines for fault tolerance, all engines must point to the same shared location within the file system.

The remainder of this section describes using a database for process engine storage.

Because fault-tolerant engines are expected to be on separate machines, you should specify to use a database for storage for each process engine. This allows you to specify the same JDBC Connection resource for the master and secondary engines, and therefore all engines can share the information stored for process instance checkpoints.

If all engines share the checkpoint information, and then the secondary engines can recover process instances up to their last checkpoint. If engines do not share the checkpoint information, process instances are not restarted.

To change checkpoint data repository properties, perform the following procedure:

Procedure

1. In TIBCO Administrator, click **Application Management**.

2. Select an application and expand it.
3. In the `Configuration Builder` pane, click a process name. A process is named with a `.par` suffix.
4. Click the **Advanced** tab.
5. Change properties as required. The value defaults to **Checkpoint Data Repository**. If the **JDBC Connection Resource** has been configured for the project, you also have the option to choose database.
6. Click **Save**.

Changing Runtime Variables for a Process or Service

Some service and process specific runtime variables can be defined in TIBCO Designer and changed in TIBCO Administrator. When defining runtime variables in TIBCO Designer, you specify whether the variable should be settable at design-time only, for the deployment, or for the service. All variables that were designated settable for the service are then displayed in TIBCO Administrator.

To reset to the default values defined in the enterprise archive file, click **Reset to Defaults**.

To change runtime variables for a service or process, perform the following procedure:

Procedure

1. In TIBCO Administrator, click **Application Management**.
2. Select an application and expand it.
3. In the `Configuration Builder` pane, click a service or process name. A service is named with a `.arr` suffix. A process is named with a `.par` suffix.
4. Click the **Advanced** tab.
5. Change runtime variables as required.
6. Click **Save**.

Application Management Configuration Dialog

The application management configuration dialog displays the following panes, side by side, if `Show deployed configuration` is selected.

- Configuration Builder Pane
- Deployed Configuration Pane

Configuration Builder Pane

Deploy

Click to deploy the application. The deploy dialog appears.

Revert

When you revert an application, you select a different configuration of the currently deployed configuration. You can then decide to deploy this deployment configuration. If you do, service instances are stopped, updated, and restarted. Any component that is removed from a machine as a result of the revert is undeployed from that machine. For more information, see Reverting to a Previously Deployed Application .

Undeploy

Click to undeploy the application. When you undeploy a deployed application, TIBCO Administrator stops all running services and removes them from the list of services that can be started. In effect, it completely removes all traces of the deployment (with the exception of the logs). For more information, see Undeploying a Deployed Application .

History

Click to view the deployment history for this application. For more information, see Viewing Application Deployment History.

Upgrade

If you have installed new TIBCO software on a machine that is running process or service instances, you can upgrade the instances to use the new software by clicking **Upgrade**. For more information, see Upgrading an Application .

Show deployed configuration

Select to display the Deployed Configuration dialog box where you can view detailed information about the components deployed in the application.

Configuration List

Each component and service in the application is listed along with one of the following descriptors in the `Deployability` column

- Deployable, (Remove) — On Component. The last uploaded enterprise archive file does not contain this component. The component and all service instances will be removed from the application on deploy.

  On Service Instance — The service instance has been deleted. This will take effect on deployment.
- Deployable, (New) — The component or service instance has never been deployed successfully. If all service instances are removed and new ones added, the component will be in this state.
- Deployable (Archive Update) — The last uploaded enterprise archive file has changes related to this component. Changes will take effect on deployment.
- Deployable (Configuration Update) — The last uploaded enterprise archive file had deployment descriptors updated (typically global variables) that effect this component.
- Deployable (Configuration Changes) — Changes have been made to the service instance configuration and will take effect on deployment.
- Deployable (Last Deploy Failed) — The last deployment failed. History should have details. Likely problems are the TIBCO Hawk agent needs to be started on the target machine, or TIBCO Rendezvous communication or configuration parameters are not correct.
- Synchronized — The configuration is correct. There have been no changes since last successful deployment.

- Needs configuration — You must select a component or service instance and then each tab. Workflow in particular requires this for some automatic configuration to be done. Must be remedied or the component must be disabled before deployment can succeed.
- Need to bind to a Service — Not currently used.
- Deployable, services require deployment — The undeploy command was run. All services are configured correctly and are ready for deployment.
- Deployable, containers require deployment — The component had a service instance modified, added or removed. The change will take effect on deployment.
- Services require configuration — A component has a service instance that needs to be configured. Deployment can not be done until this is remedied or the component is disabled.
- Containers require configuration — Not currently used.
- Disabled — The component is marked disabled and will not be deployed. If deployment is attempted, the component will be undeployed when deployment is done.
- Disabled, will remove existing configuration — The component for the deployed service instance was marked Disabled. When deployment is done, the service instance will be undeployed.

Deployed Configuration Pane

Displays deployed components for this application and their status. Click each component to view detailed information about the deployed component.

Edit Application Configuration Dialog

Fields can be edited if this dialog is invoked from the Configuration Builder pane. If invoked from the Deployed Configuration pane, the fields are read only.

The following tabs are available:

- General Tab
- Advanced Tab

General Tab

Application Archive

Provides information about the enterprise archive file including the package name, version, description, creation date and owner.

Upload New EAR File

Allows you to replace the current enterprise archive file with an updated version.

Application Parameters

Provides information about the application name, associated deployment name, description and contact name for the application.

Advanced Tab

The **Reset to Defaults** button resets all global variables to default settings as set in the enterprise archive file.

Global Variables

Displays the global variables set in the enterprise archive file for this application. The following global variables are predefined by default:

- DirLedger — Used by the system when defining the path name of the TIBCO Rendezvous certified messaging ledger file. The default is the root installation directory.
- DirTrace — Used by the system to partially create the path name for log file used by the adapter. The default is the root installation directory.
- HawkEnabled — Used by the system to indicate whether TIBCO Hawk is used to monitor the adapter. True indicates that a Hawk microagent is defined for the adapter. False indicates the microagent is not to be used. Default is False.

- JmsProviderUrl — A JMS provider URL tells applications where the JMS daemon is located. Setting this value mostly makes sense in early stages of a project, when only one JMS daemon is used.
- JmsSslProviderUrl — Specifies where the JMS server, running in the SSL mode, is located. Setting this value mostly makes sense in the early stages of a project, when only one JMS server is used.
- RemoteRvDaemon — Used by the system to identify the TIBCO Rendezvous routing daemon. See *TIBCO Rendezvous Administration* for details about specifying the routing daemon name.
- RvDaemon — Used by the system to identify the TIBCO Rendezvous daemon parameter. The parameter instructs the transport object about how and where to find the Rendezvous daemon and establish communication. The default value is `7500`, which is the default value used by the Rendezvous daemon. See *TIBCO Rendezvous Concepts* for details about specifying the daemon parameter.
- RvNetwork — Used by the system to identify the TIBCO Rendezvous network parameter. Every network transport communicates with other transports over a single network interface. On computers with more than one network interface, the network parameter instructs the TIBCO Rendezvous daemon to use a particular network for all outbound messages from this transport. See *TIBCO Rendezvous Concepts* for details about specifying the network parameter.
- RvService — Used by the system to identify the TIBCO Rendezvous service parameter. The Rendezvous daemon divides the network into logical partitions. Each transport communicates on a single service; a transport can communicate only with other transports on the same service. See *TIBCO Rendezvous Concepts* for details about specifying the service parameter. Default is 7500
- RvaHost — Used by the system to identify the computer on which the TIBCO Rendezvous agent runs. See *TIBCO Rendezvous Administration* for details about specifying the rva parameters.
- RvaPort — Used by the system to identify the TIBCO Rendezvous agent TCP port where the agent listens for client connection requests. See *TIBCO Rendezvous Administration* for details about specifying the rva parameters. Default is to 7501.
- TIBHawkDaemon — Used by the system to identify the TIBCO Hawk daemon parameter. See the *TIBCO Hawk Installation and Configuration* manual for details about this parameter. Default is the value that was set during domain creation (7474 by default).

- TIBHawkNetwork — Used by the system to identify the TIBCO Hawk network parameter. See the *TIBCO Hawk Installation and Configuration* manual for details about this parameter. Default is an empty string.
- TIBHawkService — Used by the system to identify the TIBCO service parameter. See the *TIBCO Hawk Installation and Configuration* manual for details about this parameter. Default is 7474.
- MessageEncoding — The message encoding set for the application. The default value is ISO8859-1, which only supports English and other western European languages that belong to ISO Latin-1 character set. After the project is deployed in an administration domain, the messaging encoding set at design time is overridden by the domain's encoding property. All the TIBCO components working in the same domain must always use the same encoding for intercommunication. See *TIBCO Administrator Server Configuration Guid*e for more information.

ActiveMatrix BusinessWorks Deployment Repository Instance

When TIBCO Administrator deploys an application, it creates an application repository which contains information about the application configuration. You can view and change certain aspects of the application repository.

In Transport you select the transport the administration server uses to communicate with the client application. Choose local, rv (TIBCO Rendezvous) or HTTP, or HTTPS if the administration domain has been set up to use HTTPS.

- local. By default, the transport is set to local. This means that the application repository will be sent to the target machine. This allows the application to run independently of the administration server.

  If you change the transport from local to another value, the application repository will not be pushed to the target machine, and the application will communicate with the administration server at runtime.

  For more information about these choices, see *TIBCO Administrator Server Configuration Guide*.

  The local choice is supported only if the target machines have installed TIBCO Runtime Agent 5.9.0 or later.
- rv. If selected, the client application will use TIBCO Rendezvous to communicate with the administration server. The following fields become available:
  - Server Name — administration server name.

  - Instance Name — Service instance name, that is, the instance of the service running on a particular machine.
  - User Name — User authorized for this application repository. Defaults to the user currently logged into Administrator.
  - Password — User’s password.
  - Timeout — Amount of time in seconds allowed for completing a task, such as retrieving information from the server. Defaults to 600 seconds.
  - Service, Network, Daemon — TIBCO Rendezvous connection parameters used.
  - Discovery Timeout — Amount of time in seconds allowed for the initial connection to the administration server.
  - Regional Subject — TIBCO Rendezvous subject prefix used for regional read-operation in the load balancing mode. For additional information see *TIBCO Administrator Server Configuration Guide*.
  - Operation Retry — Number of times to retry after a timeout occurs.

- http. If selected, the client application will use HTTP to communicate with the administration server.

  If your administration domain is not initially enabled for HTTPS, and there are deployed applications in the domain that use HTTP to connect to the application repository, the service instances will not restart after they are shut down. In this case, you must redeploy each service instance after changing the transport to HTTPS.

  - Server Name — administration server name.
  - Instance Name — Service instance name, that is, the instance of the service running on a particular machine.
  - User Name — User authorized for this application repository. Defaults to the user currently logged into Administrator.
  - Password — User’s password.
  - Timeout — Amount of time in seconds allowed for completing a task, such as retrieving information from the server. Defaults to 600 seconds.
  - HTTP URL, HTTPS URL — The URL on which the client attempts to connect to the server. What displays depends on whether you configured the server for HTTPS.

  Note: You cannot use HTTP or HTTPS to connect to a 4.x adapter.

Preview URL

If you have selected, `rv` or `http` in the Transport field, click the preview URL to display the URL that the application uses to access the application repository.

Edit Service Configuration Dialog

Fields can be edited if this dialog is invoked from the Configuration Builder pane. If invoked from the Deployed Configuration pane, the fields are read only.

The following tabs are available:

- General Tab
- Monitoring Tab
- Advanced Tab

General Tab

General

- Name — Service name.
- Description — Service description.
- Additional Required Components — Any other components required to run this service. You cannot enable this service unless this field is empty.
- Enable Service — Only enabled services are deployed. Disabling a service, effectively undeploys just that service while letting all other services in the application run as normal. This can be useful, for example when you wish to deploy an application that includes a service for which you don't have the required software.

Target Machines

- Remove from Selected Machines — Click to remove this service configuration from the selected machine(s).

- Add to Additional Machines — Adding services to additional machines is useful for fault tolerance. As a rule, it therefore does not make sense to run the same service on the same machine twice.
- Service Instance — Service instance from the selected machine. The service instance name includes the machine name.
- Software — The software required by this service instance.
- Deployment Status — Deployment status, as shown in the Configuration Builder
- FT Weight — The fault tolerance status and weight of the service instance. Appears only if `Run Fault Tolerant` is selected. For an in-depth discussion of this topic, see Configuring Fault Tolerant Process Engines.

FT Group Settings

Appears only if a ActiveMatrix BusinessWorks process is assigned to additional machines. Note that TIBCO Adapter services cannot be assigned fault tolerant options.

- Run Fault Tolerant — If selected, the selected service instances will run in fault tolerant mode.
- Heartbeat Interval (ms) — The master engine of a fault-tolerant group broadcasts heartbeat messages to inform the other group members that it is still active. The heartbeat interval determines the time (in milliseconds) between heartbeat messages. In the event if one process engine fails, another engine detects the stop in the master's heartbeat and resumes operation in place of the other engine. All process starters are restarted on the secondary, and services are restarted to the state of their last checkpoint.
- Activation Interval (ms) — A standard TIBCO Rendezvous fault tolerant parameter, documented in the *TIBCO Rendezvous Concepts*, Developing Fault Tolerant Programs.

  Secondary process engines track heartbeat messages sent from the master engine. This field specifies the amount of time to expire since the last heartbeat from the master before the secondary restarts the process starters and process engines.

  The Heartbeat Interval should be smaller than the Preparation Interval, which should be smaller than the Activation interval. It is recommended that Activation Interval be slightly over 2 heartbeats.
- Preparation Interval (ms) — A standard TIBCO Rendezvous fault tolerant parameter, documented in the *TIBCO Rendezvous Concepts*, Developing Fault Tolerant Programs).

When a master engine resumes operation, the secondary engine shuts down and returns to standby mode. For some situations, it may be necessary to ensure that the secondary engine has completely shut down before the master engine resumes operation.

This field is used to specify a delay before the master engine restarts. When the time since the last heartbeat from an active member exceeds this value, the ranking inactive member will receive a "hint" so that it can prepare for activation.

The Heartbeat Interval should be smaller than the Preparation Interval, which should be smaller than the Activation interval.

Monitoring Tab

Rulebases

Click Add to add an existing custom TIBCO Hawk rulebase. The rulebase must have been configured using the TIBCO Hawk Display. For more information, see Adding a Rulebase to a Process or Service.

Events

Click Add to create an event. For more information, see Adding an Event to a Service.

Failure Count

When an instance is down unexpectedly, the error count and last failure time are tracked. When the error count is greater or equal to the value set for `Reset Failure Count`, or if the value set for `Reset Failure Interval` expires (whichever comes first), the error count is reset to zero.

- Reset Failure Count. The value in this field defines how many restarts should be attempted before resetting the error counter to 0.

  When an instance is down, the TIBCO Hawk Agent will attempt to restart the instance the number of times specified in this field. If the instance restarts after the number of times specified, the event you have defined is triggered.

- Reset Failure Interval (seconds). The value in this field defines how much time should expire before resetting the error counter to 0.

For example, if you define the following three events and set the `Reset Failure Count` to **5**:

- Event 1, restart the instance and send an alert on the first failure.
- Event 2, restart the instance and send an email on the second failure
- Event 3, restart the instance and execute a command on subsequent failures.

  On the first failure, the error count is 1, the instance is restarted and an alert is sent.

  On the second failure, the error count is 2, the instance is restarted and email is sent.

  On third failure, the error count is 3, the instance is restarted and the command you configured is executed.

  On fourth failure, the error count is 4, instance is restarted and the command you configured is executed.

  On fifth failure, the error count is 5 and then reset to 0. The instance is restarted and the command you configured is executed.

  On sixth failure, the error count is 1, the instance is restarted and an alert is sent.

  The cycle repeats.

If you do not want to receive alerts frequently, Reset Failure Count should be set with a high value. When error count is reset to 0, the last failure time is reset as well. The Reset Failure Interval takes effect only after the first failure occurs.

Advanced Tab

Click **Reset to Defaults** to use the defaults defined in the enterprise archive file.

Adapter SDK Properties

Displays if an adapter service is included in the application. Allows you to change TIBCO Adapter SDK properties that were defined in the enterprise archive file.

Runtime Variables

Displays if an adapter service is included in the application. Displays the runtime variables settable for this service. You can change the runtime variable values as required.

ActiveMatrix BusinessWorks Checkpoint Data Repository

If you wish to run ActiveMatrix BusinessWorks using multiple engines in fault tolerant mode, you must specify a checkpoint data repository.

For true fault tolerance, you must store the data in a database.

You specify a `JDBC Connection` resource for the database to be used when you configure your project in TIBCO Designer. The database is then one of the available options on the pop-up menu.

For more information, see Configuring Storage for TIBCO ActiveMatrix BusinessWorks Processes .

ActiveMatrix BusinessWorks Process Configurations

Allows you to change the process configurations. For more information, see Controlling Execution of Services.

The `Flow Limit` parameter always appears in release 5.2 and later and only has meaning when deploying to a ActiveMatrix BusinessWorks release 5.2 process engine. If you use Administrator 5.2 or later to deploy to a ActiveMatrix BusinessWorks release 5.1.3 process engine, the parameter will display, but have no effect on the process engine.

Edit Service Instance Dialog

Fields can be edited if this dialog is invoked from the Configuration Builder pane. If invoked from the Deployed Configuration pane, the fields are read only.

The following tabs are available:

- General Tab
- Server Settings Tab
- Graceful Shutdown Tab

General Tab

The `General` tab displays the following information:

- Software that will run the used by the service instance.

- Machine on which this instance has been set up to run.
- Operating system used by this machine.
- Name of the service instance.
- Description for this service instance.
- Contact for this service instance.

Server Settings Tab

General

- Start on Boot — Specifies that the service instance should be started whenever the machine restarts.
- Enable Verbose Tracing — Enables verbose tracing, in particular, for TIBCO ActiveMatrix BusinessWorks service instances.
- Max Log File Size (KB) — Specifies the maximum size (in Kilobytes) a log file can reach before the engine switches to the next log file.
- Max Log File Count — Specifies the maximum number of log files to use. When log files reach the size specified in the Max Log File Size field, the engine switches to the next log file. When the maximum number of log files have been written, the engine begins writing to the first log file again.
- Thread Count — Specifies the number of threads to use to execute process instances. The number of threads determines how many process instances can execute concurrently. Set the number of threads to a value that is appropriate for your operating system and physical machine configuration.

  You should measure the available CPU and memory resources on your system under a typical processing load to determine if the default value of 8 threads is appropriate for your environment. For example, if engine throughput has reached a plateau, yet measurements show that CPU and memory are not fully utilized, increasing this value can have a positive effect on throughput. Typical numbers of worker threads range between 4 and 32. Specifying too low a value can cause higher memory use and lower engine throughput even though spare CPU resources exist. Specifying too high a value can cause CPU thrashing behavior, or an increase in latency caused by a large number of messages in the message queue.

Java

This pane is only available for Java applications.

- Prepend to Classpath — The items you supply here are prepended to your `CLASSPATH` environment variable. You can specify a Java code editor, or the jar file from a JNDI provider if you wish to use TIBCO ActiveMatrix BusinessWorks to receive and process JMS messages.
- Append to Classpath — The items you supply here are appended to your `CLASSPATH` environment variable. You can specify a Java code editor, or the jar file from a JNDI provider if you wish to use TIBCO ActiveMatrix BusinessWorks to receive and process JMS messages.
- Initial Heap Size (MB) — Initial size for the JVM used for the process engine. Default is 32 MB.
- Maximum Heap Size (MB) — Maximum size for the JVM used for the process engine. Default is 128 MB.
- Java Thread Stack Size (KB) — Size for the thread stack. Default is 128 KB.

NT Service

- Run as NT Service — Select to run this service as a Microsoft Windows Service. You can then manage the engine as you would any other service, and you can specify that it starts automatically when the machine reboots.
- Startup Type — Choose one of the service startup types, Automatic, Manual, or Disabled.
- Login As — Specify the login account for the service, if any. The domain name must be specified. If the user is defined on the local machine, the domain is ".". For example, user jeff on the local machine would be specified as `.\jeff`.
- Password — Click set to define the password for that service, if any.

Graceful Shutdown Tab

This tab appears only if you have displayed this dialog box from an undeployed process. You can specify how a graceful shutdown occurs.

Kill Jobs Timeout

Kill Jobs Timeout specifies the maximum timeout in seconds the process engine will wait for jobs to finish before shutting down the engine. A zero (0) value means 0 seconds, which effectively turns the graceful shutdown into an immediate shutdown.

Wait for Checkpoint

When selected, causes the process engine to wait for all jobs to finish (up to the maximum timeout) before shutting down the engine, rather than removing jobs at their next checkpoint.

View Service Configuration

The following tabs are available:

- General
- Monitoring
- Advanced

General

The `General` pane displays the name of the service and description provided when the service was deployed. It also displays whether additional components are required and whether the service is enabled.

The `Target` pane displays the service instance, the software used for the instance and the deployment status.

Monitoring

Displays the TIBCO Hawk rulebases defined for the service, events defined and the failure count. For more information, see Failure Count.

Advanced

Displays the TIBCO Adapter SDK properties and runtime variables for the service.

Managing and Monitoring Process Engines

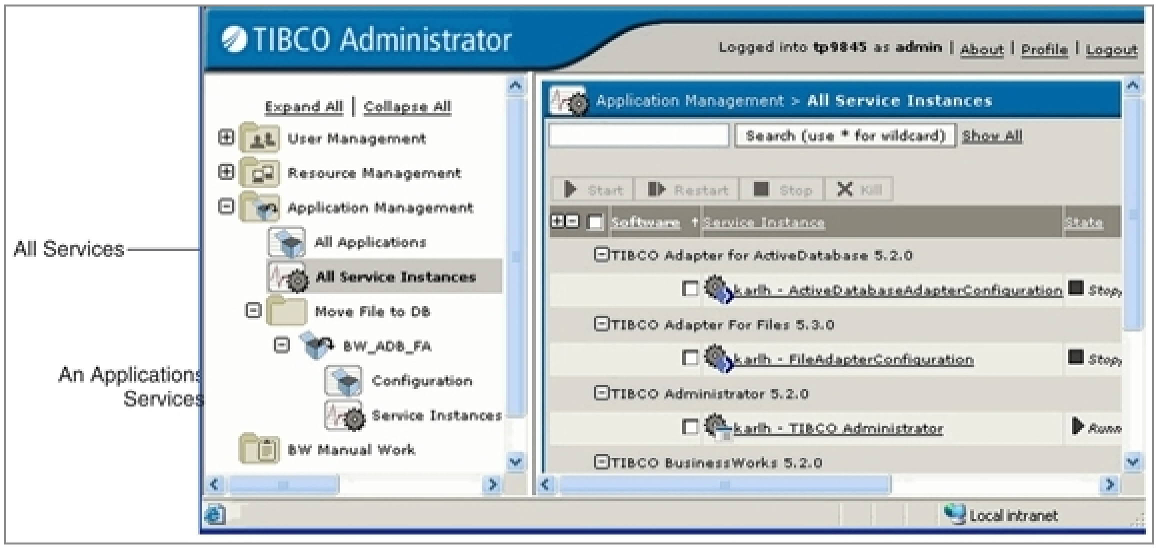
This section explains how to manage and monitor process engines.

Process Engines Overview

You can access process engines as follows:

- To view all process engines for the administration domain, select the Application Management > All Service Instances console. It allows you to view and change the status of all instances running in the administration domain. The console displays the software for which there are running instances (for example, ActiveMatrix BusinessWorks or TIBCO Enterprise Message Service), and then allows you to view all instances for that software.
- To view all process engines for an application, select the corresponding application, and then choose Service Instances. Only process engines for the that application are displayed.

The next diagram shows all service instances in an administration domain.



Software	Service Instance	State
TIBCO Adapter for ActiveDatabase 5.2.0		
	karlh - ActiveDatabaseAdapterConfiguration	Stop
TIBCO Adapter For Files 5.3.0		
	karlh - FileAdapterConfiguration	Stop
TIBCO Administrator 5.2.0		
	karlh - TIBCO Administrator	Runn
TIBCO BusinessWorks 5.2.0		

The All Service Instances console is organized to show the software at top level. You can then select the software to display all associated service instances.

For example, if you created multiple deployments of a ActiveMatrix BusinessWorks process engine, you will see one engine for each deployed process archive. You can select a ActiveMatrix BusinessWorks process engine to view information about its corresponding jobs.

Starting or Stopping a Process Engine

After you have deployed an application, you can change the process engine state from different parts of TIBCO Administrator:

- In the Application Management > All Service Instances console, click the check box next to a service or process, and then choose the appropriate button, which becomes available.
- In the Application Management module, select the All Service Instances console and click the service instance name. In the window that is displayed, select the General tab. State has a clickable word next to it that allows you to start or stop service instances.
- In the Application Management module, select the Service Instances console for the application, click the check box next to a service instance, and then choose the appropriate button, which becomes available.
- You can start a service instance when it is deployed, or you can explicitly start services after deployment. For information about starting services when deploying, see Deploying an Application.
- TIBCO Administrator itself is also listed as component software.

Warning: Shutting down the administration server is not recommended. You cannot start a stopped server from TIBCO Administrator.

You can, however, *re*start the administration server if you are using a database domain or have a secondary server defined in the domain: Select the server and choose **Restart**, which stops and restarts the server. In that case, a "page cannot be displayed" error results in the TIBCO Administrator GUI. You must invoke TIBCO Administrator GUI again and log in once more.

Warning: If you shut down the administration server, all currently running services and processes in the domain will continue to run. However, you can no longer monitor any project and you cannot restart any component in case of failure. In addition, some components load configuration information on demand which will fail if the administration server is not available.

To start or stop a process engine, perform the following procedure:

Procedure

1. Under Application Management, select **All Service Instances** or go directly to a specific application and select **All Service Instances**.
2. Select the service instances, process engines or both to start and click **Start**.

   Click **Stop** or **Kill** to shutdown the service instance or process engine. For more information, see All Service Instances Dialog .

Viewing Log File Information

Tracing options are set in TIBCO Designer when configuring a service or process. You can view the tracing options for a process or service instance and set search parameters to display only a subset of a log file. You can also export the log to a file.

When you display the **Trace** tab, you specify the log file to get information from and the number of lines to return. You can supply one or more search conditions to filter the amount of information to return.

Procedure

- Date/Time before/after — Specify a date to filter. Use two dates to create a range.
- Role — Allows you to choose only traces to certain roles. Choose Info, Warning, Debug, Error, or specify a Custom role.

  The role you choose depends on the role defined for the application. By default, Info, Warning, Error, and Debug are available. Custom roles may also be available if supported by the application.
- Category — Specifies a category. Items for that category are then sent to the trace. For example, if you choose Database, any database access or database errors are included.

Categories include, for example, Configuration, Application, Adapter, Database, TibRvComm, and XML. Custom categories may also be available if supported by the application.

- Detail description — Allows you to specify a detail description for which you want to display (or not) all log entries.

Make sure the process or service `State` indicates the process or service is running.

To view tracing results for a process engine, perform the following procedure:

Procedure

1. Under Application Management, select **All Service Instances** or go directly to a specific application and select **All Service Instances**.
2. Click the process engine name.
3. Click the **Tracing** tab.
4. Click **details** to drill-down view the entry.
5. To export an entry to a file, select the item(s) you wish to export.
6. Click **Export**.
7. Click **Done**.

Editing Process Engine Properties

You can edit active processes, process starters, process definitions and lock properties defined for a process engine.

To edit process engine resource properties, perform the following procedure:

Procedure

1. Under Application Management, select **All Service Instances** or go directly to a specific application and select **All Service Instances**.
2. Click a process engine name.

3. Click the **BW Processes** tab. For more information, see BW Processes
4. Select an item from the drop-down menu. The panel changes, depending on your selection.
5. Click **Done**.

Viewing the TIBCO Administrator Audit Log

For TIBCO Administrator, you cannot configure tracing. You can, however, view the audit log, and filter it to better view the information you need.

In many cases, your browser cannot display the complete log. In that case, define a search condition as discussed in Viewing Log File Information, and then click **Search**.

To view the audit log, perform the following procedure:

Procedure

1. Choose **Application Management > All Service Instances.**
2. Select **TIBCO Administrator**
3. Click the **Audit Log** tab.
4. To display the Administrator log, click **Search**.
5. Optionally, add a search condition and click **Search**.
6. Click **details** to drill-down view the entry.
7. To export an entry to a file, select the item(s) you wish to export. Click **Export**. Click **Done**.

Managing Recoverable Process Instances

A checkpoint saves the state of a process instance at a given point in time. Checkpoints are used to restart or recover a process instance either when an engine fails or if the process instance fails by encountering an unhandled exception or by manual termination in TIBCO Administrator or TIBCO Hawk.

In the event of an engine failure, by default all process instances executing at the time of the failure are automatically restarted and begin executing from the last checkpoint. You can optionally specify that process instances should not be automatically restarted when the engine restarts. This allows you to handle any resource availability problems, such as a database or web server being down, and then later recover the process instances manually when the resource is available. To specify that checkpointed process instances should not automatically restart when a process engine restarts, set the custom property `bw.engine.autoCheckpointRestart` to `false`.

Normally, when a process instance fails, checkpoint data for the process instance is deleted. You can optionally specify that checkpoint data should be saved for terminated process instances so that the process instance can be recovered at a later time. You can enable failed process recovery by setting the custom property `bw.engine.enableJobRecovery` to `true`. For more information about setting custom properties, see Custom Engine Properties.

You can manage recoverable processes either through TIBCO Administrator, TIBCO Hawk, or programmatically using the Engine Command activity in a process instance. This section describes how to manage recoverable process instances using TIBCO Administrator.

For more information about using TIBCO Hawk microagent methods, see Performance Tuning .

For more information about using the Engine Command activity, see *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

To manage and restart recoverable process instances, perform this procedure:

Procedure

1. Under Application Management, select **All Service Instances** or go directly to a specific application and select **All Service Instances**.
2. Click a process engine name.
3. Click the **BW Processes** tab.
4. Select Recoverable Processes from the drop-down menu.
5. Check the box next to the process instances you wish to manage. Click the **Restart** button to restart the process instances or click the **Remove** button to remove the process instances from the list of recoverable processes.

Limitations of Recoverable Process Instances

Because the process engine is running when a process instance is recovered, there are certain limitations. The following describe the limitations:

- If a Java object reference is normally shared among process instances, process instances that start normally or are restarted during process engine startup retain the shared references to the object. However, a process instance that is recovered after a process engine starts receives a unique copy of each Java Object that it references.
- If a recovered process instances uses sequencing, then the process instance may not be able to be restarted if there is already a process instance currently executing that has the same sequence key. Such a job can only be restarted when no other process instances are running.
- If a recovered process instance has a duplicate detection key, then the key is only released when the process instance finishes normally or is deleted.

Storing Process Instance and Activity Statistics

The View Service Instance Dialog displays cumulative statistics for process engines. You can also store statistics for each executed process instance and each executed activity. Statistics are stored in a comma-separated value (CSV) format that can be imported into most analytical tools, such as Microsoft Excel.

Enabling/Disabling Statistic Collection

Statistic collection is controlled differently for process engine statistics and activity statistics. Statistic collection for process instances is controlled by a custom engine property. Statistic collection for activities is controlled dynamically while the process engine is running.

Collecting statistics on a per process instance or activity basis affects the performance of the process engine. Extra processing and I/O is required for gathering and writing the statistics to a file.

Collecting statistics on an activity basis is especially resource intensive. It is recommended that you collect activity statistics for brief periods of time while determining the performance requirements of a system or while tuning an application. It is not recommended that you continuously run activity statistic collection on a production system.

Process Instance Statistic Collection

Process instance statistic collection is controlled by the custom engine property `bw.engine.jobstats.enable`. The default value of this property is `false` indicating that statistics for each process instance should not be stored. Setting this property to `true` enables the gathering of statistics for each process instance.

For more information about custom engine properties, see Custom Engine Properties.

Activity Statistic Collection

You can enable or disable activity statistic collection using either TIBCO Administrator or TIBCO Hawk commands. This section describes the TIBCO Administrator approach. For more information about TIBCO Hawk commands, see Performance Tuning.

To control activity statistic collection, perform the following procedure:

Procedure

1. Under Application Management, select **All Service Instances** or go directly to a specific application and select **All Service Instances**.
2. Click a process engine name.
3. Click the **Engine Control** tab.
4. In the Statistics Collection pane, click the Start button to begin statistics gathering for activities or click the Stop button to halt statistics gathering. The current status of statistic gathering is detailed in the Status field. If statistics gathering is disabled, the Status is Off. If statistics gathering is enabled, the Status field details the location of the file containing activity statistics.

Note: You can enable or disable statistic collection for the activity elapsed time in the Output schema of JDBC activities by setting the property `java.property.bw.activity.output.stats.elapsedTime.`*activty_name* in `designer.tra` for design-time and `bwengine.tra` for runtime. Setting this property causes a performance overhead and hence it is recommended that you use this property in a non-production environment as a diagnostic tool. You can turn off the property and not calculate the elapsed time at run-time by setting the property `java.property.bw.activity.output.stats.elapsedTime.turnoff` in the `bwengine.tra` file. For more information, see *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

Managing Statistics Files

Separate files are kept for process instance statistics and for activity statistics. The name of the activity statistics file(s) are in the format `stats-`*<timestamp>*`-`*<filenumber>*`.csv`. The name of the process instance statistics file(s) are in the format `jobStats-`*<timestamp>*`-`*<filenumber>*`.csv`. These files are stored in a location determined by the `bw.engine.stats.dir` custom engine property. The default location of this property is *<engineWorkingDir>*`/stats`.

For activity statistics, a new file is created each time statistics collection is enabled. For process instance statistics, a new file is created each time the process engine is started or when the statistics file reaches the specified size. You specify the maximum size for job statistics files with the custom engine properly `bw.engine.jobstats.rollover`. The property specifies the maximum size in megabytes

For more information about custom engine properties, see Custom Engine Properties.

Process Instance Statistics

One record per process instance is created and stored in the process instance statistics file. The following table describes the statistics stored for each process instance.

Stored process instance statistics

Statistic	Description
jobId	ID of the process instance.
processName	Name of the process definition the process instance is executing.
startTime	Time stamp at process instance start.
endTime	Time stamp at process instance completion.
elapsedTime	The endTime minus the startTime. The total clock time that has elapsed during the execution of the process instance.
evalTime	The sum of the evaluation times for each executed activity in the process instance. Evaluation time is the time between the beginning and end of the evaluation period. This should be close to but is not exactly equivalent to CPU time, due to the limitations of statistic computation.
status	Status of the process instance at completion. This can be either `succuess` or `error`.

Activity Statistics

One record per completed activity execution is created and stored in the activity statistics file. The following table describes the statistics stored for each activity.

Stored activity statistics

Statistic	Description
startTime	Offset (in milliseconds) from the start of statistic gathering to the time the activity was invoked.
elapsedTime	The time when the activity ended minus the time the activity started. The total clock time that has elapsed during the execution of the activity.
evalTime	The time between the beginning and end of the evaluation period for the

Statistic	Description
	activity. If the activity completes in one step, the evalTime and elapsedTime would be the same. However, some activities, such as Request/Reply or Wait for... activities typically do not complete in one step.
jobId	ID of the process instance in which this activity executed.
processName	Name of the process definition the process instance in which this activity is located.
callStack	When the activity is executed as part of a called process, this column contains the call stack from the original process to the called process.
paged	Specifies whether the process instance was paged out to disk when the activity was invoked.

All Service Instances Dialog

Search

Allows you to display only the items that match a search criteria.

Start

Starts the selected process engine(s).

Restart

Stops the selected process engine(s), and then starts it.

Stop

Stops the selected process engine(s). If graceful shutdown options are set for a process engine, the options are applied. Click a process engine name to access graceful shutdown options.

Kill

Forces an immediate shutdown of each selected process engine. If checkpoints or other graceful shutdown options are defined for a process engine, the options are ignored. Current jobs are terminated before given a chance to complete.

Group By

Determines how items in the display are grouped.

Instances List

- Service Instance — Displays the TIBCO ActiveMatrix BusinessWorks engine, adapter instance, JMS Server service, and so on. Click the component name for additional information.
- State — Stopped, Starting Up, Running, or Shutting Down. If the component belongs to an FT group, `Standby` is also an option.

  Shutting down TIBCO Administrator is not recommended. Restarting is, however, an option.
- Status — Indicates the status for the application. The ??? icon indicates that the instance has lost contact with the endpoint Hawk Agent. Status cannot be determined.
- FT Group — Fault Tolerance group to which this component belongs, if any.
- Machine — The computer on which this component is running.
- Software — Name of the installed TIBCO software that runs the application. The highest alert for that software is displayed in the left-most column.

View Service Instance Dialog

The following tabs are available:

- General Tab
- BW Processes
- Tracing Tab

- Graceful Shutdown Tab

General Tab

General

Displays the following information about a process engine or service instance:

- Uptime for this component.
- Process ID for this component.
- Name of the process.
- Status of the component. If stopped, click **start** to start it. If running, click **stop** to stop it.
- Name of the machine on which this process engine or service instance is running.
- Name of the fault tolerant group, if any, to which this component belongs.

Statistics

This pane only displays for process engines.

- Created Processes — The processes created by the process engine.
- Suspended Processes — The processes currently suspended.
- Swapped Processes — The total number of times processes were swapped up to current.
- Queued Processes — The processes currently queued.
- Aborted Processes — The processes that were aborted.
- Completed Processes — The processes that were completed.
- Checkpointed Processes — The processes currently checkpointed.
- Total Execution (ms) — Total execution time for all processes. This refers to the total time the process was executing but does not include any wait times.
- Average Execution (ms) — Average duration for execution of a process.

Active Alerts

Displays information about the active alerts for this component.

- Date/Time — The date and time at which the alert occurred.
- Alert Level — The alert level set when the alert was created.
- Text — Description defined when creating the alert.

BW Processes

This tab displays only for process engines.

Select Active Processes, Process Starters, Process Definitions, Locks, Recoverable Processes, or Blocked Resources.

Active Processes

Displays active process engines. As a rule, this includes process engines that are suspended or waiting. Examples include process engines that contain a Wait activity and are waiting in a loop. All other process engines usually complete before TIBCO Administrator is updated by auto-refresh and are therefore not displayed.

- Add Search Condition — You can add one or more search condition to narrow the display.
- Export — Click to export information about the selected process engine to a comma-separated file.
- Kill — Stops the selected process engine.
- Suspend — Suspends the selected process engine.
- Resume — Resumes the selected suspended process engine.

Statistic	Description
Process ID	ID of the running process instance.
Status	Status of the process.

Statistic	Description
Tracking ID	Tracking ID for the process instance.
Custom ID	Custom ID for the process instance.
Start Time	Time when the process instance started.
Duration (ms)	Elapsed clock time (in milliseconds) since the process instance started.
Process Defn	Name of the process definition.
Current Activity	Name of the currently executing activity in the process instance.
Starter	Name of the process starter that started this process instance.

Process Starters

Displays all process starters in the process engine. You can then select individual process starters and enable or disable them. This can be useful, for example, if you wish to understand the performance impact of one of the process engines.

Process Definitions

Use the search field to limit the display. The * character can be used as a wildcard. Click a process engine to display the process definition details. The following information is displayed.

- Name — Activity for which throughput is displayed.
- Called Process — This field only shows information if you’re running a process engine called by another process engine.
- Execution Count — Number of jobs in which this activity is currently participating.
- Elapsed Time (ms) — Average time this activity took to complete.
- CPU time (ms) — CPU time used by this activity.
- Errors — Number of errors encountered for this activity.
- Status — Activity status.
- Function — Name of the activity resource.

- details — Click details for more detailed information about this particular activity.

Locks

Lock object shared configuration resources are used by Critical Section groups to ensure that only one process engine executes the activities within a Critical Section group at a time. The lock name, wait position, process id and requestor display. You can export lock information to a comma separated file, or kill a lock, if necessary. For more information, see *TIBCO ActiveMatrix BusinessWorks™ Process Design Guide*.

Recoverable Processes

Recoverable processes are process instances that have been checkpointed but not restarted. The Recoverable Processes option allows you to manage recoverable processes and either restart or remove them. For more information, see Managing Recoverable Process Instances.

Blocked Resources

A process instance can become blocked when resources that it depends upon are unavailable and TIBCO Enterprise Management Advisor blocks their use. For example, a process instance my require a database connection. This option allows you to resume a blocked process instance once the resource becomes available. For more information, see Using TIBCO Enterprise Management Advisor.

Tracing Tab

Allows you to view the trace logs for this application. You can create one or more search conditions to narrow the search scope.

To see the default log, leave `Where File is <project.component>.log` and click **Search**.

The log may grow quite large, and you are therefore encouraged to add one or more a search conditions before you click **Search**. The number of lines displayed is governed by `tibcoadmin.monitor.traceLogMaxLines` in `tibcoadmin_<domain>.tra` and defaults to 1000.

Graceful Shutdown Tab

Edit

Click to change the parameters under this tab.

Kill Jobs Timeout

Kill Jobs Timeout specifies the maximum timeout in seconds the process engine will wait for jobs to finish before shutting down the engine. A zero (0) value means 0 seconds, which effectively turns the graceful shutdown into an immediate shutdown.

Wait for Checkpoint

When selected, causes the process engine to wait for all jobs to finish (up to the maximum timeout) before shutting down the engine, rather than removing jobs at their next checkpoint.

UDDI Servers Module

The UDDI Servers module allows you to browse and publish to UDDI Operator Sites. This section describes the UDDI Servers module.

Overview of UDDI

Universal Description, Discovery, and Integration (UDDI) refers to the protocol used by web-based registries to publish information about web services.

Businesses publish information about the web services they offer to public UDDI Operator Sites. This allows other businesses to locate and access published web services. ActiveMatrix BusinessWorks supports both browsing and publishing to UDDI registries that comply with the UDDI Version 2.0 API specification.

For more information about the UDDI API, see http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm .

The UDDI Servers module allows you to define connections to UDDI servers and view the web services contained in the servers. If you have been granted access to publish your own web services, you can also use the UDDI Servers module to publish information about your business and the web services you offer.

Clicking the UDDI Servers module displays the UDDI Servers panel. You must add UDDI servers to the server list before proceeding with any other operations. The following sections describe the process for adding UDDI servers and the subsequent operations you can perform.

Managing UDDI Servers

The UDDI Servers panel allows you to add, view, or remove UDDI servers from the list of servers.

Adding UDDI Servers

You must add UDDI servers to the server list before performing any other operation in the UDDI module.

To add a new UDDI server to the server list, perform the following procedure:

Procedure

1. Click the **UDDI Servers** module, then click **Add**. The Manage UDDI Server panel appears.
2. Enter the following information about the UDDI server.

Field	Description
Name	Name of the UDDI server.
Description	Description of the UDDI server.
Inquiry URL	URL for browsing the businesses contained in this server.
Publish URL	URL for publishing your business information to this server.
Username	Name of the user that has publish access to this server.
Password	Password for the specified user.
SSL Configuration	Folder containing one or more certificates from trusted certificate authorities. This folder is checked when a client connects to the registry server to ensure that the server is trusted. This prevents connections to rogue servers.
Proxy Server Settings Specify these fields when you access the registry by way of a proxy server.	
Host	Host name of the proxy server.

Field	Description
Port	Port number on the proxy server.
User Name	User name on the proxy server.
Password	Password on the proxy server.

3. Click **Done**.

Viewing and Editing UDDI Server Details

Once a UDDI server has been added to the server list, you can view the server details by clicking the server’s name in the server list. The View UDDI Server Detail panel is displayed. This panel allows you to add businesses to the server, if you have been granted access to publish to this server. You can also edit the server’s details by clicking the Edit button. For more information about adding a business to the server, see Managing Business Entities.

Managing Business Entities

Business entities describe the businesses that publish web services to UDDI servers. A business entity contains discovery URLs, contact information, and services. The following sections describe how to view, add, and edit business entity information.

Viewing or Editing a Business Entity and Services

To view a business entity, perform the following procedure:

Procedure

1. Click the **UDDI Servers** module, then click the UDDI server name that contains the business in the server list.
2. The View UDDI Server Detail panel appears. The Businesses area lists all businesses contained in the UDDI server. The business list contains a list of business names, descriptions, and UUID key values for each business.

Use the Search field to locate the desired business if the list of businesses is not easily seen.

3. Click the name of the business to display the View Business Detail panel. This panel displays the following:

Field	Description
Name	Name of the business.
Key	UUID key for the business.
Description	Description of the business.
Authorized Name	The name of the user that published the business entity.
Services	A list of web services available from this business. For more information about adding and editing services, see Adding Services .

4. Click the **Edit** button to display the Manage Business panel and edit any of the business details. Only authorized users can edit business entities. The Manage Business panel also allows you to add discovery URLs or contact information for this business. For more information, see Adding a Business Entity.
5. The Services area lists the services available for this business entity. You can add web services that are in deployed and running TIBCO Administrator applications by clicking the Add button. For more information, see Adding Services.

Adding a Business Entity

If you are authorized to publish to a UDDI sever, you can add a business entity by performing the following procedure:

Procedure

1. Click the **UDDI Servers** module, then click the UDDI server name to which you wish to add a business entity. The View UDDI Server Detail panel appears.
2. Click **Add** to add a business entity to the list. The Manage Business panel appears.
3. Enter the following information about the business:

Field	Description
Name	Name of the business.
Key	UUID key for the business.
Description	Description of the business.
Authorized Name	The name of the user that published the business entity.

4. Add discovery URLs to this business entity by clicking **Add** in the Discovery URLs area. A discovery URL provides a link to additional information, either technical or descriptive, about your business. Enter the URL and the use type for the URL. UDDI defines two use types for discovery URLs: businessEntity and businessEntityEx. For more information about discovery URLs, see the UDDI specification .
5. Add contact information to this business entity by clicking **Add** in the Contacts area. The Edit Contacts Detail panel appears where you can add a contact name, description, phone, email, and address.
6. Click **Done**.

Adding Services

You can add ActiveMatrix BusinessWorks web services that are deployed and running to your business entities. This allows you to publish the details of your web services to an external audience. You can only add Service resources or process definitions that contain a SOAP Event Source process starter to a business entity.

To add ActiveMatrix BusinessWorks web services to your business entity, perform the following procedure:

Procedure

1. Click the **UDDI Servers** module, then click the UDDI server name that contains your business entity.
2. Click your business name in the list of businesses for the selected UDDI server.
3. Click **Add** in the Services area to add a ActiveMatrix BusinessWorks web service. The Add Service panel appears.

4. Click **Browse** to view a list of deployed and running applications that contain web services. A panel appears that allows you to select the desired Service resources or process definitions that contain SOAP Event Source process starters.
5. Click **Done** once you have selected the desired web services. The web services are added to the Services area.
6. Click each service name to bring up the View Business Service Detail panel.
7. Add bindings to each web service by clicking **Add** in the Bindings area. The Edit Service Binding Detail panel appears.
8. Enter the appropriate binding information:

Field	Description
Access Point	The URL where the service can be accessed.
Binding Key	The UUID binding key for the service. See the UDDI specification for more information.
Service Key	The UUID service key for the service. See the UDDI specification for more information.

9. Click **Done** to dismiss the Edit Service Binding Detail panel and save your changes.
10. Click **Done** to dismiss the View Business Service Detail panel and save your changes.

Custom Engine Properties

ActiveMatrix BusinessWorks process engines can be configured using custom properties in configuration files. This section describes the custom properties that can be altered.

Overview of Custom Engine Properties

The ActiveMatrix BusinessWorks process engine is responsible for running instances of your process definitions. The default configuration settings of the engine are sufficient for most users. However, you can specify custom properties in the engine's configuration files to configure the process engine to suit your needs. For example, custom properties are available for enabling or disabling and setting the level of tracing for the engine. Custom properties are also available for configuring the maximum and minimum number of connections for the HTTP server that handles incoming HTTP requests for ActiveMatrix BusinessWorks.

Properties are set by specifying their name and value in the configuration files. For example, the following line sets the property `Trace.Role.error` to false. This prevents any trace messages for the role named `error` from being written to the log file or console.

```
Trace.Role.error = false
```

Some properties can be set for specific process definitions or activities, and the property name can be variable. Properties that have variable portions can use the wildcard character (`*`) to indicate the property should be set to the specified value for all potential names. For example, `Trace.Role.*` is the property to control tracing for all roles.

Property names and values can be separated by either a space ( ), an equal sign (`=`), or a colon (`:`). If a property value contains a space, equal sign, or colon, you must escape these characters in the property value by using a \ (for example, `\ `, `\=`, or `\:`). You can place comments in the configuration files by placing a hash (`#`) as the first character in a comment line.

The following sections describe how to set custom engine properties and list the custom properties that you can set.

Setting Custom Engine Properties for the Testing Environment

TIBCO Designer runs a process engine when you test process definitions using the Tester tab. To set custom properties for the process engine that TIBCO Designer runs, you must create a properties file and specify its location. Perform the following procedure to set custom engine properties for the testing environment.

Procedure

1. Create a properties file containing the custom properties you wish to set in the process engine that runs in the testing environment. For example, create a file named `properties.cfg`.
2. Add properties to your file.
3. Start TIBCO Designer and open the project you wish to test.
4. Click the Start Testing Viewed Process button to start the test engine. For more information about the process engine during testing, see *TIBCO ActiveMatrix BusinessWorks™ Process Design Guide*.
5. On the Select Processes to Load dialog, click **Advanced**.
6. In the Test Engine User Args field, enter the `-p` argument, followed by the location of your `properties.cfg` file. For example,

   ```
   -p c:/tibco/properties.cfg
   ```

Setting Custom Engine Properties in Deployed Projects

TIBCO Administrator is responsible for deploying process engines in a production environment. ActiveMatrix BusinessWorks provides a file for specifying any custom properties you wish to set in deployed engines. The `bwengine.xml` file is located in the `lib\com\tibco\deployment` subdirectory of the ActiveMatrix BusinessWorks installation directory. For example, on Microsoft Windows machines, this file would by default be located in `c:\tibco\bw\`<*release_number*>`\lib\com\tibco\deployment\bwengine.xml` where <*release_number*> is the release number of the currently installed TIBCO Designer.

The `bwengine.xml` file has a `<properties>` element that defines all of the properties you would like to have available in deployed process engine. Each property is contained in a `<property>` element with the following structure:

```
<property>
   <name>Name to display in TIBCO Administrator</name>
   <option>name of property</option>
   <default>default value</default>
   <description>short description of property</description>
</property>
```

For example, to include the `Trace.Role.*` property in deployment configurations, you would add the following to the `bwengine.xml` file:

```
<property>
   <name>Trace All Roles</name>
   <option>Trace.Role.*</option>
   <default>false</default>
   <description>Controls tracing of all roles.</description>
</property>
```

Once the property is defined in the `bwengine.xml` file, it is available in Enterprise Archive Files that are created by TIBCO Designer and are displayed in the **Advanced** tab of the deployment configuration in TIBCO Administrator. Make sure to re-save EAR files in TIBCO Designer and re-load them into any deployment configurations created in TIBCO Administrator after changing the `bwengine.xml` file. You can alter the value of any property on the **Advanced** tab of the deployment configuration and that value is used in the deployed project.

For more information about creating and managing deployment configurations, see Creating and Deploying Applications.

Enabling Built-in Resource Provider

ActiveMatrix BusinessWorks provides parameters to enable or disable the Built-in Resource Provider feature. This feature performs tasks similar to that of the **Retrieve Resource** activity, but eliminates the need to create the Retrieve Resource process.

The properties to enable, disable and control the Built-in Resource Provider feature can be set in the `bwengine.xml`. For more information on the Built-in Resource Provider feature and the available properties, see "Built-in Resource Provider" in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

Available Custom Engine Properties

The following sections describe the custom properties that you can set. Most properties are boolean and can be set to a value of `true` or `false` to enable or disable them. When a property has a non-boolean value, its syntax is explained in the property description.

Engine Properties

This section describes properties that control the behavior of the process engine.

bw.engine.autoCheckpointRestart

This property controls whether checkpointed process instances are automatically restarted when a process engine restarts. By default, this property is set to `true`, indicating that checkpointed process instances should automatically be restarted. You can set this property to `false`, and any checkpointed process instances can later be recovered using the Job Recovery dialog in TIBCO Administrator. This allows you to handle any resource availability problems such as database recovery or bringing up a web server before handling the process instance recovery.

For more information about process instance recovery, see Managing Recoverable Process Instances.

bw.engine.dupKey.enabled

This property controls whether duplicate detection is performed. `true` (the default) indicates the process engine will check for identical duplicateKey values. `false` indicates `duplicateKeys` when specified are ignored.

For more information about duplicate detection, see *TIBCO ActiveMatrix BusinessWorks™ Process Design*.

bw.engine.dupKey.timeout.minutes

This property specifies how long (in minutes) to keep stored `duplicateKeys`. The default is 30 minutes. 0 indicates the duplicateKey is removed when the job is removed. However, if bw.engine.enableJobRecovery=true, the job is not automatically removed after a failure so the duplicateKey will remain as long as the job remains. Such a job can be restarted or purged later. -1 indicates to store duplicateKey values indefinitely. Any positive integer greater than 0 indicates the number of minutes to keep stored duplicateKeys.

For more information about duplicate detection, see *TIBCO ActiveMatrix BusinessWorks™ Process Design*.

bw.engine.dupKey.pollPeriod.minutes

Specifies the number of minutes to wait before polling for expired `duplicateKey` values.

For more information about duplicate detection, see *TIBCO ActiveMatrix BusinessWorks™ Process Design*.

bw.engine.enableJobRecovery

This property specifies whether checkpoint data for process instances that fail due to unhandled exceptions or manual termination should be saved. Saving the checkpoint data allows the process instance to be recovered at a later time. By default, this property is set to false indicating that checkpoint data for failed process instances is not saved. Setting this property to true saves checkpoint data for failed process instances and these process instances can be recovered at a later time using the Job Recovery dialog in TIBCO Administrator.

For more information about process instance recovery, see Managing Recoverable Process Instances.

bw.engine.stats.dir

This property specifies the location of the process instance and activity statistic files when statistics storing is enabled. The default location of this property is *<engineWorkingDir>*`/stats`.

For more information about statistic collection, see Storing Process Instance and Activity Statistics.

bw.engine.jobstats.enable

This property controls process instance statistic collection. The default value of this property is `false` indicating that statistics for each process instance should not be stored. Setting this property to `true` enables the gathering of statistics for each process instance.

For more information about statistic collection, see Storing Process Instance and Activity Statistics.

bw.engine.jobstats.rollover

This property specifies the maximum size (in bytes) for process instance statistic files. Once a file reaches the specified size, a statistics are written to a new file. The default value of this property is 1024 (1 MB).

For more information about statistic collection, see Storing Process Instance and Activity Statistics.

EnableMemorySavingMode or EnableMemorySavingMode.<*processName*>

Memory saving mode can reduce the memory used by actively running process instances as well as potentially improve the performance of checkpoints. By default, memory saving mode is disabled, but you can enable garbage collection on specific process instances by setting the `EnableMemorySavingMode.`<*processName*> property to `true`. You can enable memory saving mode for all process instances by setting the `EnableMemorySavingMode` property to `true`.

For more information, see *TIBCO ActiveMatrix BusinessWorks™ Process Design*.

Engine.dir

When the process engine is configured to use local file for storage, this property controls the location of the process engine storage. By default, this is set to <*TIBCO_Home*>`/tra/domain/`<*domainName*>`/application/`<*appName*>. Normally, you should not need to change the default location of engine storage. For more information, see Configuring Storage for TIBCO ActiveMatrix BusinessWorks Processes.

Engine.ShutdownOnStartupError

By default, checkpointed process instances are restarted when the engine restarts, and if the engine encounters errors during startup, the restarted process instances continue to be processed and may eventually be lost depending upon the type of error at startup. You can specify that the process engine should shutdown if any errors are encountered during startup so that checkpointed jobs are not lost in the event of an error. The custom engine property named `Engine.ShutdownOnStartupError` controls this behavior. By default, the value of the property is `false`, but setting it to `true` shuts the engine down if errors are encountered when the engine starts.

For more information about the Checkpoint activity, see *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

Engine.StandAlone

Under some situations, a unique constraint violation is thrown when using a database as the data manager for process engines. Set this property to `false` if you encounter this situation.

Engine.StepCount

This property controls the max number of execution steps unless inside a transaction for a job before an engine thread switch occurs. The default value of this parameter is 20.

Frequent thread switching can cause engine performance degradation, but when a process instance keeps the tread too long, this may cause less concurrency for executing process instances (and therefore inefficient use of CPU). Therefore, it is difficult to determine the correct value for this property. The default value is sufficient for most situations, but if your process definitions contain a large number of activities and especially if they contain a large number of activities in iteration loops, you may benefit from setting this property to a higher value.

Engine.ThreadCount

This property controls the number of threads available for executing process instances concurrently. The default value is 8.

On a multi-CPU machine, the ThreadCount value can be increased.However, too many threads can cause resource contention. Hence you need to experiment with it to decide on a higher ThreadCount value.

TIBCO Hawk Properties

TIBCO Administrator is the preferred monitoring and management tool for ActiveMatrix BusinessWorks. However, process engines have a TIBCO Hawk microagent as well. The properties in this section should be set only on deployed engines. These properties are not intended to be used with process engines started by TIBCO Designer for testing process definitions.

For more information about using TIBCO Hawk to monitor and ActiveMatrix BusinessWorks, see Performance Tuning.

Hawk.Enabled

Controls whether or not TIBCO Hawk can be used to monitor and manage the process engine. Also, allows the Engine Command activity to be used. The following table describes the valid values for this property:

Value	Description
`true`	Enables both TIBCO Hawk and Engine Command activity usage.
`local`	Enables only Engine Command activity. TIBCO Hawk cannot be used when this value is used.
`false`	Disables both TIBCO Hawk and Engine Command activity usage.

Hawk.Service

Specifies the service parameter for the TIBCO Rendezvous transport of your TIBCO Hawk configuration. By default this is set to `7474`. For more information about the syntax of the service parameter of TIBCO Rendezvous transports, see the TIBCO Rendezvous documentation .

Hawk.Network

Specifies the network parameter for the TIBCO Rendezvous transport of your TIBCO Hawk configuration. By default this is set to "". For more information about the syntax of the network parameter of TIBCO Rendezvous transports, see the TIBCO Rendezvous documentation .

Hawk.Daemon

Specifies the daemon parameter for the TIBCO Rendezvous transport of your TIBCO Hawk configuration. By default this is set to `tcp:host:7474`. For more information about the syntax of the daemon parameter of TIBCO Rendezvous transports, see the TIBCO Rendezvous documentation .

Instrumentation.<*processName*>

Some of the TIBCO Hawk instrumentation methods require runtime actions that impose performance and memory overhead. These actions can be enabled or disabled on a per-process definition basis at any time by setting this property. The actions that can be enabled or disabled are:

- Collection of activity statistics for the GetActivity microagent method
- Calls to OnProcessActivity and OnProcessStatusChanged microagent methods

Setting the engine property `Instrumentation.*` to true enables those actions for all process definitions. Setting the property `Instrumentation.`<*processName*> to true enables those actions for a specified process definition. Setting this property to false disables the actions.

The instrumentation properties can be set at runtime by calling the TIBCO Hawk `setInstrumentProperties` method. The property value specified in a call to `setInsrumentProperties` takes effect immediately.

TIBCO Enterprise Management Advisor Property

ActiveMatrix BusinessWorks can work with TIBCO Enterprise Management Advisor (EMA) to suspend business processes when external resources become unavailable.

bw.engine.emaEnabled

Setting this property to true enables communication with TIBCO EMA. A resource dependency list for all process definitions executing in this engine is created and processes are suspended when TIBCO EMA communicates the unavailability of any dependent resources. For more information about TIBCO EMA, see Using TIBCO Enterprise Management Advisor .

Trace Properties

Trace properties control which trace messages are sent and where they are sent to. Tracing is controlled either by roles, by activities, or by process definitions. For roles, you can configure system role tracing (Error, Warn, Info, Debug), or you can configure tracing for user-defined roles. The Write to Log activity allows you to specify a user-defined role for the message to write.

Specifying Location of Trace Messages

The following properties control where trace messages are sent. Messages can be sent to the log file, to the console, or published as TIBCO Rendezvous messages.

Trace.Role.<*userRoleName*>.Term or Trace.<*systemRoleName*>.Term

`Trace.Role.<`*userRoleName*`>.Term` controls whether or not messages for the specified user-defined role are sent to the console; use `Trace.Role.*.Term` to control console output for all user-defined roles.

`Trace.<`*systemRoleName*`>.Term` controls whether or not messages for the specified system role (Error, Warn, Info, or Debug) are sent to the console.

Trace.Role.<*userRoleName*>.Log or Trace.<*systemRoleName*>.Log

`Trace.Role.<`*userRoleName*`>.Log` controls whether or not messages for the specified user-defined role are sent to the log file; use `Trace.Role.*.Log` to control log output for all user-defined roles.

`Trace.<`*systemRoleName*`>.Log` controls whether or not messages for the specified system role (Error, Warn, Info, or Debug) are sent to the log file.

Trace.Role.<*systemRoleName*>.Publish

Trace.Role.<systemRoleName>.Publish controls whether or not messages for the specified system role (Error, Warn, Info, or Debug) are published as a TIBCO Rendezvous message. By default, the messages are sent on TIBCO ActiveMatrix BusinessWorks default transport. You can specify a different transport for published trace messages with the following properties:

- `Trace.`<*systemRoleName*>`.Publish.Subject`
- `Trace.`<*systemRoleName*>`.Publish.Service`
- `Trace.`<*systemRoleName*>`.Publish.Network`
- `Trace.`<*systemRoleName*>`.Publish.Daemon`

See the *TIBCO Rendezvous* documentation for the correct syntax for specifying transport parameters.

Specifying Rolling Log Files for UserRole

You can specify that entries for the role named UserRole are sent to a set of rolling log files. To accomplish this, you specify the location of the log files, log file name, the number of log files, and the maximum size of each log file. Entries will be written to the first log file until it reaches its maximum size, and then entries are then directed to the second log file until it reaches its maximum size, and so on. Once the maximum number of log files is reached, entries are then directed back to the first log file again. The following engine properties allow you to configure rolling log files:

- `Trace.Role.UserRole.Log.Dir` — Location for the set of rolling log files.
- `Trace.Role.UserRole.Log.File` — Filename for the log files. A number is appended to each new log file created up to the specified maximum number of log files.
- `Trace.Role.UserRole.Log.MaxSize` — Maximum size of a log file before entries are directed to the next log file in the sequence.
- `Trace.Role.UserRole.Log.Maximum` — Maximum number of log files to create. Entries are directed back to the first log file when the maximum number of log files have been created.

Tracing by Role

The following properties enable or disable all tracing for user-defined and system roles.

Trace.Role.<*userRoleName*> or Trace.<*systemRoleName*>.*

Enables or disables the specified role. `Trace.Role.<`*userRoleName*`>` enables or disables the specified user-defined role; specify Trace.Role.* to enable or disable all user-defined roles. `Trace.<`*systemRoleName*`>.*` enables or disables the specified system role (Error, Warn, Info, or Debug).

Tracing by Resource

The following properties enable or disable tracing for activities and process starters.

Trace.Task.*

Controls whether or not trace messages for all activities are output.

Trace.Task.<*processDefinition*>.<*activityName*>

Controls whether or not trace messages for a given activity in a process definition are output. Specifying a wildcard for the process definition name indicates you would like to control trace messages for all activities with a given name. Specifying a wildcard for the activity name indicates you would like to control trace messages for all activities in the specified process definition.

Trace.JC.<processStarterName>

Controls whether or not trace messages for a given process starter are output. Specify `Trace.JC.*` to control trace messages for all process starters.

Including Activity Input/Output in Trace Messages

When resource tracing is enabled, you can optionally include the resource input or output XML in the trace messages. The following properties determine whether input or output are included.

bw.engine.showInput

When set to `true`, resources that have input will include the input XML in the trace messages for that resource.

bw.engine.showOutput

When set to `true`, resources that have output will include the output XML in the trace messages for that resource.

TIBCO Rendezvous Advisory Messages

TIBCO Rendezvous advisory messages can be written to the ActiveMatrix BusinessWorks log file. There are three types of advisory messages: Error, Warn, and Info. Error advisories are logged by default. The following properties control whether TIBCO Rendezvous advisory messages are sent to the log file:

- `Trace.RV.Advisory.Error`
- `Trace.RV.Advisory.Warn`
- `Trace.RV.Advisory.Info`

Prefix the above properties by "java.property." to enable the associated advisory messages.

XPath and XML Properties

The following properties control behavior of XPath and XML in ActiveMatrix BusinessWorks. Prefix these property names by "`java.property`." while setting the properties in the configuration files.

com.tibco.xml.xpath.create-dateTime.has.timezone

This property determines whether a time zone is added by the XPath function `create-dateTime`. TIBCO strongly advises against modifying this property unless you are told to do so by TIBCO Support.

com.tibco.xml.xpath.variable-declaration-required

This property controls whether variable references are checked. The default value of this property is `false`, indicating that variable references are not checked.

com.tibco.xml.schema.preserve-boolean-lexical-value

This property specifies whether the lexical value of xs:boolean is preserved. By default, this property is `false`. TIBCO strongly advises against modifying this property unless you are told to do so by TIBCO Support.

Security Properties

The following properties control the behavior of Secure Sockets Layer (SSL) and other security settings. Some protocols such can use SSL to ensure secure communication. Properties in this section apply to resources that use SSL.

bw.plugin.security.strongcipher.minstrength

The `bw.plugin.security.strongcipher.minstrength` property specifies the cipher suites that you want to exclude when the **Strong Cipher Suites Only** checkbox is checked in an SSL configuration. This property allows you to choose the types of cipher suites you want to disable. Equivalent key strength is considered for example ciphers like 3DES using 168 bits would be equivalent to an equivalent key length of 112 bits. The default value of this property is DISABLED_CIPHERS_BELOW_128_BIT. This property is also only applicable for resources that have the **Strong Cipher Suites Only** field checked.

The following are the valid values for this property:

Property Value	Description
`DISABLED_ CIPHERS_ EXPORTABLE`	Cipher suites that are suitable for export out of the United States are disabled. This list of exportable cipher suites is controlled by the US government. This usually refers to asymmetric algorithms (such as RSA) with a key of modulus lower than 512 bits or symmetric algorithms (such as DES) of key length 40 or lower. Typically exportable cipher suites contain _EXPORT_ in the suite name, but this is not always the case.
`DISABLED_ CIPHERS_ BELOW_128_`	Cipher suites whose key length (or equivalent) is below 128 bits are disabled.

Property Value	Description
BIT	
DISABLED_CIPHERS_128BIT_AND_BELOW	Cipher suites whose key length (or equivalent) is 128 bits or less are disabled.
DISABLED_CIPHERS_BELOW_256BIT	Cipher suites whose key length (or equivalent) is below 256 bits are disabled.

By default, the jurisdiction policy files shipped with ActiveMatrix BusinessWorks are not unlimited strength. When you disable lower strength cipher suites, you may receive an error suggesting that you should upgrade your policy files. To download and install unlimited strength policy files, perform these steps:

Procedure

1. Download the required files from the following website:

   For all platforms except IBM: http://java.sun.com/javase/downloads/index.jsp

   For IBM platforms: https://www14.software.ibm.com/webapp/iwm/web/reg/pick.do?source=jcesdk&lang=en_US
2. Unzip `jce_policy-1_7_0.zip`.
3. Copy US_export_policy.jar and local_policy.jar to: *TIBCO_home*`\jre\1.7.0\lib\security`.

General Activities Properties

The following properties control behavior of activities in the General Activities palette.

Engine.WaitNotify.SweepInterval

Notify timeouts cause the notify information to be marked for removal, but the information is removed at regular intervals. The default interval for checking Notify timeouts is 60 seconds. If you wish to alter the interval, you can do so by setting the

`Engine.WaitNotify.SweepInterval` property to the desired number of seconds. However, as you decrease the number of seconds in the interval you will incur greater engine overhead.

log.file.encoding

The value of this property specifies the character encoding to use when writing to the log file. Any valid Java character encoding name can be used. For a list of potential character encoding names, see the **Encoding** field on the **Configuration** tab of the **Parse Data** activity. If this property is not specified, the default encoding of the Java Virtual Machine used by the process engine is used.

WaitNotify.Service

When **Wait** and **Notify** activities are used across multiple engines, TIBCO Rendezvous is used for communication between the engines. This property specifies the service parameter for the TIBCO Rendezvous transport. For more information about the syntax and default value of the service parameter of TIBCO Rendezvous transports, see the TIBCO Rendezvous documentation.

WaitNotify.Network

When **Wait** and **Notify** activities are used across multiple engines, TIBCO Rendezvous is used for communication between the engines. This property specifies the network parameter for the TIBCO Rendezvous transport. For more information about the syntax and default value of the daemon parameter of TIBCO Rendezvous transports, see the TIBCO Rendezvous documentation.

WaitNotify.Daemon

When **Wait** and **Notify** activities are used across multiple engines, TIBCO Rendezvous is used for communication between the engines. This property specifies the daemon parameter for the TIBCO Rendezvous transport. For more information about the syntax and default value of the daemon parameter of TIBCO Rendezvous transports, see the TIBCO Rendezvous documentation.

HTTP Properties

In some situations, you may wish to alter the configuration of the HTTP server that receives incoming HTTP requests for TIBCO ActiveMatrix BusinessWorks. This section lists the properties for configuring the HTTP server.

bw.plugin.http.server.restrictHttpMethods

Using this property, specific HTTP methods can be disabled. By default, none of the HTTP methods are restricted by the server. You can specify a comma-separated list of methods that are to be restricted. These restrictions are then applicable to all resources accessed on this server, for all roles.

You cannot disable methods selectively for a particular service or for a particular server.

bw.plugin.http.protocol.single-cookie-header

This property allows you to send multiple cookies in a single, non-repeating Cookie header element for outgoing HTTP requests in the Send HTTP Request activity.

bw.plugin.http.server.allowIPAddresses

This property allows you to specify a pipe-separated list of regular expression patterns that is compared with the remote client's IP address before accepting or rejecting requests from the client. The remote IP address of the client must match the expression patterns for the request to be accepted.

For example : To allow requests from any IP address matching with pattern
`127\.\d+\.\d+\.\d+ or 10\.\d+\.\d+\.\d+`
specify the property as

```
bw.plugin.http.server.allowIPAddresses=127\\.\\d+\\.\\d+\\.\\d+|10\\.\\d+\\.\\
d+\\.d+
```

Note: Use Double back slash in place of single back-slash in an expression.

bw.plugin.http.server.restrictIPAddresses

This property allows you to specify a comma-separated list of regular expression patterns that is compared with the remote client's IP address before accepting or rejecting requests from the client. The remote address of the client must not match for any request from this client to be accepted.

bw.plugin.http.server.acceptCount

This property specifies the maximum queue size for incoming requests. Incoming requests that are not handled by available threads (see `bw.plugin.http.server.minProcessors` and `bw.plugin.http.server.maxProcessors`) are placed on the queue until they can be processed. If the queue is full, new incoming requests are refused with an error. The default value of this property is 100. This property is available only when the server type 'Tomcat' is selected.

bw.plugin.http.server.serverType

This property specifies the server type that is to be used for the HTTP Connection resource. Two server types are available: Tomcat and HTTP Component. The default value of this property is Tomcat.

bw.plugin.http.server.httpcomponents.workerThread

This property specifies the maximum number of web server threads available to handle HTTP requests for the HTTPComponents server type. The default value of this property is 50.

bw.plugin.http.server.minProcessors

This property specifies the minimum number of threads available for incoming HTTP requests. The HTTP server creates the number of threads specified by this parameter when it starts up. The default minimum number of threads is 10.

If the Flow Limit deployment property is set, the value of this property is set to *<valueOfMaxProcessorsProperty>*`/2`.

bw.plugin.http.server.maxProcessors

This property specifies the maximum number of threads available for incoming HTTP requests. The HTTP server will not create more than the number of threads specified by this parameter. The default maximum number of threads is 75.

If the Flow Limit deployment property is set, the value of this property is set to *<valueOfFlowLimit>* `- 1`.

bw.plugin.http.server.maxSpareProcessors

This property specifies the maximum number of unused request processing threads that can exist until the thread pool starts stopping the unnecessary threads. The default maximum number of spare threads is 50.

bw.plugin.http.client.ParseEntireMultipartMessage

This property enables the HTTP client to parse the entire multi-part message.

For ActiveMatrix BusinessWorks 5.2, an HTTP response message that is received by the HTTP client, is parsed on the content-type header. When the message is a multi-part message, the attachments are put in the attachment list. In ActiveMatrix BusinessWorks 5.2, the message body that is exposed to the user should contain the entire message body, including the attachments.

However, parsing a multi-part message is not a problem in ActiveMatrix BusinessWorks 5.3 and later versions as MIME attachments are handled differently.

bw.plugin.http.client.ResponseThreadPool

By default, each Request/Response activity that uses the HTTP protocol (for example, Send HTTP Request or SOAP Request Reply) is associated with a unique thread pool. Each request is executed in a separate thread, belonging to the thread pool associated with the activity. The size of each thread pool is 10 by default, therefore, only 10 requests can execute concurrently.

Setting this property to a value specifies the size of the thread pool to use for request/response activities. This thread pool can be for each activity, or all activities can share the same thread pool. For more information about determining the type of thread pool to use, see bw.plugin.http.client.ResponseThreadPool.type .

The thread pool is created when the engine starts, therefore be careful to set the value of this property to a reasonable number for your system. If you set the value too high, it may result in extra resources allocated that are never used.

bw.plugin.http.client.ResponseThreadPool.type

This property determines the type of thread pool to use for request/response activities. Either one thread pool per activity is created, or one common thread pool is created to be shared across all activities. Specify `default` as the value of this property if you wish to create a thread pool for each activity. Specify single as the value of this property if you wish to create a single, common thread pool for all activities.

The size of the thread pool is determined by the value of the property bw.plugin.http.client.ResponseThreadPool. When the thread pool type is `default`, a thread pool of the specified size is created for each request or response activity. When the thread pool type is `single`, one thread pool of the specified size is created and all activities share the same thread pool.

bw.plugin.http.client.usePersistentConnectionManager

This property specifies that a pool of HTTP connections to each HTTP server should be created so that connections can be reused by Send HTTP Request activities. Not all HTTP servers support persistent connections. Refer to your HTTP server documentation for more information about support for persistent connections.

When this property is set to true, a pool of connections is created for each HTTP server that Send HTTP Request activities connect to. The total number of connections in the pool is limited by the `bw.plugin.http.client.maxTotalConnections` property. The number of connections for each host is limited by the `bw.plugin.http.client.maxConnectionsPerHost` property.

The default value of this property is false.

For more information, see "Send HTTP Request" in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

bw.plugin.http.client.maxConnectionsPerHost

The value of this property is ignored unless the `bw.plugin.http.client.usePersistentConnectionManager` property is set to true. This property specifies the maximum number of persistent connections to each remote HTTP server.

The default value for this property is 20.

For more information, see "Send HTTP Request" in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

bw.plugin.http.client.maxTotalConnections

The value of this property is ignored unless the `bw.plugin.http.client.usePersistentConnectionManager` property is set to true. This property specifies the maximum number of persistent connections to create for all HTTP servers.

The default value for this property is 200.

For more information, see "Send HTTP Request" in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

bw.plugin.http.client.checkForStaleConnections

The value of this property is ignored unless the `bw.plugin.http.client.usePersistentConnectionManager` or `bw.plugin.http.client.usePersistentConnectionManagerForSSL` property is set to true. For persistent connections, the HTTPComponent Library 4.5.5 defines the period of inactivity in milliseconds after which persistent connections must be re-validated before being allocated to the consumer. This check helps to detect connections that have become stale when the connections were kept inactive in the pool.

A non-positive value passed to this method disables connection validation. The default value of this property is 2000 ms

For more information, see "Send HTTP Request" in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

bw.plugin.http.handleAllMimePartsAsAttachment

In previous releases, when the content-type of an incoming message was "multipart/*", the first part of the message was presented as the POSTDATA. This is incorrect according to MIME specification. The `bw.plugin.http.handleAllMimePartsAsAttachment` property fixes this problem.

If this property is set to `true` and the top-level content-type of the incoming HTTP message is "multipart/*", then an HTTP Receiver will present all the MIME parts as attachments and the POSTDATA field will be empty. If this property is set to `false` (the default value), backward compatibility is maintained and the first MIME part is presented as the POSTDATA.

Warning: Do not check the Parse Post Method Data field on the HTTP Receiver process starter when this property is set to true. This causes an error to be thrown.

bw.plugin.http.server.debug

When set to `true`, specifies that the contents of incoming HTTP requests are written to the log file. Writing each request to a log file does incur some overhead and additional processing time.

bw.plugin.http.server.defaultHost

Specifies the name of the default host to use when the machine has multiple domains or IP addresses. The value of this parameter can be either a host name or IP address.

When the hostname is `localhost`, ActiveMatrix BusinessWorks considers the machine as a non multi home environment. Hence it is *not required* to set the `bw.plugin.http.server.defaultHost` property in `bwengine.tra` file.

However, when the hostname is anything other than the `localhost`, then ActiveMatrix BusinessWorks considers the machine as a multi home environment. Set the `bw.plugin.http.server.defaultHost` property in `bwengine.tra` file to the same value as has been set in the host field of HTTP Shared Connection for default host.

JDBC Properties

This section describes custom engine properties that can be set for resources in the JDBC palette.

Engine.Database.TestStatement.<*name*>

When a SQL error occurs during statement execution, TIBCO ActiveMatrix BusinessWorks executes a test SQL statement to determine if the error is caused by a bad connection. If the error is due to a bad connection, the statement can be re-executed using a different connection in the connection pool.

This property allows you to specify a test SQL statement. Specify the database name in the <*name*> portion of the property and set the value of the property to a valid SQL statement.

Engine.DBConnection.idleTimeout

Normally, connections in the database connection pool close after a period of time when they are idle. This property specifies the time (in minutes) to allow database connections to remain idle before closing them. The default timeout for database connections is 5 minutes, but you can set this property to the amount of time you would like to keep database connections open.

Config.JDBC.Connection.SetLoginTimeout

Time (in seconds) to wait for a successful database connection. Only JDBC drivers that support connection timeouts can use this property. If the JDBC driver does not support connection timeouts, the value of this field is ignored. Most JDBC drivers should support connection timeouts. The value of this property overrides any value set for connection timeouts in the Configuration tab of the JDBC Connection resource.

JMS Properties

This section describes custom engine properties that can be set for resources in the JMS palette.

bw.plugin.jms.receiverTimeout

This property specifies the polling interval for JMS activities that receive messages (for example, JMS Topic Subscriber or Wait for JMS Queue Message). Specify an integer as the value of the property to determine the number of seconds to set the default polling interval for all JMS activities that receive messages. Individual activities can override this default polling interval by specifying a value in the Receiver Timeout field on the Advanced tab of the activity. For more information about the activities that have the Receiver Timeout field, see *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

bw.plugin.jms.recoverOnStartupError

When a process engine attempts to startup and the JMS server that JMS activities connect to is not up, the JMS process starters cannot connect to the JMS server. Setting this property to true allows the process engine to start and the JMS process starters will wait until the JMS sever is up before starting.

Mail Properties

This section describes custom engine properties that can be set for the resources in the Mail palette.

bw.plugin.mail.receiverFlattenNesteedAttachments

In previous releases, the **Receive Mail** activity threw exceptions when receiving email, if the email was in rich text format and the any mime part contained nested mime sub-parts. You can fix this by setting this property to `true` which creates a flat output structure where all sub-parts are siblings. For example, the following nested structure:

```
<mimeEnvelopeElement>
   <mimePart>
      <mimePart>
         <textContent />
      </mimePart>
   </mimePart>
</mimeEnvelopeElement>
```

would be flattened out to the following:

```
<mimeEnvelopeElement>
   <mimePart>
      <textContent />
   </mimePart>
</mimeEnvelopeElement>
```

If you rely on the behavior of previous releases, keep this property set to its default value of `false`.

bw.plugin.mail.receiverHandleDiscreteTypes

In previous releases, the Receive Mail activity did not handle incoming mime messages with mime types application/*, audio/*, video/*, or image/*. While fetching these types of emails, ActiveMatrix BusinessWorks threw exceptions. You can fix this by setting this property to `true`. If you rely on the behavior of previous releases, keep this property set to its default value of `false`.

bw.plugin.mail.receiverRetryCount

When a mail sender is in the process of sending a message, the mail server may expose the message to the Receive Mail process starter, but indicate later that the message is unavailable. This typically occurs when sending large messages. The Receive Mail process starter attempts to receive the message during subsequent polls of the mail server. By default, the process starter will attempt to receive the message for three minutes. The number of retries within that three-minute limit depends upon the value of the polling interval. For example, if the polling interval is set to 30 seconds, there will be up to six retries. If the polling interval is set to 4 minutes, there will be only one retry.

This property allows you to specify the number of times the Receive Mail process starter will attempt to receive the same message. The amount of time allotted for retries will be the value of this property multiplied by the polling interval. For example, if the polling interval is every 10 seconds, and the retry count is set to 12, then the Receive Mail process starter will attempt to receive the message for two minutes.

Rendezvous Properties

This section applies to activities that can use TIBCO Rendezvous transports. This includes activities in the **Rendezvous** palette and the **ActiveEnterprise Adapter** palette.

Config.Tibrv.cmQueueTransport.TaskBacklogLimitInBytes

When the RVCMQ transport is used, ActiveMatrix BusinessWorks applies the value of this property to the RVCMQ transport using the RVCMQ API `setTaskBacklogLimitInBytes()` method to set the scheduler task queue limits in bytes for the distributed queue transport. Ffor more information about this method, see the TIBCO Rendezvous documentation. The value of this property must be set to a positive integer.

Transaction Properties

This section describes custom engine properties that can be set for the resources in the **Transaction** palette.

bw.plugin.transaction.xa.isolation

By default in an XA transaction, the transaction isolation level is set to the default value for the JDBC driver you are using. If you wish to ensure a particular transaction isolation level, set the `bw.plugin.transaction.xa.isolation` custom engine property to one of the following values:

Value	Transaction Isolation Level Description
1	java.sql.Connection.TRANSACTION_READ_UNCOMMITTED
2	java.sql.Connection.TRANSACTION_READ_COMMITTED
3	java.sql.Connection.TRANSACTION_REPEATABLE_READ
4	java.sql.Connection.TRANSACTION_SERIALIZABLE

bw.plugin.transaction.xa.lock.connection

By default, JDBC activities in an XA Transaction groups obtain database connections from a connection pool and release the connections when the activity completes. This can cause a database connection to be used concurrently in multiple transactions. Some databases or JDBC drivers support this behavior and others do not. If you are using a database or JDBC driver that requires database connections to be used in only one transaction at a time (for example, IBM DB2), set the `bw.plugin.transaction.xa.lock.connection` custom engine property to `true`. When the value of this property is set to `true`, once a connection is associated with a transaction, the connection remains associated with the transaction until the transaction completes. The default value of this property is `false`.

TCP Properties

This section describes custom engine properties that can be set for the resources in the **TCP** palette.

TCPRead.ThreadCount

Through this property the TCPRead Activity can be configured for any number of threads.

bw.plugin.tcp.server.acceptCount

This property specifies the maximum number of incoming requests that can be handled by the TCP Server. The default value for this property is 50.

Properties for Backwards Compatibility

From time to time, functional behavior of ActiveMatrix BusinessWorks changes. If you rely on the behavior of previous releases, there are properties that allow you to revert to the behavior of previous releases. This section lists properties that are included for backwards compatibility with projects created in previous versions.

While properties in this section can be used to revert to behavior of previous releases, use of these properties is not recommended for most circumstances. Functionality changes are usually introduced to improve the product or to correct erroneous behavior. Therefore, relying on the behavior of previous releases is not recommended for new projects.

The properties in this section are intended to allow backward compatibility of legacy projects until the project can be corrected to accommodate the new behavior. These properties are not intended for long-term use.

bw.plugin.ftp.stripLineFeedInPut

Prior to release 5.2.0, the **FTP Put** activity stripped the \n when \r\n was used for a new line in a file. This caused files to be unusable when a file was taken from a MS Windows machine and put onto a VMS machine. The **FTP Put** activity no longer strips the \n, and if you rely on this behavior in existing projects, you can set the `bw.plugin.ftp.stripLineFeedInPut` to true to obtain the behavior of previous releases.

bw.plugin.http.client.urlEncodeQueryString

As of release 5.2.0, the QueryString input element of the **Send HTTP Request** activity is not automatically URL encoded. Prior to release 5.2, the activity used URL encoding for the Query specified in the QueryString element. It is now the user's responsibility to properly URL-encode the query specified in the QueryString. Therefore, the activity does attempt to encode the value supplied in this element. This change may cause backward compatibility issues if you rely on the activity to perform the URL-encoding of the QueryString. This property is set to false by default, but setting it to true reverts to the behavior of previous releases.

bw.plugin.javaCode.explicitNull

To indicate a null reference, the **Java Code** activity omits the value in its output. This causes a String value used as a null place holder when another activity attempts to read the null in its input. However, other activities did not behave in this way. Other activities pass an explicit null for null references.

To preserve backward compatibility, the **Java Code** activity still behaves the same. However, you can set the `bw.plugin.javaCode.explicitNull` to `true` to cause the **Java Code** activity to behave in the same way as other activities. When this property is set to `true`, an explicit null is set for a null reference. This property is set to `false` by default, maintaining the behavior of the previous releases.

bw.plugin.parseData.enforceLineLength

In previous releases, the line length specified in the **Data Format** resource was not enforced. This allowed files with one large line to be parsed in some situations. In more recent releases, the line length is enforced so that files containing one large line are no longer allowed. If you rely on the behavior of the previous releases, set the `bw.plugin.parseData.enforceLineLength` property to `false`. By default, this property is set to `true`.

bw.plugin.timer.useJavaMonth

In previous releases, the **Timer** process starter used the Java convention (0-11) for month numbers in its output, however, the expected convention for month numbers is 1-12. In release 5.2.0, the month is returned as a number between 1 and 12. If you rely on the behavior of previous releases, you can set this property to `true` to maintain compatibility with previous releases.

com.tibco.plugin.soap.no_xsi_type

SOAP activities were enhanced in release 2.0.5 to emit `xsi:type` attributes. If you wish to maintain backward compatibility and not emit these attributes, you must set this property to `true`.

com.tibco.xml.xpath.create-dateTime.has.timezone

In Release 2.x, the XPath function `create-dateTime()` returned a value that included a time zone. In Release 5.1.2 and 5.1.3, the function was changed to omit the time zone. This property controls whether the time zone is included in the output of the `create-dateTime()` function. Setting this property to false (the default value) omits the time zone from the function output (the same behavior as 5.1.x). Setting this property to true causes the time zone to be included in the function output (the same behavior as 2.x).

Config.JDBC.CallProcedure.InputOptional

In releases prior to 5.2.0, the **JDBC Call Procedure** activity created input elements that were optional for stored procedure parameters. Optional parameters have never been supported by this activity. See the Known Issues list under the JDBC Palette heading in *TIBCO ActiveMatrix BusinessWorks™ Release Notes*. When migrating a project from a previous release, there will be validation errors for any unspecified input elements for stored procedure parameters. These migrated projects cannot be executed until the errors are resolved by using the **Mapper Check and Repair** button on the **Input** tab.

If you wish to migrate a project without fixing this problem, you can do so by setting this property to true.

Config.JDBC.CallProcedure.OutputUseNil

Prior to release 5.1.2, if a value returned from a database table was null, the output element corresponding to that table value was not placed into the output schema for a **JDBC Call Procedure** activity, if the output element was optional. The element is now placed into the output schema and has "`xsi:nil = true`" to indicate the element is null.

You should surround elements that can be nil with an if statement to determine whether to output the element. To maintain the behavior of previous releases, this property controls whether elements that are nil are contained in the output. Set the property to `false` to achieve the behavior of previous releases.

ignore.delimiters.under.quotes

Prior to Release 2.0.4, when using the activities in the Parse palette, delimiter-separated data was not treated in a standard way. There was no mechanism to escape the specified delimiter character. For example, if you chose a comma as the delimiter, there was no way to have a field contain a comma as in "`Fresno, CA`". Also, there was no way to have a field span multiple lines or include leading and trailing spaces.

Now fields can be surrounded in double quotes. For more information about the new semantics for parsing input text, see the description of the **Data Format** shared configuration resource in *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*.

To disable this functionality, set the value of this property to `true`.

java.property.DiscardUTF8BOM

When a file is saved on a Windows platform using UTF-8 encoding, Windows adds a Byte Order Mark (BOM) to the beginning of the file. This BOM is not necessary for UTF-8, but it is valid. Prior to release 2.0.6, the File Reader activity's output includes the BOM at the beginning of the data read from the file.

The BOM is now stripped when it is encountered. If you wish to retain the functionality of previous releases, you can set this property to false. In most cases, you will not need to set this property. You may need to set this property to true if your process definition is expecting a file that contains the BOM.

java.property.com.tibco.schema.ae.makeNillable

Certain TIBCO ActiveEnterprise-based schema elements do not display as nillable in the Input mapping tab. This can result in mappings (optional to optional) that do not copy the `xsi:nil` attributes at runtime to the output elements, and subsequently validation errors.

Setting this property to `true` causes mappings that meet the criteria to show warnings. Selecting the input mapping with an error and clicking the **Mapper Check and Repair** displays yellow warnings: "The input and this element are both nillable, set to copy-nil". Clicking OK changes the mappings to add the copy-of for the nil attribute ("Optional and nillable to optional and nillable"). This is generally a better way to map this structure and ensures if the element in the source data has the `xsi:nil` attribute, it will be copied to the target element.

In Release 5.2.1 and subsequent releases, the default setting for this property is `true`, which may cause new warnings to appear in existing projects. Typically, the Mapper Check and repair button can be used to update the mappings to copy `xsi:nil` attributes. If it is preferable to have empty elements emitted in this case, then the property can be set to `false`.

Any new mapping done by drag-and-drop with the property set to `true` will have the "Optional and nillable" style mapping, instead of the "optional to optional" style.

Performance Tuning

This section provides an overview of tuning the ActiveMatrix BusinessWorks environment to optimize performance. It also describes how altering certain parameters can affect the system performance.

Overview of Engine Performance

The heart of ActiveMatrix BusinessWorks is the ActiveMatrix BusinessWorks Engine.

The ActiveMatrix BusinessWorks engine handles a continuous stream of processes, each with a number of activities, in an operating environment with finite resources. Apart from scheduling process instances (jobs), the ActiveMatrix BusinessWorks engine also performs other functions like data validation, connection management, flow control, job recovery, logging, managing, and monitoring services.

In an enterprise world, the factors that influence performance range from physical hardware resources to individual steps in a process diagram. These factors are discussed under various sections in the ActiveMatrix BusinessWorks documentation set. Categories of Performance Factors for ActiveMatrix BusinessWorks, lists some common ActiveMatrix BusinessWorks performance factors and references to the relevant sections in the ActiveMatrix BusinessWorks documentation other than the current section.

Categories of Performance Factors for ActiveMatrix BusinessWorks

Category	Examples	Reference, if any
Hardware	CPU, Memory and Disk resources	Setting Deployment Options
Java	JVM and JVM configuration	Setting Deployment Options
Engine	Number of engines, number of threads, job creators, flow control, job pool, etc.	Setting Deployment Options
Job /	Job size	None

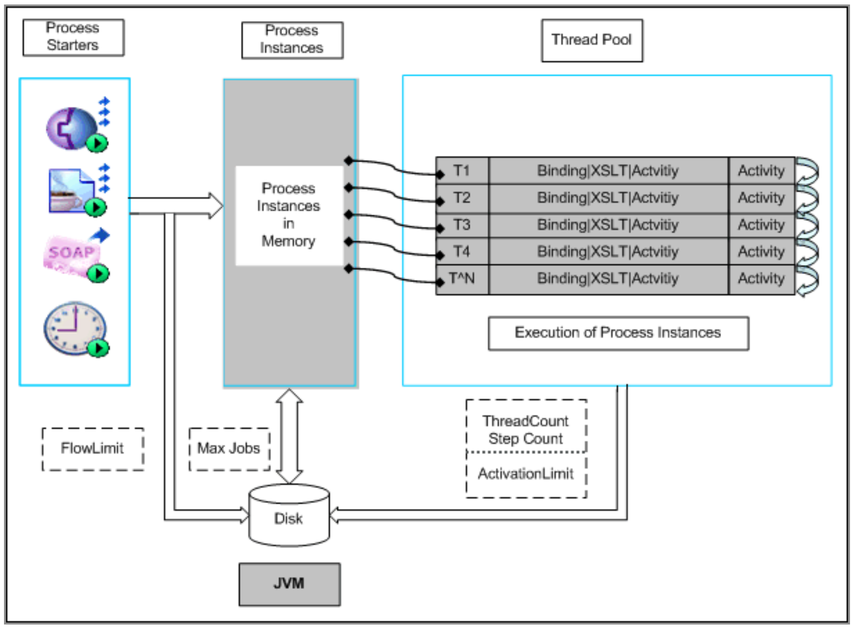
Category	Examples	Reference, if any
Message		
Process Design	User scripts, sub-processes, inline processes, checkpoints, logging activities	ActiveMatrix BusinessWorks Process Design Guide
Other Software	External software, like relational DB, other TIBCO Software product	Product documents provided with the software

Components Affecting the Engine Performance

This section gives an overview of the message flow architecture and the factors that affect the performance of the ActiveMatrix BusinessWorks engine.

Process Execution in BusinessWorks Engine, illustrates the way process instances are processed in ActiveMatrix BusinessWorks and the various factors that affect the performance of the engine.

Figure 6: Process Execution in BusinessWorks Engine



The process instances created are typically held in memory. However, this may not be the case if the following parameters are set:

`Max Jobs`: If the number of process instances in memory have reached the value of `Max Jobs`, then the process instances created are temporarily held on a disk. These process instances will be moved back into memory when sufficient memory is available.

`Flow Limit`: When set, this property limits the number of process instances that can be created. If the number of process instances being created exceeds the value of `FlowLimit`, the engine suspends the creation of new process instances. However, it continues executing the process instances in memory. The engine resumes creating new process instances when process instances, approximately half the value specified for `FlowLimit`, have completed. For more information about `FlowLimit` property, see *Controlling Execution of ActiveMatrix BusinessWorks Services* in *Setting Deployment Options* .

The number of process instances that can be created in memory is also limited by the memory available on the machine and the memory allocated to the JVM on which the process engine executes.

The process instances in memory are executed by the ActiveMatrix BusinessWorks engine. The number of process instances that can be executed concurrently by the engine is limited by the maximum number of threads, specified by property `ThreadCount`. Threads execute a finite number of tasks or activities uninterrupted and then yield to the next process instance that is ready.

The engine property `StepCount` determines the number of tasks that are executed by a thread uninterruptedly. However, the exceptions to `StepCount` occur when the job is blocked or in a transaction. When a job is in a transaction, the thread will not be released even if the `StepCount` is exceeded. However, if a process instance is waiting or is in a blocked state, it can be paged out and the freed memory used to process another process instance.

`Activation Limit` can be set if a process instance that is blocked should remain in memory till completion. Setting the `ActivationLimit` affects the engine performance substantially.

For more information on setting these properties, see Custom Engine Properties.

In addition to the components above, the performance of the ActiveMatrix BusinessWorks engine is also affected by external factors such as

- rate of incoming messages,
- network latency,
- performance of external applications with whom BW processes communicate, and

- other OS processes that may be running on the system.

Characteristics of ActiveMatrix BusinessWorks Activities

The activities in ActiveMatrix BusinessWorks can be grouped as blocking and non-blocking, based on how each activity works with threads.

Consider an activity being executed by an engine thread. For the time the activity is being executed, the resources are being used by it. However, when an activity is waiting for an event, or executes the `Sleep` command, one of the following can happen:

- The activity continues to execute on the same thread and affects the performance of the engine. Such an activity is called a blocking activity.
- A thread switch occurs and the activity continues to execute on its private thread. When the activity is ready again, it is picked up by an engine thread available at that time. These activities do not affect the performance of the engine. Such an activity is called a non-blocking activity.

Process starters such as, **HTTP Receiver**, **JMS Receiver** and activities like **WaitFor** and **Sleep**, are all non-blocking activities. They listen for incoming events on their private threads. However, activities like **JMS Sender** and **JDBC Query** are blocking in nature. These activities execute on the engine thread and do not work on private threads.

For a list of activities and the threads on which they work, see Thread Based Grouping of Activities.

Performance Considerations

This section describes the memory and throughput considerations to be made, while tuning factors affecting the performance of a ActiveMatrix BusinessWorks engine.

Memory Considerations

Engine parameters

Max jobs

If the number of process instances in memory have reached the value of `Max Jobs`, then the process instances created are temporarily held on a disk. These process instances are moved back into memory when sufficient memory is available.

Flow limit

When the job processing starts lagging, the memory allocated to the job pool is utilized by the new jobs created and can exhaust the job pool. The `FlowLimit` property can be set to specify the number of jobs that can reside in the job pool.

Once the job pool is full, the job creator is suspended and is in a `FLOW_CONTROLLED` state. The job creator comes out of the `FLOW_CONTROLLED` state after approximately half the value specified for `FlowLimit` jobs are executed to completion.

Activation limit

`Activation Limit` can be set if a job that is blocked should remain in memory till completion. Setting the `ActivationLimit` affects the engine performance substantially.

JVM parameters

Every ActiveMatrix BusinessWorks engine runs in a Java Virtual Machine. As a result, the settings on the JVM influence the engine performance. Java provides some parameters to tune the memory usage and optimize engine performance.

Heap size

Following parameters are used to set the heap size for the engine:

`-Xms`: Minimum amount of memory used

`-Xmx`: Maximum amount of memory used

ActiveMatrix BusinessWorks 5.2.0 and higher versions may see an increase in memory footprint and a potential memory sizing issue. This is because ActiveMatrix BusinessWorks 5.2.0 and above point to server JVM, while versions older than ActiveMatrix BusinessWorks 5.2.0 point to client JVM in the `bwengine.tra` file. While sizing and tuning your environment, consider the following:

- The total memory used by the JVM (memory footprint) is dependent on the type of JVM used (Server JVM or Client JVM), the JVM version, and the JVM vendor.
- A Server JVM occupies a higher memory footprint and may result in higher performance when compared to a Client JVM.
- A Client JVM may have a lower startup time and memory footprint.

For more information about tuning the JVM to suit your application, refer to the JVM Tuning Guide of the JVM used in your environment. You may also consult your JVM vendor for details about the exact memory footprint and heap management in the JVM version used in your environment.

Garbage collection

The java object, such as a job, occupies memory from the time it is created to the time it is destroyed. Java provides garbage collection, an automated mechanism to clean up objects that still exist but are no longer used.

To retrieve the garbage collection metrics for the ActiveMatrix BusinessWorks engine, specify the `-verbose:gc` option when starting the JVM used by the engine.

You can set the Java memory parameters by using the `java.extended.properties` option in the `bwengine.tra` file.

For example, specify the following in the `bwengine.tra` to set the heap size to 768M and retrieve the garbage collection metrics for the engine.

```
java.extended.properties -Xms768m -Xmx768m -verbose:gc
```

Throughput Considerations

Throughput of the ActiveMatrix BusinessWorks engine is the rate at which the engine can execute and complete processes. The throughput of the engine is determined by factors that can be grouped into engine parameters and HTTP parameters.

Engine parameters

Thread Count

`ThreadCount` specifies the number of process instances (jobs) that can be executed concurrently by ActiveMatrix BusinessWorks engine. By default, the thread count is eight.

TIBCO recommends that you measure the CPU and memory resources under a typical processing load to determine if the default value is suitable for your environment.

If the engine throughput has reached a plateau, but the CPU and memory are not fully utilized, you can increase the thread count to have a positive effect on the throughput. However, specifying too high value can cause CPU thrashing, or an increase in latency caused by a large number of messages in the queue. Specifying too low value can cause higher memory use and lower engine throughput as some CPU resources remain unutilized.

Each process instance consists of multiple activities that have to be executed in a sequence for the process instance to complete. If one of these activities is a blocking activity, the thread that is executing this process instance is idle and resources are under-utilized. This affects the engine's throughput.

Step Count

You can specify the parameter, `Engine.StepCount`, to control the maximum number of steps executed (unless the job is in a transaction) for a job after which the thread yields to another job ready in the job pool. A low value of `StepCount` can degrade the engine performance due to frequent thread switches. A high value of `StepCount` may cause less concurrency in executing jobs and hence, result in an inefficient use of CPU.

For more details about `Engine.StepCount`, see Available Custom Engine Properties .

HTTP specific parameters

minProcessors

Specifies the number of threads created when the HTTP Server is started. These threads process HTTP requests. The default value of this property is 10.

Setting a high value for `minProcessors` can produce a large number of excessive threads and hence block the system resources.

maxProcessors

Specifies the maximum number of threads that can be created by the HTTP Server to process incoming HTTP requests. The default value of this property is 75.

Setting a low value for `maxProcessors` results in the following:

- limiting the number of threads in the system and therefore limiting the number of requests that can be processed simultaneously.
- reduced memory contention.
- less number of context switches.
- increased throughput.

Setting a high value for `maxProcessors` results in the following:

- more number of requests can be processed simultaneously.
- degraded throughput for all the processes.

acceptCount

Specifies the maximum number of incoming connection requests that can be accepted when all HTTP processors are in use. Incoming requests received after the `acceptCount` limit is reached are rejected. The default value for `acceptCount` is 100.

When `maxProcessors` is low, you may wish to set `acceptCount` to a higher value so that more client requests are accepted and queued, rather than being rejected.

Enable DNS Lookups

Checking this field enables a Domain Name System (DNS) lookup for HTTP clients so that the IP address is resolved to a DNS name.

Setting this field adds latency to every request because it requires the DNS lookup to complete before the request can be completed. Since the throughput is adversely affected, this field should be set only when required.

Processor Affinity

On a multi-CPU machine, process(es) can be configured to run only on a dedicated set of CPUs using processor affinity.

Processor affinity takes advantage of the fact that some remnants of a process may remain in one processor's state (in particular, in its cache) from the last time the process ran. Scheduling it to run on the same processor the next time could result in the process running more efficiently than if it were to run on another processor.

JDBC Activities

Every **JDBC Connection** shared resource has a connection pool. The parameter `Maximum Connections` determines the connection pool size and hence, the maximum number of connection requests that can be processed by a JDBC shared resource.

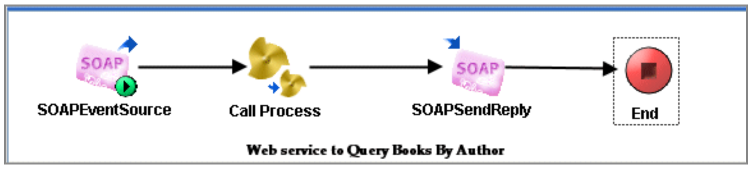
JDBC activities use engine threads to connect to a database configured in the **JDBC Connection** shared resource and process requests. The engine threads are released once the operation in the JDBC activity completes and the connection is closed.

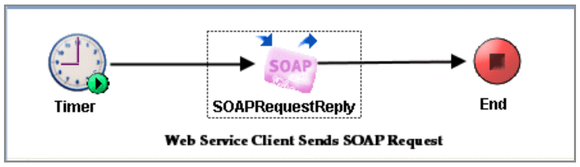
For a typical scenario using a single **JDBC Connection** shared resource, ensure that the parameter `Maximum Connections` specified is inline with the engine thread pool size specified by the property `engine.ThreadCount`.

Sample Scenario

Consider a web service that queries for books in a bookstore by the name of their Author. A client for the web service is configured using the service’s WSDL file. The client sends a SOAP request containing the name of an Author to the web service and receives a list of books.

See the example at *<TIBCO_HOME>*`\bw\`*<version_number>*`\examples\activities\soap\soap_over_http` for more details of the sample scenario.





Consider the web service is running on a two CPU Windows machine with 3GHz and 2G RAM.

When the number of concurrent SOAP requests sent to the web service increases, the ActiveMatrix BusinessWorks engine is loaded and the memory required increases.

The JVM heap size determines the memory allocated for the ActiveMatrix BusinessWorks engine and processes instances. To specify the JVM heap size, set the following property in `bwengine.tra`:

```
java.extended.properties -Xms768m -Xmx768m
```

Note: The JVM heap size can also be set in TIBCO Administrator. For more information, see Server Settings Tab .

The memory required for a ActiveMatrix BusinessWorks engine is defined by the workload that the engine is designed to handle.

To specify the number of concurrent incoming HTTP requests that can be handled by the web service, set the following properties in the `bwengine.tra` file:

```
bw.plugin.http.server.minProcessors
```

```
bw.plugin.http.server.maxProcessors
```

Setting `maxProcessors` to 100, ensures that upto 100 requests can be accepted concurrently.

To improve the ability to execute process instances concurrently, more engine threads are required. The number of engine threads to be allocated can be set using the property `Engine.ThreadCount`. The default value is eight. Theoretically, the value of `Engine.ThreadCount` can be increased till maximum CPU utilization is reached. However, an increase in the thread count and hence, an improved concurrency, may not always result in an improved performance. This is because an improved concurrency also implies an increased resource consumption.

Set the `Engine.StepCount` property to specify the maximum number of execution steps for a job, unless in a transaction or when the ActivationLimit is set. The default value is 20. Depending on the nature of the jobs being executed, the value of this property can be increased or decreased. A low value of StepCount results in frequent thread switches. This is an overhead, especially when the number of execution steps for most jobs is high.

To specify the size of the thread pool used by the Request-Reply activity on the web service client side, set the properties: `bw.plugin.http.client.ResponseThreadPool`

`bw.plugin.http.client.ResponseThreadPool.type`

As the thread pool is created when the engine starts, use a reasonable number to specify the size of the ResponseThreadPool for your system. A high value results in extra resources being allocated which may never be used.

TIBCO Hawk MicroAgent Methods

TIBCO Administrator is the preferred monitoring and management application for ActiveMatrix BusinessWorks. However, the process engine is instrumented with a TIBCO Hawk microagent that can be used to perform most administrative functions. This appendix describes the microagent methods available for the ActiveMatrix BusinessWorks process engine.

Enabling TIBCO Hawk

Before using the TIBCO Hawk with ActiveMatrix BusinessWorks, you must enable the TIBCO Hawk microagent in the process engine. To do this, set the `Hawk.Enabled` property to `true`. This can only be set for deployed process engines. Process engines in the test environment are not normally monitored and administered, therefore TIBCO Hawk is not recommended for use in the testing environment.

If you are using non-default transport parameters for TIBCO Hawk, you must also set the `Hawk.Service`, `Hawk.Network`, and `Hawk.Daemon` properties to the values for the transport you are using.

Some microagent methods require memory and processor overhead for gathering statistics or for getting information on the current state of the process. Because of the performance implications, certain instrumentation is disabled by default. You can enable instrumentation for a specific process definition with the `Instrumentation.`<*processName*> property. To enable instrumentation for all processes, use the `Instrumentation.*` property. Enabling instrumentation can lead to significant performance degradation. You should only enable instrumentation for brief periods while testing performance.

For more information about setting properties for process engines, see Custom Engine Properties.

TIBCO Hawk Microagent Methods

This section describes the TIBCO Hawk microagent methods for the ActiveMatrix BusinessWorks process engine.

ConfigureActivityTracing

Description:

Enables or disables tracing for specified activity. Activities can be specified by process definition and activity name or by class name. The class name is the Java implementation class' name.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition for which you wish to configure tracing. Specifying `*` signifies all process definitions. If ActivityClass is specified, this argument is optional.
Activity	Name of the activity for which you wish to configure tracing. Specifying `*` signifies all activities. If ActivityClass is specified, this argument is optional.
ActivityClass	Java implementation class name of the activity for which you wish to configure tracing. This argument is optional if you specify the ProcessDefinition and Activity arguments.
Enable	Specify `true` if you wish to enable tracing, `false` if you wish to disable tracing.

Output:

None

ConfigureAllTracing

Description:

Controls tracing for all activities and process starters.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
EnableAllActivities	If `true`, tracing is enabled for all activities. If `false`, tracing will be disabled for all activities.
EnableAllStarters	If `true`, tracing is enabled for all process starters. If `false`, tracing will be disabled for all process starters.
EnableAllUserRoles	If `true`, tracing is enabled for all user-defined roles. If `false`, tracing will be disabled for all user-defined roles.

Output:

None

ConfigureProcessStarterTracing

Description:

Enables or disables tracing for specified process starter. Process starters can be specified by name or class name. The class name is the Java implementation class' name.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessStarter	Name of the process starter for which you wish to configure tracing. Specifying `*` signifies all process starters. If StarterClass is specified, this argument is optional.
StarterClass	Java implementation class name of the process starter for which you wish to configure tracing. This argument is optional if you specify the ProcessStarter argument.
Enable	Specify `true` if you wish to enable tracing, `false` if you wish to disable tracing.

Output:

None

ConfigureRole

Description:

Enables or disables tracing for the specified role.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Role	Name of the role for which you wish to configure tracing. Specifying `*` signifies all roles.
Enable	Specify `true` if you wish to enable tracing, `false` if you wish to disable tracing.

Output:

None

ConfigureUserDefinedTracing

Description:

Enables or disables tracing for the specified user-defined role.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Role	Name of the user-defined role for which you wish to configure tracing. Specifying `*` signifies all user-defined roles.
Enable	Specify `true` if you wish to enable tracing, `false` if you wish to disable tracing.

Output:

None

DelayedStopApplicationInstance

Description:

Instructs all process starters to stop further job creation but stay active. The engine shuts down after all process instances have completed or the specified maximum delay has been reached. After shutdown, any remaining checkpoint files are preserved and the engine's operating system process exits.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
MaxDelay	Specifies the amount of time (in seconds) to wait before shutting down the process engine.
WaitForCheckpoints	When true is specified, the engine waits for any checkpointed process instances to complete before shutting down.

Output:

None

GetActivities

Description:

Retrieves information about the activities that have been executed for a given process definition since the engine was started. The activity information is cumulative. A single activity name represents all executions of that activity. The min/max fields can be reset with the ResetActivityStats method.

The ExecutionTime computation for the Call Process Activity includes the sum of the execution times for all activities in the called process, not just the execution time for the call process activity itself.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition.

Output:

The following table describes the output of this microagent method:

Column Name	Description
ProcessDefName	Name of the process definition.
Name	Name of the activity.
ActivityClass	Name of the class that implements the activity.
ExecutionCount	Number of times the activity has been executed.
ElapsedTime	Total clock time (in milliseconds) used by all executions of this activity. This includes waiting time for Sleep, Call Process, and Wait For... activities.
ExecutionTime	Total clock time (in milliseconds) used by all executions of this activity. This does not include waiting time for Sleep, Call Process, and Wait For... activities.
ErrorCount	Total number of executions of the activity that have returned an error.
LastReturnCode	Status code returned by most recent execution of this activity. This can be either OK, DEAD, or ERROR.
Tracing	True if tracing is enabled for this activity, false if tracing is disabled.

Column Name	Description
MinElapsedTime	Elapsed clock time (in milliseconds) of the activity execution that has completed in the shortest amount of elapsed time.
MaxElapsedTime	Elapsed clock time (in milliseconds) of the activity execution that has completed in the longest amount of elapsed time.
MinExecutionTime	Execution time (in milliseconds) of the activity execution that has completed in the shortest amount of execution time.
MaxExecutionTime	Execution time (in milliseconds) of the activity execution that has completed in the longest amount of execution time.
MostRecentElapsedTime	Elapsed clock time (in milliseconds) of the most recently completed activity execution.
MostRecentExecutionTime	Execution time (in milliseconds) of the most recently completed activity execution.
TimeSinceLastUpdate	Time (in milliseconds) since the statistics have been updated.
CalledProcessDefs	A comma-separated list of names of process definitions called by this activity.
ExecutionCountSinceReset	Number of activity executions that have completed since the last reset of the statistics.

GetExecInfo

Description:

Retrieves the process engine execution information.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Status	Engine status. Can be one of the following: • ACTIVE • SUSPENDED • STANDBY • STOPPING
Uptime	Elapsed time (in milliseconds) since the process engine was started.
Threads	Number of worker threads used by the process engine.
Version	Version of the process engine.

getHostInformation

Description:

Returns the value of the specified property on the host machine on which the process engine is running.

Method Arguments:

he following table describes the arguments of this microagent method:

Argument Name	Description
Name	Name of the property to return. Leave this

Argument Name	Description
	argument blank to return all properties. The following are the properties that can be returned: • Application Instance — is the name of the project that is running in the process engine. • Application Name — TIBCO Hawk display name of the process engine. This is set by the engine property Hawk.AMI.DisplayName. • Application State — state of the process engine. Can be UNINITIALIZED, INITIALIZING, RUNNING, STOPPING, or STOPPED

Output:

The following table describes the output of this microagent method:

Column Name	Description
Name	Name of the property returned.
Value	Value of the property.

GetMemoryUsage

Description:

Retrieves information about the process engine’s memory usage.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
TotalBytes	Total number of bytes allocated to the process engine.
FreeBytes	Total number of bytes that are not currently in use.
UsedBytes	Total number of bytes that are currently in use.
PercentUsed	Percentage of total bytes that are in use.

GetProcessCount

Description:

Returns the total number of running process instances.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
TotalRunningProcesses	Total number of currently executing process instances.

GetProcessDefinitions

Description:

Retrieves information about process definitions.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Name	Name of the process definition.
Starter	Name of the process starter for the process.
Created	Number of process instances created for this process definition.
Suspended	Number of times process instances have been suspended.
Swapped	Number of times process instances have been swapped to disk.
Queued	Number of times process instances have been queued for execution.
Aborted	Number of times process instances have been aborted.
Completed	Number of process instances that have been successfully completed.
Checkpointed	Number of times process instances have executed a checkpoint.

Column Name	Description
TotalExecution	Total execution time (in milliseconds) for all successfully completed process instances.
AverageExecution	Average execution time (in milliseconds) for all successfully completed process instances.
TotalElapsed	Total elapsed time (in milliseconds) for all successfully completed process instances.
AverageElapsed	Average elapsed clock time (in milliseconds) for all successfully completed process instances.
MinElapsed	Elapsed clock time (in milliseconds) of the process instance that has completed in the shortest amount of elapsed time.
MaxElapsed	Elapsed clock time (in milliseconds) of the process instance that has completed in the longest amount of elapsed time.
MinExecution	Execution time (in milliseconds) of the process instance that has completed in the shortest amount of execution time.
MaxExecution	Execution time (in milliseconds) of the process instance that has completed in the longest amount of execution time.
MostRecentExecutionTime	Execution time (in milliseconds) of the most recently completed process instance.
MostRecentElapsedTime	Elapsed clock time (in milliseconds) of the most recently completed process instance.
TimeSinceLastUpdate	Time (in milliseconds) since the statistics have been updated.
CountSinceReset	Number of process instances that have completed since the last reset of the statistics.

GetProcesses

Description:

Retrieves information about active process instances. If arguments are specified, information for process instances that match the specified arguments is returned.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Id	ID for the process instance.
Name	Name of the process definition used by the process instance.
EarliestStartTime	Earliest time (in milliseconds) at which the process instance started. All process instances started after the specified time will be retrieved.
MinimumDuration	Minimum time (in milliseconds) in elapsed clock time since the process instance started. All process instances that have elapsed times greater than the specified minimum duration will be retrieved.
MainProcessName	Name of the main process definition.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Id	ID for the process instance.
Name	Name of the process definition used by the process instance.

Column Name	Description
TrackingId	Tracking ID for the process instance.
CustomId	Custom ID for the process instance.
Status	Status of the process.
StartTime	Time when the process instance started.
Duration	Elapsed clock time (in milliseconds) since the process instance started.
MainProcessName	Name of the main process definition.
CurrentActivityName	Name of the currently executing activity in the process instance.
StarterName	Name of the process starter that started this process instance.
SubProcessName	Name of the process definition for the sub-process.

GetProcessesExceptions

Description:

Retrieves error information reported by the specified process.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Id	ID for the process instance. If not specified, or if 0 is specified, exceptions for all process instances are returned.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Seq	Sequence number of the exception, with the most recent exception first.
Id	ID for the process instance.
Message	Exception message.
StackTrace	Exception stack trace.
ExceptionClass	Exception class name.
ProcessStack	Process stack at exception. This displays the [ProcessName/GroupName/ActivityName] of the activity issuing the exception. If the activity is in a called sub-process, then the calling activity's process stack plus a '>' separator character will be pre-pended to the normal information to produce the process stack of the activity issuing the exception.
TrackingId	Tracking ID for the process instance.
ProcessDef	Name of the process definition.
State	State of the process.

GetProcessStarters

Description:

Retrieves information about either active or inactive process starters. The information is cumulative. A single process starter name represents all executions of that process starter.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ActiveOrInactive	Specify `Active` to retrieve information about process starters with the ACTIVE or READY status. Specify `Inactive` to retrieve information about process starters with the INACTIVE status.

Output:

The following table describes the output of this microagent method:

Column Name	Description
ProcessDef	Name of the process definition.
Name	Name of the process starter.
Status	Status of the process starter. The status can be INACTIVE, ACTIVE, or READY.
Created	Number of process instances created by this process starter.
CreationRate	Number of process instances per hour created by this process starter.
Running	Number of process instances currently executing.
Completed	Number of process instances that have completed.
StartTime	Time (in milliseconds) at which the process starter was started.
Duration	Elapsed clock time since the process starter was started.
CheckpointedStart	True if the process was restarted from a checkpoint.
Tracing	True if tracing is enabled for this process starter, false if tracing is disabled.

GetRecoverableProcesses

Description:

Retrieves the process instances that can be recovered. For more information about recoverable process instances, see Managing Recoverable Process Instances. Use the returned process instance ID in the RestartRecoverableProcess or RemoveRecoverableProcess commands.

Method Arguments:

None.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Id	ID for the process instance.
Status	Status of the process instance. Can be one of the following: • faulted — this status occurs when the process instance is terminated due to an unhandled exception. • interrupted — this status occurs when the process instance is terminated due to engine failure (and the instance did not automatically restart when the engine restarted) or a manual termination.
TrackingId	Tracking ID for the process instance.
CustomId	Custom ID for the process instance.
ProcessName	Process definition for this process instance.
RestartActivity	Name of the last executed Checkpoint activity in the process instance. This is the point at which the process instance will begin executing when it is restarted.

getRvCmConfig

Description:

Get information about all the TIBCO Rendezvous certified message transports.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
certifiedName	Name used for certified delivery.
service	Service parameter for the Rendezvous daemon.
daemon	Daemon parameter for the Rendezvous daemon.
network	Network parameter for the Rendezvous daemon.
ledgerFile	Name of the ledger file for the Rendezvous daemon.
cmTimeout	Timeout value for certified delivery.

GetStaticActivityInfo

Description:

Retrieves design time activity information for all activities in a given process definition.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Name	Name of the activity as specified in TIBCO Designer.
Type	A Java class name, for the type of the activity. For example, com.tibco.pe.core.CallProcessActivity.

GetStatsCollectorDestination

Description:

Retrieves the file name of the file where activity statistics are currently being collected. See Storing Process Instance and Activity Statistics for more information about collecting activity statistics.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
fileName	Name of the file where activity statistics are currently collected. This value is blank if statistics are not currently being collected.

GetStatus

Description:

Retrieves basic status information about the engine.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
InstanceID	Name of this instance of the process engine.
AdapterName	Name of the application.
Uptime	Number of milliseconds since this process engine was started.
NewErrors	Total number of errors encountered since the last time this method was called.
TotalErrors	Total number of errors encountered since the process engine was started.
ProcessID	Operating system process ID of the process engine.
Host	Name of the host machine on which the process engine is running.

IsActivityTracingEnabled

Description:

Reports whether tracing is enabled or disabled for the specified activity. If tracing is enabled for all activities by way of a wildcard (*), this method returns false because tracing is not enabled for the activity specifically.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition for which you wish to determine tracing status.
Activity	Name of the activity for which you wish to determine tracing status.

Output:

The following table describes the arguments of this microagent method:

Column Name	Description
Enabled	`true` if tracing is enabled for the specified activity, `false` if tracing is disabled.

IsAllTracingEnabled

Description:

Reports whether tracing is enabled or disabled for all activities and process starters. True is returned when tracing is enabled using a wildcard (*). If tracing is enabled for all activities and/or process starters by specifying a tracing parameter individually for each one, this method will return false.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
ActivityTracingEnabled	`true` when tracing is enabled for all activities. `false` when tracing is disabled for all activities.
ProcessStarterTracingEnabled	`true` when tracing is enabled for all process starters. `false` when tracing is disabled for all process starters.

IsProcessStarterTracingEnabled

Description:

Reports whether tracing is enabled or disabled for the specified process starter. If tracing is enabled for all process starters by way of a wildcard (*), this method returns false because tracing is not enabled for the process starter specifically.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessStarter	Name of the process starter for which you wish to determine tracing status.

Output:

The following table describes the arguments of this microagent method:

Column Name	Description
Enabled	`true` if tracing is enabled for the specified process starter, `false` if tracing is disabled.

IsRoleEnabled

Description:

Reports whether tracing is enabled or disabled for the specified role. If tracing is enabled for all roles by way of a wildcard (*), this method returns false because tracing is not enabled for the role specifically.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Role	Name of the role for which you wish to determine tracing status.

Output:

The following table describes the arguments of this microagent method:

Column Name	Description
Enabled	`true` if tracing is enabled for the specified role, `false` if tracing is disabled.

KillAllProcesses

Description:

Kills all process instances. All process instances are stopped immediately and are permanently removed from the engine.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	The name of the process definition. Only process instances for the specified process definition are killed. If unspecified, this action applies to all process definition.

Output:

None

KillProcess

Description:

Kills the specified process instance. The process instance is stopped immediately and permanently removed from the engine.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessNameOrId	The name or process ID of the process instance you wish to kill. You can retrieve the process ID for a process instance by using the GetProcesses method.

Output:

None

ListAllRoles

Description:

Returns a list of all roles, along with the current state (enabled or disabled) of each role.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Role	Name of the role.
Enabled	True if the role is enabled, false if the role is disabled.

ListDbConnections

Description:

Returns a list of all open and idle JDBC connections that have been opened by a single instance of ActiveMatrix BusinessWorks engine.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Connection-State	State of the JDBC Connection. The connection state can be Active or Idle.
Connection-Name	Name of the JDBC Connection.
User-Name	Username used by the JDBC Connection resource to to connect the database.
Connection-Owner	Owner of the JDBC Connection. For active connections, this field displays the activity name and the job-id for which the connection has been acquired. For idle connections, this field is empty.
Idle-Time	Time in milliseconds for which the JDBC Connection has been idle. For active connections, the value is 0.

Note The statistics for idle connections are available for the time set by the engine property `Engine.DBConnection.idleTimeout`. By default, this property is set to 5 minutes.

ListInstrumentProperties

Description:

Retrieves the current settings for all `Instrumentation` properties.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Property	Lists the Instrumentation properties that are currently set in the form: *<ProcessDefinitionName>=<CurrentValue>*

ListTraceProperties

Description:

Returns the names and current values for all engine trace properties.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Property	Lists the tracing properties and their values in the form: `<TracingPropertyName>=<CurrentValue>`

ListUserDefinedRoles

Description:

Returns a list of user-defined roles, along with the current state (enabled or disabled) of each role.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
Role	Name of the role.
Enabled	True if the role is enabled, false if the role is disabled.

OnProcessActivity

Description:

This method is called when a process executes an activity, and it is only called when instrumentation is on.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
ProcessId	ID of the process instance.
ProcessDef	Process definition name.
ActivityName	Name of the activity.
TrackID	ID of the execution track in which the activity was executed.

OnProcessAdded

Description:

This method is called whenever a process instance is added, and it is only called if instrumentation is on.

Method Arguments:

None

Output:

The following table describes the arguments of this microagent method:

Column Name	Description
ProcessId	Process ID of the process instance that was added.

OnProcessRemoved

Description:

This method is called whenever a process instance is removed, and it is only called if instrumentation is on.

Method Arguments:

None

Output:

The following table describes the arguments of this microagent method:

Column Name	Description
ProcessId	Process ID of the process instance that was added.

OnProcessStatusChanged

Description:

This method is called when a process is suspended or resumed, and it is only called when instrumentation on.

Method Arguments:

None

Output:

The following table describes the output of this microagent method:

Column Name	Description
ProcessId	ID of the process instance.
When	Date and time when the status of the process instance changed.
Active	Status of the process instance. True when the process instance is active, false when it is inactive.
TrackingID	Tracking ID for the process instance.
ProcessDef	Process definition name.

PreRegisterListener

Description:

Pre-register a listener for certified delivery.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Publisher CM Name	Publisher certified name.

Argument Name	Description
Subject	Subject used for certified delivery.
Listener CM Name	Listener certified name.

Output:

None

RemoveRecoverableProcess

Description:

Removes the specified recoverable process instance from the list of potential recoverable processes. After executing this method, the checkpoint data of the specified process instance is removed and the process instance will no longer be able to be recovered. Obtain the process ID of the recoverable process with the GetRecoverableProcesses method.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessId	Process ID of the process instance you wish to remove from the recoverable process instance list.

Output:

None

ResetActivityStats

Description:

Resets the min and max time calculations for each activity in the specified process definition. This method is for internal use only and should not be invoked.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition.

Output:

None

ResetProcessDefinitionStats

Description:

Resets the minimum, maximum, and average time statistics gathered for process instances. Ffor more information about process instance statistics, see GetProcessDefinitions.

Method Arguments:

The following table describes the argument of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition whose statistics you wish to reset.

Output:

None.

RestartRecoverableProcess

Description:

Restarts the specified recoverable process. Obtain the process ID of the recoverable process with the GetRecoverableProcesses method.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessId	Process ID of the process to restart. The process will be restarted from its last saved checkpoint.

Output:

None

ResumeAll

Description:

Resumes all process starters and/or processes.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Action	Specifies what to resume. Can be one of the following: • `AllProcessStarters` — resumes all process starters. • `AllProcesses` — resumes all processes. • `AllProcessStartersAndProcesses` — resumes all processes and process starters.
ProcessDefinition	The name of the process definition.

Output:

None

ResumeProcess

Description:

Resumes the specified process instance.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessNameOrId	The name or process ID of the process instance you wish to resume. You can retrieve the process ID for a process instance by using the GetProcesses method.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Status	Status of the process instance after executing this operation.

ResumeProcessStarter

Description:

Resumes the specified process starter.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition whose process starter you wish to resume.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Status	Status of the process starter after executing this operation.

reviewLedger

Description:

Returns information retrieved from the certified message ledger for the given CM name and subject. If no values are supplied for the method arguments, all ledgers are returned.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
CM Name	Certified name.
Subject	Subject used for certified delivery.

Output:

The following table describes the output of this microagent method:

Column Name	Description
CM Name	Name used for certified delivery.
Subject	Subject used for certified delivery.
Last Sent Message	Sequence number of the most recent message sent with this subject name.
Total Messages	The total number of messages with this subject name for the given CM name.
Total Size	The total storage in bytes occupied by all messages with this subject name.

Column Name	Description
Listener CM Name	Name of the delivery-tracking listener for this subject.
Last Confirmed	Sequence number of the last message confirmed by the listener.
Unacknowledged Messages	Number of messages pending for this listener.

SetInstrumentProperty

Description:

Sets the `Instrumentation` property for the specified process definition to a given value. The OnProcessActivity and OnProcessStateChanged methods will be called for the specified processes definition names.

For example, use property name "*" and value "true" to enable those asynchronous methods for all process definitions. The property name does not need to begin with "`Instrumentation.`", but if it does, the leading "`Instrumentation.`" will be ignored.

For more information about the Instrumentation property, see Enabling TIBCO Hawk.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Name	Name of the process definition for which you wish to alter the `Instrumentation` property. Specify `*` for this argument if you wish to enable or disable instrumentation for all process definitions.
Value	`true` if you wish to enable instrumentation for the given process definition. `false` if you wish to disable instrumentation for the given process definition.

Output:

None

SetTraceProperty

Description:

Sets the specified engine tracing property to the specified value. While you can set properties with this method, ConfigureActivityTracing, ConfigureProcessStarterTracing, and ConfigureUserDefinedTracing are simpler to use for setting trace properties.

For more information about tracing properties, see Trace Properties.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Name	Name of the tracing property you wish to set.
Value	`true` if you wish to enable the property. `false` if you wish to disable the property.

Output:

None

StartStatsCollection

Description:

Enables collection of statistics for each executed activity. For more information about collecting activity statistics, see Storing Process Instance and Activity Statistics.

Method Arguments:

None

Output:

None

stopApplicationInstance

Description:

Shuts down the process engine immediately. All checkpoint files are preserved and the engine's operating system process exits.

Method Arguments:

None

Output:

None

StopStatsCollection

Description:

Disables collection of statistics for each executed activity. For more information about collecting activity statistics, see Storing Process Instance and Activity Statistics.

Method Arguments:

None

Output:

None

SuspendAll

Description:

Suspends all process starters and/or processes.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Action	Specifies what to suspend. Can be one of the following: • `AllProcessStarters` — suspends all process starters. • `AllProcesses` — suspends all processes. • `AllProcessStartersAndProcesses` — suspends all processes and process starters.
ProcessDefinition	The name of the process definition.

Output:

None

SuspendProcess

Description:

Suspends the specified process instance.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessNameOrId	The name or process ID of the process instance you wish to suspend. You can retrieve the process ID for a process instance by using the GetProcesses method.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Status	Status of the process instance after executing this operation.

SuspendProcessStarter

Description:

Suspends the specified process starter.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
ProcessDefinition	Name of the process definition whose process starter you wish to suspend.

Output:

The following table describes the output of this microagent method:

Column Name	Description
Status	Status of the process starter after executing this operation.

unRegisterListener

Description:

Unregister a certified delivery listener.

Method Arguments:

The following table describes the arguments of this microagent method:

Argument Name	Description
Publisher CM Name	Publisher certified name.
Subject	Subject for certified delivery.
Listener CM Name	Listener certified name.

Output:

None

_onUnsolicitedMsg

Description:

Subscribing to this method returns any unsolicited notifications that are sent from the managed application corresponding to this microagent. Invoking this method returns the last such message that was received (if any).

Method Arguments:

None

Monitoring the ActiveMatrix BusinessWorks Engine Using JMX

ActiveMatrix BusinessWorks processes that are deployed on TIBCO Administrator can be monitored using the Java Management Extensions (JMX) API.

This section describes how to enable JMX monitoring for the ActiveMatrix BusinessWorks engine.

Enabling JMX Support

Monitoring ActiveMatrix BusinessWorks engine using the JMX API is possible only for projects deployed on TIBCO Administrator.

To enable JMX monitoring for the ActiveMatrix BusinessWorks engine, set the following property in `bwengine.tra`:

```
Jmx.Enabled=true
```

For more information about configuring JMX with SSL and Authentication, see http://docs.oracle.com/javase/1.5.0/docs/guide/management/agent.html

Once these properties are set, Java monitoring tools such as JConsole can be used instead of TIBCO Hawk to monitor the ActiveMatrix BusinessWorks engine.

Using JConsole

Perform the following tasks to monitor a ActiveMatrix BusinessWorks engine using JConsole.

- Set the properties mentioned in the topic Enabling JMX Support.

- Deploy the ActiveMatrix BusinessWorks project on TIBCO Administrator. Ensure that the TIBCO Hawk Agent has been started.
- Start JConsole and connect to the application that needs to be monitored. By default, JConsole is available in the installation location of Java Development Kit (JDK) for 1.7.0 and higher versions.
- On the JConsole window, click the **MBeans** tab. Select the node `com.tibco.bw in` the tree on the right-hand side. The methods available to monitor the ActiveMatrix BusinessWorks engine are displayed under the Operations tab. These methods are the TIBCO Hawk Microagent methods that you can also invoke using TIBCO Hawk.

For a complete list of the available methods, see TIBCO Hawk Microagent Methods.

Thread Based Grouping of Activities

This section lists the threads on which each ActiveMatrix BusinessWorks activity works.

Active Enterprise Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Adapter Request-Response Server	event-source	bw.aeRRServer	
Adapter Subscriber	event-source	bw.aeSubstriction	
Wait for Adapter Message	signal-in	bw.aeSubSignalInActivity	
Invoke an Adapter Request-Response Service	activity	bw.aeOpClientReqActivity	Private
Respond to Adapter Request	activity	bw.aeOpServerReplyActivity	
Publish to Adapter	activity	bw.aePubActivity	
Send Exception to Adapter Request	activity	bw.aeOpServerFaultActivity	

File Palette

Activity Name	Activity Kind	Activity Type	Threads Used
File Poller	event-source	`bw.FileEventSourceResource`	Private
Wait for File Change	signal-in	`bw.FileSignalInUI`	Private
Copy File	activity	`bw.FileCopyActivity`	Engine
Create file	activity	`bw.FileCreateActivity`	Engine
List File	activity	`bw.ListFilesActivity`	Engine
Read File	activity	`bw.FileReadActivity`	Engine
Remove File	activity	`bw.FileRemoveActivity`	Engine
Rename File	activity	`bw.FileRenameActivity`	Engine
Write File	activity	`bw.FileWriteActivity`	Engine

FTP Palette

Activity Name	Activity Kind	Activity Type	Threads Used
FTP Change Default Directory	activity	`bw.FTPChangeDefaultDirActivityUI`	Engine
FTP Delete File activity	activity	`bw.FTPDeleteFileActivityUI`	Engine
FTP Dir	activity	`bw.FTPDirActivityUI`	Engine
FTP Get Default Directory	activity	`bw.FTPGetDefaultDirActivityUI`	Engine

Activity Name	Activity Kind	Activity Type	Threads Used
FTP Get	activity	`bw.FTPGetActivityUI`	Engine
FTP Make Remote Directory	activity	`bw.FTPMakeRemoteDirActivityUI`	Engine
FTP Put	activity	`bw.FTPPutActivityUI`	Engine
FTP Quote	activity	`bw.FTPQuoteActivityUI`	Engine
FTP Remove Remote	activity	`bw.FTPRemoveRemoteDirActivityUI`	Engine
FTP Rename File	activity	`bw.FTPRenameActivityUI`	Engine
FTP SYS Type	activity	`bw.FTPSysTypeActivityUI`	Engine

General Activities Palette

Activity Name	Activity Kind	Activity Type	Threads Used
On Event Timeout	event-source	`bw.onEventTimeout`	Engine
On Notification Timeout	event-source	`bw.onNotificationTimeout`	
On Shutdown	event-source	`bw.onShutdown`	Engine
On Startup	event-source	`bw.onStartup`	Engine
On Error	event-source	`bw.onErrorProcess`	Engine
Receive Notification	event-source	`bw.waitStarter`	
Timer	event-source	`bw.timer`	
Catch	signal-in	`bw.catch`	

Activity Name	Activity Kind	Activity Type	Threads Used
Sleep	signal-in	`bw.sleep`	Engine
Wait	signal-in	`bw.waitActivity`	Engine
Assign	activity	`bw.assignActivity`	Engine
Call Process	activity	`bw.subprocess`	Engine
Checkpoint	activity	`bw.checkpoint`	
Confirm	activity	`bw.confirm`	Engine
Engine Command	activity	`bw.enginecommand`	Engine
External Command	activity	`bw.CmdExecActivity`	
Generate Error	activity	`bw.throw`	
Get Shared Variable	activity	`bw.getSharedVariable`	
Inspector	activity	`bw.inspectorActivity`	
Label	activity	`bw.label`	
Mapper	activity	`bw.MapperActivity`	Engine
Notify	activity	`bw.notifyActivity`	
Null	activity	`bw.null`	
Rethrow	activity	`bw.rethrow`	
Set Shared Variable	activity	`bw.setSharedVariable`	Engine
Write To Log	activity	`bw.log`	Engine

HTTP Palette

Activity Name	Activity Kind	Activity Type	Threads Used
HTTP Receiver	event-source	`bw.httpEventSource`	Private
Wait for HTTP Request	signal-in	`bw.httpSignalIn`	
HTTP Send Request	activity	`bw.httpRequest`	Private
HTTP Send Response	activity	`bw.httpWebResponse`	

JAVA Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Java Code	activity	`bw.javaActivity`	
Java Event Source	event-source	`bw.JavaEventSource`	
Java Method	activity	`bw.JavaMethodActivity`	
Java To XML	activity	`bw.JavaToXmlActivity`	
XML To Java	activity	`bw.XmlToJavaActivity`	

JDBC Palette

Activity Name	Activity Kind	Activity Type	Threads Used
JDBC Call Procedure	activity	`bw.JDBCCallActivity`	
JDBC Get Connection	activity	`bw.JDBCGetConnectionActivity`	
JDBC Query	activity	`bw.JDBCQueryActivity`	Engine

Activity Name	Activity Kind	Activity Type	Threads Used
JDBC Update	activity	`bw.JDBCUpdateActivity`	
SQL Direct	activity	`bw.JDBCGeneralActivity`	

JMS Palette

Activity Name	Activity Kind	Activity Type	Threads Used
JMS Queue Receiver	event-source	`bw.JMSQueueEventSource`	Private
JMS Topic Subscriber	event-source	`bw.JMSTopicEventSource`	
Wait for JMS Queue Message	signal-in	`bw.JMSQueueSignalInActivity`	
Get JMS Queue Message	activity	`bw.JMSQueueGetMessageActivity`	
JMS Queue Requestor	activity	`bw.JMSQueueRequestReplyActivity`	Private
JMS Queue Sender	activity	`bw.JMSQueueSendActivity`	Engine
JMS Topic Publisher	activity	`bw.JMSTopicPublishActivity`	
JMS Topic Requestor	activity	`bw.JMSTopicRequestReplyActivity`	Private
Reply to JMS Message	activity	`bw.JMSReplyActivity`	

Mail Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Receive Mail	event-source	`bw.MailEventSourceResource`	Private
Send Mail	activity	`bw.MailActivityResource`	Engine

Parse Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Parse Data	activity	`bw.ParseActivity`	
Render Data	activity	`bw.RenderActivity`	

RMI Palette

Activity Name	Activity Kind	Activity Type	Threads Used
RMI Lookup	activity	`bw.lookup`	
RMI Server	activity	`bw.starter`	

RV Palette

Activity Name	Activity Kind	Activity Type	Threads Used
RV Subscriber	event-source	`bw.RVEventSource`	
Wait for RV Message	signal-in	`bw.rvSignalInActivity`	
Publish RV Message	activity	`bw.RVPubActivity`	

Activity Name	Activity Kind	Activity Type	Threads Used
Reply to RV Request	activity	`bw.RVReplyActivity`	
Send RV Request	activity	`bw.RVRequestActivity`	Private

Service Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Get Context	activity	`bw.getContext`	Engine
Invoke Partner	activity	`bw.invokePartner`	Private/Engine (Depends on the binding)
Set Context	activity	`bw.setContext`	Engine

SOAP Palette

Activity Name	Activity Kind	Activity Type	Threads Used
SOAP Event Source	event-source	`bw.SOAPEventSourceUI`	
Retrieve Resources	activity	`bw.RetrieveResource`	
SOAP Request Reply	activity	`bw.SOAPSendReceiveUI`	Private
SOAP Send Fault	activity	`bw.SOAPSendFaultUI`	
SOAP Send Reply	activity	`bw.SOAPSendReplyUI`	
MIME Parse	activity	`bw.MimeParserActivity`	

TCP Palette

Activity Name	Activity Kind	Activity Type	Threads Used
TCP Receiver	event-source	`bw.TCPEventSource`	
Wait for TCP Request	signal-in	`bw.TCPSignalIn`	
Read TCP Data	activity	`bw.TCPRead`	
TCP Close Connection	activity	`bw.TCPCloseConnection`	
TCP Open Connection	activity	`bw.TCPOpenConnection`	
Write TCP Data	activity	`bw.TCPWrite`	

Transaction Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Transaction State	activity	`bw.TransactionStateActivity`	

XML Activities Palette

Activity Name	Activity Kind	Activity Type	Threads Used
Parse XML	activity	`bw.XMLParseActivity`	
Render XML	activity	`bw.XMLRendererActivity`	
Transform XML	activity	`bw.XMLTransformActivity`	

TIBCO Product Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

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

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The following documentation for this product is available on the TIBCO ActiveMatrix BusinessWorks™ Product Documentation page:

- *TIBCO ActiveMatrix BusinessWorks™ Release Notes*
- *TIBCO ActiveMatrix BusinessWorks™ Administration*
- *TIBCO ActiveMatrix BusinessWorks™ Concepts*
- *TIBCO ActiveMatrix BusinessWorks™ Error Codes*
- *TIBCO ActiveMatrix BusinessWorks™ Getting Started*
- *TIBCO ActiveMatrix BusinessWorks™ Installation*
- *TIBCO ActiveMatrix BusinessWorks™ Palette Reference*
- *TIBCO ActiveMatrix BusinessWorks™ Process Design*

To directly access documentation for this product, double-click the following file:

`TIBCO_HOME/release_notes/TIB_<productID>_version_docinfo.html` where *TIBCO_HOME* is the top-level directory in which TIBCO products are installed. On Windows, the default *TIBCO_HOME* is `C:\tibco`. On UNIX systems, the default *TIBCO_HOME* is `/opt/tibco`.

Other TIBCO Product Documentation

When working with ActiveMatrix BusinessWorks, you may find it useful to read the documentation of the following TIBCO products:

- TIBCO Designer™: *TIBCO Designer is an easy to use graphical user interface for design-time configuration of TIBCO applications. TIBCO Designer includes online help for each palette.*
- TIBCO Runtime Agent™: *TRA supplies a number of TIBCO and third-party libraries used by ActiveMatrix BusinessWorks*.
- TIBCO Administrator™: *TIBCO Administrator is the monitoring and managing interface for new-generation TIBCO products such as ActiveMatrix BusinessWorks*.
- TIBCO Rendezvous® : *TIBCO Rendezvous software uses messages to enable distributed application programs to communicate across a wide variety of hardware platforms and programming languages.*

How to Access Related Third-Party Documentation

When working with ActiveMatrix BusinessWorks, you may find it useful to read the documentation of the following third-party products:

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

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

How to Join TIBCO Community

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

Legal and Third-Party Notices

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