TIBCO Silver® Fabric Enabler for ActiveMatrix BusinessWorks™ User's Guide

Software Release 3.5
June 2017

Two-Second Advantage®

Important Information

Contents

Figures

TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, visit:

https://docs.tibco.com

Product-Specific Documentation

The following documents for this product can be found on the TIBCO Documentation site:

- *TIBCO Silver® Fabric Enabler for ActiveMatrix BusinessWorks™ Installation*
- *TIBCO Silver® Fabric Enabler for ActiveMatrix BusinessWorks™ User's Guide*
- *TIBCO Silver® Fabric Enabler for ActiveMatrix BusinessWorks™ Developer's Guide*
- *TIBCO Silver® Fabric Enabler for ActiveMatrix BusinessWorks™ Release Notes*
- *TIBCO Silver® Fabric Enabler for ActiveMatrix BusinessWorks™ Plug-in Distribution Quick Start Guide*

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

How to Join TIBCO Community

TIBCO Community is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCO Community offers forums, blogs, and access to a variety of resources. To register, go to the following web address:

https://community.tibco.com

Introduction

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks is a complementary software component. TIBCO ActiveMatrix BusinessWorks projects can be quickly published using standardized component configurations in cloud environments based on TIBCO Silver® Fabric. This accelerates publishing of BusinessWorks projects, enforces its industry best practices, and provides elastic optimization of computing resources.

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks simplifies configuration and publishing of TIBCO ActiveMatrix BusinessWorks server, TIBCO ActiveMatrix BusinessWorks Runtime Nodes, and associated dependencies to run an integrated ActiveMatrix BusinessWorks system environment on the TIBCO Silver Fabric Cloud.

Create Silver Fabric ActiveMatrix BusinessWorks stacks with their component dependencies and get a complete TIBCO ActiveMatrix BusinessWorks environment on the cloud.

Main Functionalities

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks provides the following main functionality:

- Quickly set up BusinessWorks environments on multiple TIBCO Silver Fabric Engines.
- Publish your BusinessWorks archives to run on TIBCO ActiveMatrix BusinessWorks instances running on TIBCO Silver Fabric Engines. Scale those archives based on conditions you define.
- Publish BusinessWorks projects onto this environment using traditional tools, such as the TIBCO Administrator User Interface or its command-line tools.
- Define, set up, and publish a complete stack based on BusinessWorks projects onto a set of virtual or physical machines. These actions include installation of this software, creation of TIBCO Domain, starting up TIBCO Administrator server, and publishing of the BusinessWorks projects on one or multiple machines.
- Publish stack of components that follow a set of recommended and supported TIBCO practices to implement fault tolerance, load balancing, software updates, and stack updates.
- Leverage selected BusinessWorks metrics or selected BusinessWorks engine statistics to set thresholds for scaling and other actions. It scales up and down the number of BusinessWorks engines required to process the workload. Therefore, it provides elasticity and optimization of computing resources.
- Create pools of TIBCO Domain Logical Machines (TLM) for scaling up and scaling down TIBCO ActiveMatrix BusinessWorks Engines, component archives.
- Gather and report statistics from each ActiveMatrix BusinessWorks™ service instance deployed by TIBCO Silver® Fabric to support automated rule-based scaling from archive statistics.
- Configure fast TLM restart for engines from shared drives. Planned TLM restarts are streamlined so that large numbers of applications with multiple archives can be restarted individually and more quickly.

Components

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks creates a component wizard within TIBCO Silver Fabric Administrator.

You can configure TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks to use one of several distributions of TIBCO® ActiveMatrix BusinessWorks™. The TIBCO® ActiveMatrix BusinessWorks™ Distribution is a compressed grid library that can be used to instantiate the product on multiple TIBCO Silver Fabric engines.

You must consider dependencies while selecting TIBCO® ActiveMatrix BusinessWorks™distribution version for your component. Though you can use any distribution, a good practice is to use the latest distribution releases to take advantage of the best product functionality.

Supported Distributions

This release of TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks supports TIBCO® ActiveMatrix BusinessWorks Distribution for TIBCO Silver® Fabric release versions 5.9.2- 5.13.x, and 6.2 .x recent version.

As the architecture of BusinessWorks 5.x and 6.x are very different, the component wizard changes dramatically depending on the selected version of TIBCO ActiveMatrix BusinessWorks Distribution. The information about BusinessWorks 5.x and 6.x component wizard interface is provided in separate sections.

- When using **BusinessWorks Distribution 5.9.2- 5.13.x**, refer to the section Creating BusinessWorks Components with 5.x Distributions**.**
- When using **BusinessWorks Distribution 6.2 and more recent**, refer to the section Creating BusinessWorks Components with 6.x Distributions.

  The prerequisites and other dependencies required for the distribution and applications are distinct for the 5.x releases versus the 6.x releases, so refer to the appropriate documentation set. Refer the readme file and Installation guide for both optional and required supporting software.

Enabler Distribution Support

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks 3.x was made primarily for distributions of TIBCO® ActiveMatrix BusinessWorks™ 6.x releases, but it is backward compatible for all supported 5.x distribution releases.

The tasks, procedures, and configurations described in this document are applicable when using this product with TIBCO® ActiveMatrix BusinessWorks Distribution for TIBCO Silver® Fabric.

Prerequisites

Configuration of an ActiveMatrix BusinessWorks component requires specific knowledge about external dependencies. In most cases, you also need files such as JDBC driver to create connections with the databases.

Component configurations depend on environment implementation details to establish connections with external services hosted somewhere beyond the local host. Using external services can be optional for development systems, but many production systems require a more complex architecture with externally hosted services and dependencies.

In particular, prior to publishing and deploying the rest of your BusinessWorks stack, you must keep the database and the messaging service in place or ready to be deployed. When ActiveMatrix BusinessWorks server is published and run on TIBCO Silver Fabric, the database is initialized with an ActiveMatrix BusinessWorks specific schema.

Additionally, you can plan to host several services separately from the TIBCO ActiveMatrix BusinessWorks server. To externally host the following services with respect to TIBCO ActiveMatrix BusinessWorks, make sure that these services are already installed and running.

- External EMS for communication transport and as a Group Provider technology
- External FTL Server for communication transport or as a group provider technology
- External MySQL or PostgreSQL, or Oracle database in case of ActiveMatrix BusinessWorks 6.x

During the component configuration, you can then specify the URL and connection profile values for the mentioned services.

Required and Optional Products

Refer to the product readme to get the list of supported versions.

Software Requirements

Software	Description
TIBCO Software The following software products are distributed and installed separately from this product. See the readme file for the supported versions.	
TIBCO Silver® Fabric	Required. To use Silver Fabric, you must install TIBCO Silver Fabric.
TIBCO Silver(R) Fabric Enabler for TIBCO Administrator™ - Enterprise Edition	Optional. If you want to create a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks 5.x application, you need this product. It creates a TIBCO Administrator component, TIBCO Domain, and start TIBCO Hawk® services and a TIBCO Administrator server.

Software	Description
TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator	Optional. If you want to create a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks 6.x application, you need this product. TIBCO ActiveMatrix BusinessWorks registers the bwagent with TIBCO Enterprise Administrator and lets you manage the BusinessWorks Domain, AppSpaces, AppNodes, applications and so on. Please refer to *TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator* for details.
TIBCO ActiveMatrix® BusinessWorks Distribution for TIBCO Silver® Fabric	Required. This distribution is published to the engine instances. You must download and install this product according to instructions in the *TIBCO Silver® Fabric Installation Guide*.
TIBCO Enterprise Message Service™ Or TIBCO Silver® Fabric Enabler for TIBCO Enterprise Message Service™	Optional. If your implementation uses TIBCO Enterprise Message Service (EMS) the message transport to your database. TIBCO EMS and an associated database product must be installed separately. If EMS is used as a Group Provider technology. The configurations of TIBCO EMS product are defined in the TIBCO Enterprise Message Service product documentation.
TIBCO FTL®	Optional. If your implementation uses TIBCO FTL for transport or if it is used as group provider technology.
Third-party Software See the readme file for the supported versions.	

Software	Description
Database	Optional. **For 5.x version**: If you want to create a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks 5.x application, you can optionally store the domain data in a database. However, if you use TIBCO Enterprise Message Service (EMS) to transport your messages, you must store the domain data in a database. The EMS and the database product must be installed separately or using the Silver Fabric Enabler for EMS it can be deployed to a Silver Fabric Engine as a dependency. The configurations of these products are defined in the related product documentation. For information about the connection of TIBCO Domain to a database, refer to TIBCO Administrator Application component or Creating a Domain that Uses a Database in Domain Utility User's Guide in TIBCO Runtime Agent documentation. **For 6.x version**: If you want to create a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks 6.x application, refer to the *TIBCO ActiveMatrix BusinessWorks*™ *Readme* for a list of supported release versions of PostgreSQL, MySQL, Microsoft SQL Server, Oracle, and DB2 databases. Proper ActiveMatrix BusinessWorks component deployment requires an administrator connection with a supported database for automated table and schema creation and runtime management.
Microsoft Visual C++ 2005 Redistributable Package	Optional. It is required only if you want to create a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks 5.x application. To use TIBCO Rendezvous on the BusinessWorks domain, Microsoft Visual C++ 2005 Redistributable Package must be installed on your Windows operating system. Note: If Microsoft Visual C++ 2005 Redistributable Package is not included in the Windows version, refer to the ReadMe file for the supported platforms, versions, and required patches. It should be installed separately. • Microsoft Visual C++ 2005 Redistributable Package (x86) can be downloaded from the following website: http://www.microsoft.com/downloads/en/details.aspx?FamilyId=32BC1BEE-A3F9-4C13-9C99-220B62A191EE&displaylang=en • Microsoft Visual C++ 2005 Redistributable Package (x64) can be downloaded from the following website: http://www.microsoft.com/en-us/download/details.aspx?id=21254

Refer to *TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks Installation Guide* for more information.

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks

TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks creates a component wizard within TIBCO Silver® Fabric Administrator.

- The component wizard provides an interface to make specific component configurations for publishing TIBCO ActiveMatrix BusinessWorks to one or multiple engines on the TIBCO Silver Fabric cloud.
- Running that published component installs and runs TIBCO ActiveMatrix BusinessWorks on a TIBCO Silver Fabric Engine with any BusinessWorks Application Archives that you upload and configure.

Before building and running a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks stack, make sure you have:

1. Created and published one or more BusinessWorks components.

   For 5.x: Refer to Creating BusinessWorks Components with 5.x Distributions.

   For 6.x: Refer to Creating BusinessWorks Components with 6.x Distributions.
2. Created a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks stack. Refer to Stacks in Silver Fabric.
3. Set a dependency to the TIBCO Administrator component or TIBCO Enterprise Administrator Component for each BusinessWorks components. Refer to Dependency Requirements.
4. Optionally, set rules for TIBCO Silver® Fabric Engines. Refer to BusinessWorks Components Scaling and Statistics.

After completing these tasks, you can run and update a TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks stack. Refer to Running Stacks for information on how to run and Updating the TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks Stack on how to update a stack.

Creating BusinessWorks Components with 5.x Distributions

This component will perform the following tasks:

- Add a virtual machine to TIBCO Domain.
- Start TIBCO Hawk® Agent.
- Publish one or more BusinessWorks projects. This task is optional.

If you do not upload the BusinessWorks projects, you still can publish the BusinessWorks projects on the machine where BusinessWorks registers to TIBCO Domain using the TIBCO Administrator GUI or deploys with REST service.

Prerequisites

Before creating the TIBCO ActiveMatrix BusinessWorks component, you must first create a TIBCO Administrator component. It creates a TIBCO Domain, and starts TIBCO Hawk® services and a TIBCO Administrator server. Refer to *TIBCO Silver Fabric Enabler for TIBCO Administrator User's Guide* for creating and configuring a TIBCO Enterprise Administrator component.

Procedure

1. Enter name and description for the new BusinessWorks component.
   See Specifying Component Name for more details on configuration options.
2. Enter values for each parameter field as applicable, in the Choose Product distribution page.
   See Selecting the Distribution Version for more details.
3. Configure the basic configuration parameters in the TIBCO BusinessWorks Basic Configuration page.
   See Setting the BusinessWorks Basic Configuration for more details on configuration options.
4. (Optional) Upload BusinessWorks archive(s) to publish and run BusinessWorks projects.
   See Uploading a BusinessWorks Project for more details on uploading BusinessWorks archives.
5. (Optional) Configure the Hawk Application Management Interface.
   See Hawk Application Management Interface (AMI) Configuration (optional) for more details on configuration options.
6. Upload the EMS SSL Certification
   See Upload EMS SSL Cerification for more details on uploading EMS SSL Certification.
7. (Optional) Enter the base values for the variables on the HTTP Port Management Page.
   See HTTP Port Settings Management (optional) for more details.
8. Configure the running condition parameters in the TIBCO Hawk Agent Running Condition page.
   See Configure TIBCO Hawk Agent Running Condition for details on configuration options.
9. (Optional) Upload Hawk Micro agent Plug-ins to Hawk agent as .zip file.
   See Uploading Hawk MicroAgent Plugin (optional) for more details.
10. (Optional) Configure the runtime parameters for the BusinessWorks component.
    See Adding or Editing Runtime Context Variables (optional) for details on configuration options.
11. (Optional) Upload a content file specific to the BusinessWorks component.
    See Upload, Add, Customize, or Remove a Content File for more details.
12. Add or remove statistics on component for tracking.
    See Adding or Removing BusinessWorks Component Statistics for available list for 5.x components.

13. Add rules to specify and set component behaviour.

    See Allocation Rule Settings for more details.

14. The following tasks of the Component Wizard are generic for all Silver® Fabric Enablers. The configuration of these is optional for BusinessWorks component.

    - Add/edit Application Component scripts
    - Edit the configuration file

15. After creating the component, publish it.

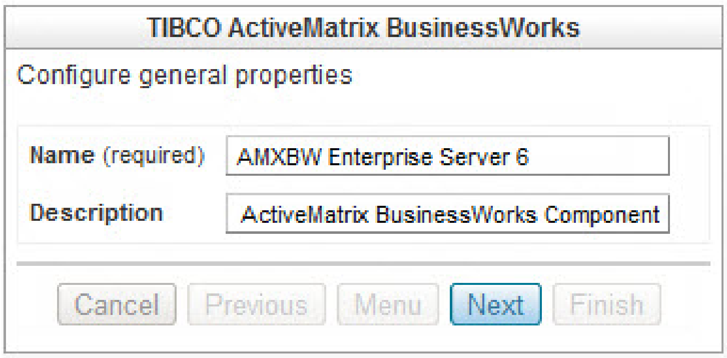
    See Finishing Configuring the Component for more details.

Specifying Component Name

Procedure

1. In TIBCO Silver® Fabric Administration Tool, select **Stacks** > **Components** .
2. In the Components page, select **Create New TIBCO ActiveMatrix BusinessWorks Component** in the Global Actions list.
3. Provide a **Name** and **Description** for the component. Click **Next**.

   Name and Description of the Component


TIBCO ActiveMatrix BusinessWorks

Configure general properties

Name (required)	AMXBW Enterprise Server 6
Description	ActiveMatrix BusinessWorks Component

Where previous versions of the TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks were installed, you can select an older version of Silver® Fabric BusinessWorks Enabler. Ensure that you use the latest version of the installation, unless you have a specific reason to select an older version.

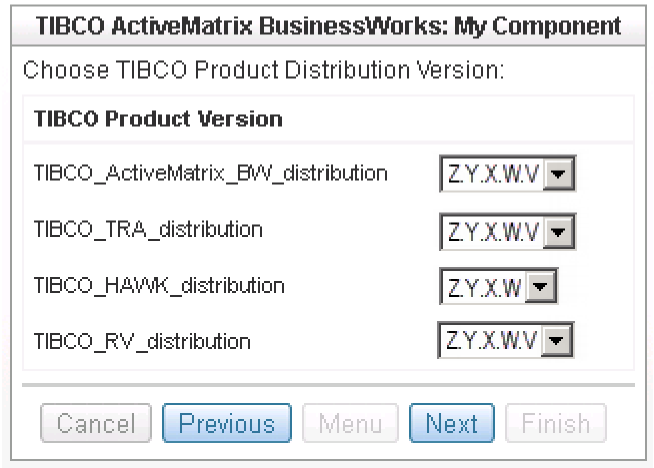
Selecting the Distribution Version

Procedure

1. Select the versions of each distribution that should be published with your TIBCO® ActiveMatrix BusinessWorks Distribution for TIBCO Silver® Fabric .

   Select from those distributions already installed in the Silver® Fabric Broker. By default, the latest versions of the distributions are displayed.

Choose TIBCO Product Distribution Versions



TIBCO ActiveMatrix BusinessWorks: My Component

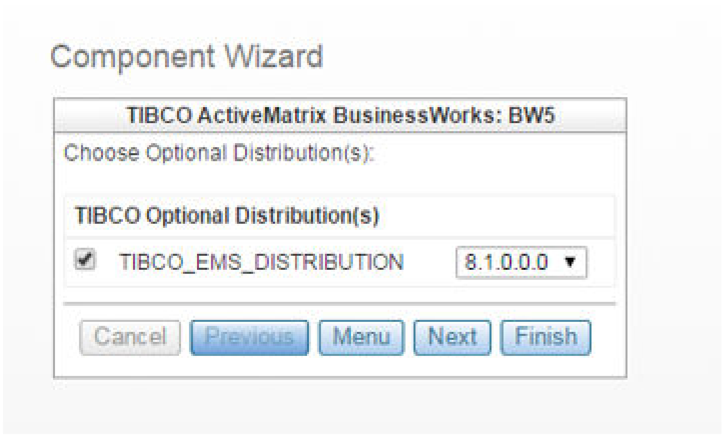
Choose TIBCO Product Distribution Version:

TIBCO Product Version	
TIBCO_ActiveMatrix_BW_distribution	Z.Y.X.W.V
TIBCO_TRA_distribution	Z.Y.X.W.V
TIBCO_HAWK_distribution	Z.Y.X.W
TIBCO_RV_distribution	Z.Y.X.W.V

There is no EMS client embedded in TIBCO Runtime Agent since version 5.9.x. If you use EMS as a transport, you must install the EMS distribution and then select an EMS distribution version in the optional dependency screen.

2. Select the optional distribution and click **Next**.

   Choose Optional Distributions



Component Wizard

TIBCO ActiveMatrix BusinessWorks: BW5

Choose Optional Distribution(s):

TIBCO Optional Distribution(s)	
✓ TIBCO_EMS_DISTRIBUTION	8.1.0.0.0

You must choose Optional Distribution(s) only if your TIBCO Administrator implementation uses TIBCO Enterprise Messaging Service™ (EMS) or the deployed BusinessWorks projects uses EMS. You must then upload the latest TIBCO EMS distribution to the TIBCO Silver Fabric Broker.

There is no EMS client embedded in TIBCO Runtime Agent since version 5.9.x. If you use EMS as a transport, you must install the EMS distribution and then select an EMS distribution version in the optional dependency screen.

Setting the BusinessWorks Basic Configuration

The TIBCO BusinessWorks Basic Configuration page allows for definition of the JDBC driver, setting a specific TIBCO Domain machine name (the TIBCO Logical Machine name - TLM), provides for specification of Fast TLM Restart, and allows uploading of external jar files and prepending/appending the path in the Classpath.

Each of the settings on this page deserves an explanation so they will be described in the order of their appearance on the page.

Procedure

1. Upload a JDBC Driver to publish with the TIBCO® ActiveMatrix BusinessWorks Distribution for TIBCO Silver® Fabric.

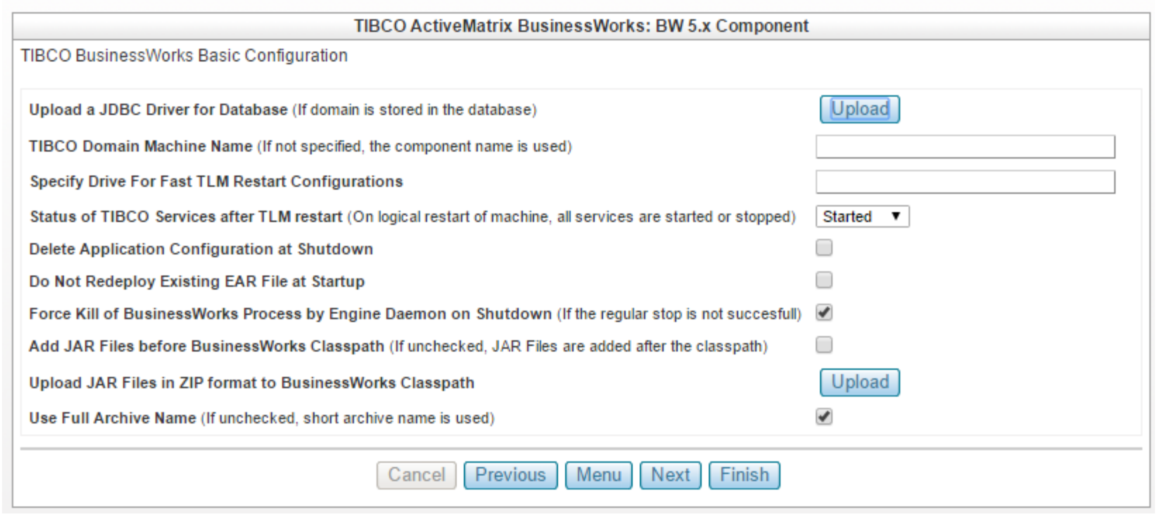
   See Uploading a JDBC Driver.

2. If the Domain is Enable EMS SSL

   Check this option when the admin domain uses EMS SSL. A page Upload EMS SSL certifications is displayed in the component wizard. You can upload the trusted certificate, digital certificate and private key.

3. Set the **Domain Machine Name** (Optional) for publishing of a unique component.

   Configure the TIBCO Logical Machine Name

   TIBCO ActiveMatrix BusinessWorks: BW 5.x Component

   TIBCO BusinessWorks Basic Configuration

Upload a JDBC Driver for Database (If domain is stored in the database)	Upload
TIBCO Domain Machine Name (If not specified, the component name is used)	
Specify Drive For Fast TLM Restart Configurations	
Status of TIBCO Services after TLM restart (On logical restart of machine, all services are started or stopped)	Started
Delete Application Configuration at Shutdown	☐
Do Not Redeploy Existing EAR File at Startup	☐
Force Kill of BusinessWorks Process by Engine Daemon on Shutdown (If the regular stop is not succesfull)	✓
Add JAR Files before BusinessWorks Classpath (If unchecked, JAR Files are added after the classpath)	☐
Upload JAR Files in ZIP format to BusinessWorks Classpath	Upload
Use Full Archive Name (If unchecked, short archive name is used)	✓

   See TIBCO domain Logical Machine Name configuration for more details.

4. Set the Fast TLM Restart Shared Drive.

   See Setting the Fast TLM Restart Shared Drivefor more details.

5. Enter status of TIBCO Services after TLM restart.

   See TIBCO Service status after TLM Restart for more details on status setting.

6. Select **Delete Application Configuration at Shutdown**. This option undeploys and removes all the applications deployed on the TIBCO Logical Machine at component shutdown.

7. Select **Do not Redeploy Existing EAR File at Startup** check box to avoid redeployment of the EAR file whenever the TIBCO Logical Machine restarts.

8. Select **Force Kill of BusinessWorks Process by Engine Daemon at Shutdown** checkbox to forcefully kill the BusinessWorks processes at shutdown. Even though the BusinessWorks engine is always stopped at shutdown, in case the Hawk is down, BusinessWorks engine stops abruptly resulting in orphan engines. To avoid such orphan engines, the engine daemon kills the bwengine forcefully.

9. Add JAR Files before for BusinessWorks Classpath

   See Adding JAR Files(s) to the Business Component for more details.

10. Upload JAR file(s) in ZIP format to the BusinessWorks Classpath- Uploading individual JAR file(s) either appends or prepends to the Classpath. All uploaded JARs adds in the same way according to how the check box is set.

To remove unwanted JAR files use the Menu button to display the list of Wizard configuration steps and select **Add/Override/Customize Enabler and Component-specific content files**. Relative paths to external jar files added might be removed with that window.

11. Configure Archive Name for archive scaling.

    See Archive Name Configuration for more details.

Uploading a JDBC Driver (Optional)

Procedure

- Specify a JDBC driver to publish with the TIBCO® ActiveMatrix BusinessWorks Distribution for TIBCO Silver® Fabric so that it can communicate with the TIBCO Administrator Domain database. When the TIBCO Administrator uses a database as the domain storage, you must upload a JDBC driver so the component can interact with it.

  The JDBC driver must match the database type used by the TIBCO Administrator. This procedure is the same as uploading a JDBC driver for the TIBCO Administrator component.

  Each BusinessWorks 5.x component is dependent on TIBCO Administrator. Refer to Dependency Requirements for setting that dependency.

  If the domain is not stored in a database, then do not upload the JDBC driver.

TIBCO Domain Logical Machine Name Configuration

The TIBCO Domain Machine Name (also known as the TIBCO Logical Machine name TLM) provides for publishing of a unique component. The TIBCO Domain Machine virtualizes the machine publishing so that component publishing can maintain state when the targeted engine is changed for whatever reason.

If for example, the target machine is restarted because of an OS update or a hardware change, you can restart the virtualized machine on different hardware using TLM.

When the TIBCO Domain Machine name is set, the component .ear or .JAR files are republished and they start on the new virtual machine hardware. BusinessWorks Applications that have been previously published through TIBCO Administrator or through the AppManage commandline interface are published again and restarted on the new hardware.

The configuration parameter is as described.

TIBCO Domain machine name

- To activate use of the TIBCO Domain Logical Machine Name enter a value in the TIBCO Domain machine name field. The TIBCO Domain machine name and can use alphanumeric characters, hyphen (-), or underscore (_) characters. Do not use other special characters, including period or comma. The name length must be less than 64 characters.

When the component is instantiated multiple times, the TLM machine name is: <*Your_TLM_Name*>_<*ComponentInstanceNumber*> , where the <*ComponentInstanceNumber*> is just a sequential incrementing integer.

If the TIBCO Domain machine name field is left blank the component name is used for setting the domain machine name. When the component number is 0, the TLM machine name should be `<Your_TLM_Name>`

If runtime variable REMOVE_COMPONENT_INSTANCE_FROM_MACHINE_NAME is set to `true`, the TLM machine name does not add a suffix with the component instance number and the component scaling cannot be done.

Setting the Fast TLM Restart Shared Drive

Fast TIBCO Logical Machine Restart provides for accelerated restart of the engine so that many archives can be re-published within a reduced amount of time.

Enhanced restart times are achieved by using a saved state stored on a shared Network File System drive instead of synchronizing with the domain repository. To set your expectations properly, "fast" does not mean instantaneous or even amazingly fast, but it is faster than if the archives were loaded from the domain repository.

Domain configuration changes made through TIBCO Administrator or the AppManage CLI in the period between the TLM stop and the TLM restart are not captured.

If Domain configuration changes made during the period after TLM stop and before TLM restart must be captured then the user must redeploy the modified application.

Recommendation: For those implementations that use fewer than ten for BusinessWorks deployments per component, avoid using Fast TLM restart to avoid the constraints mentioned.

If you want to force redeployment (synchronization with the domain) once, add a file call `ForceRedeploy.txt` in the domainDataHome (<*domainDataDir*>/<*DomainName*>/<*TLMNAME*>/<*DomainName*>. It redeploys all applications (no FAST TLM) at startup and then deletes the file (it is one time action).

Prerequisites

The shared NFS directory drive for Fast TLM Restart requires the following:

- The shared drive must be READ/WRITE accessible to all engine daemons running as TIBCO Logical Machines in a TIBCO Silver Cloud.
- All TLM in a stack must run the same OS.

To enable Fast TIBCO Logical Machine restart simply enter the directory path of the shared NFS drive (on Windows servers - a mapped drive) and make sure that the host grants permissions allowing the user who launches the engines to read and write in that location.

TIBCO Service Status after TLM restart

Sets the desired services state when the TIBCO Logical Machine is restarted. When the TLM is restarted service instances are republished and either **started** or **stopped**. The stopped services state setting might be convenient for developers who are testing machines with many services and don’t have to start it for every logical machine change.

Adding JAR Files(s) to the Business Component

Procedure

- Add external JAR file(s) to the BusinessWorks component

  As an option, JAR files can be uploaded for publishing with the TIBCO BusinessWorks component.

  The check box **Add JAR Files before BusinessWorks Classpath (if unchecked, JAR Files are added after the Classpath)** means just what it says. The JAR file name can be prepended in front of the ClassPath by selecting the box and if it is left unchecked, the JAR file name is appended at the end of the ClassPath. Of course, this setting does not apply if a JAR file is not uploaded for addition to the classPath.

Archive Name Configuration

The Archive Name is used for archive scaling. You may choose either Full ArchiveNames or Short ArchiveNames depending on your implementation of archives in the stacks.

Use the full Archive name by checking the box if you will reuse archives more than once.

Use the short Archive name, leaving the box unchecked, if you will not use the same archives in different applications. The short Archive name makes a more convenient name to type, but the shortened name disallows reuse of the same archive in different TIBCO Administrator applications.

Within the same stack, full archive names and short archive names can't be mixed.

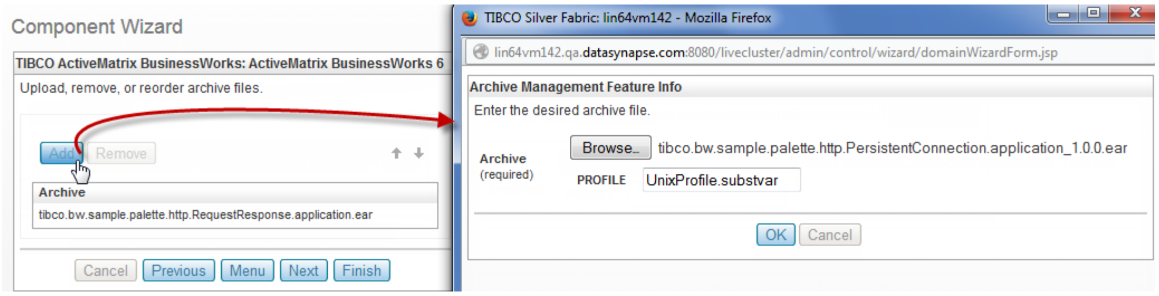
Uploading a BusinessWorks Project (optional)

If you want TIBCO Silver Fabric for BusinessWorks components to publish and run one or more BusinessWorks projects, upload one or more archive files (EAR or ZIP files) as follows:

Procedure

1. Click the **Add** button in the Upload, Remove, or Recorder Archive Files panel, as shown in the following figure:

   Uploading Archives


Component Wizard
TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6
Upload, remove, or reorder archive files.
Add
Remove
Archive
Cancel
Previous
Menu
Next
Finish
TIBCO Silver Fabric: lin64vm142 - Mozilla Firefox
Archive Management Feature Info
Enter the desired archive file.
Archive (required)
Browse...
PROFILE
OK
Cancel

tibco.bw.sample.palette.http.RequestResponse.application.ear

lin64vm142.qa.datasynapse.com:8080/livecluster/admin/control/wizard/domainWizardForm.jsp

Archive: tibco.bw.sample.palette.http.PersistentConnection.application_1.0.0.ear

PROFILE: UnixProfile.substvar

2. Click the **Browse** button in the Upload A File panel to navigate to the EAR or ZIP file and click the **OK** button.

   You can upload one of following archive files:

   - **`.ear` File**

     When you upload a BusinessWorks EAR file, it uses the default value of the global variables set in TIBCO Designer.

   - **`.zip` File**

     You can create a ZIP file that contains an EAR file and optionally an XML properties file. The XML file can contain and define all the published configurations, path name, global variables, and so on.

     For more information , see Creating a Deployment Configuration XML File.

   You can also add a .cfg file that contains a list of value pair *property=value*, where the *property* is the global variable name and the *value* is the value of global variable. These values are used to substitute the global variables at deployment time.

A `CustomFolder.properties` file put in the `.zip` archive can specify the deployment path. For example, create the `CustomFolder.properties` file with content "`ApplicationFullPath=`*`aaa/bbb/ccc`*".

When the deployment path is not specified then the deployed EAR/application files can be found in the root folder level of Application Management in TIBCO Administrator.

When the EAR application is deployed to the BusinessWorks runtime the project files are placed in the folder structure according to the file folder structure defined in the properties file, xml file, or EAR in the zip file.

In previous releases of this Enabler, when the deployment path was not specified then the deployed BusinessWorks project EAR and application files could be found in the TIBCO Administrator directory shown:
`Silver Fabric/`*`<BWComponentName>/<TLM_Name>/<AppName>`*

These files can also be uploaded separately and then deployed, undeployed, started, and stopped by HTTP REST request. Refer to the section on Continuous Deployment - Deploy Archives Directly to Endpoints for more information.

For details about the creation of the *DeploymentConfig*`.xml` file, refer to the TIBCO Runtime Agent documentation, *Scripting Deployment User's Guide*.

Alternatively, you can also deploy applications from TIBCO Administrator or using the AppManage Domain utility.

Creating a Deployment Configuration XML File

Create a deployment configuration properties XML file for either an EAR deployment or for an archive continuous deployment.

Procedure

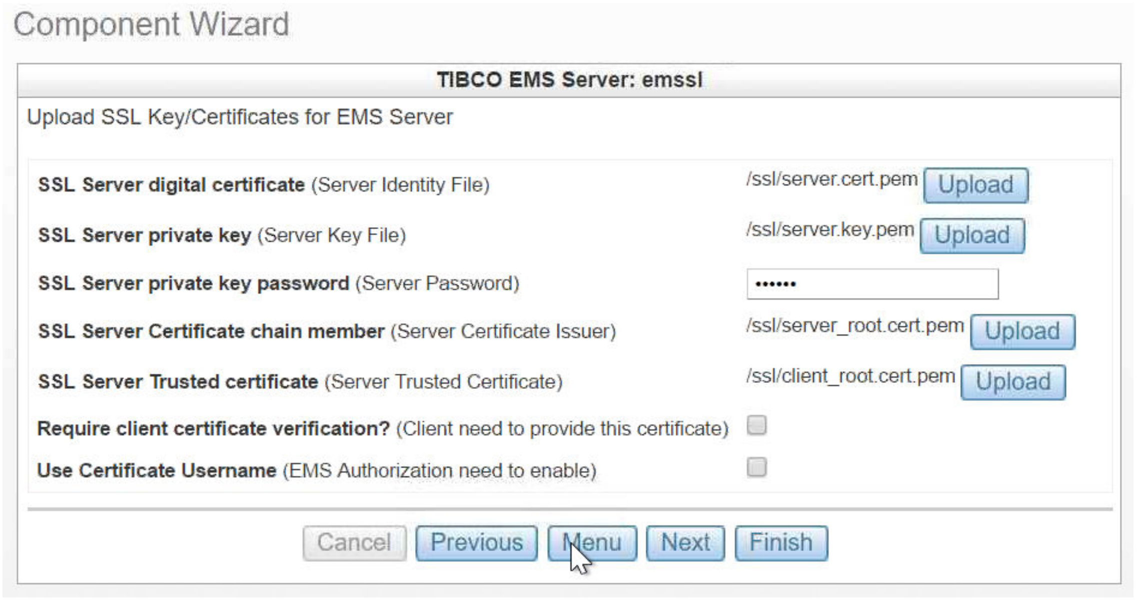
1. In *`TIBCO_HOME`*`/tra/`*`tra_version`*`/bin`, run the following command:
   ```
   AppManage -export -ear EarFile.ear -out DeploymentConfig.xml
   ```
2. Edit the file and set the values for the deployment.
3. Create an archive file with the file *`EarFile`*`.ear` and *`DeploymentConfig`*`.xml`.

Upload EMS SSL Certification

To make the domain communication though EMS is secure, SSL support is provided for the EMS that is used as a transport for TIBCO Administrator. You can configure EMS with SSL settings, on the component configuration page. You can upload the certificate and private key files with below options:

- Upload SSL Server Trusted Certificate
- Upload SSL Digital Certificate
- Upload SSL Server Private Key

SSL Key/Certificates for EMS Server

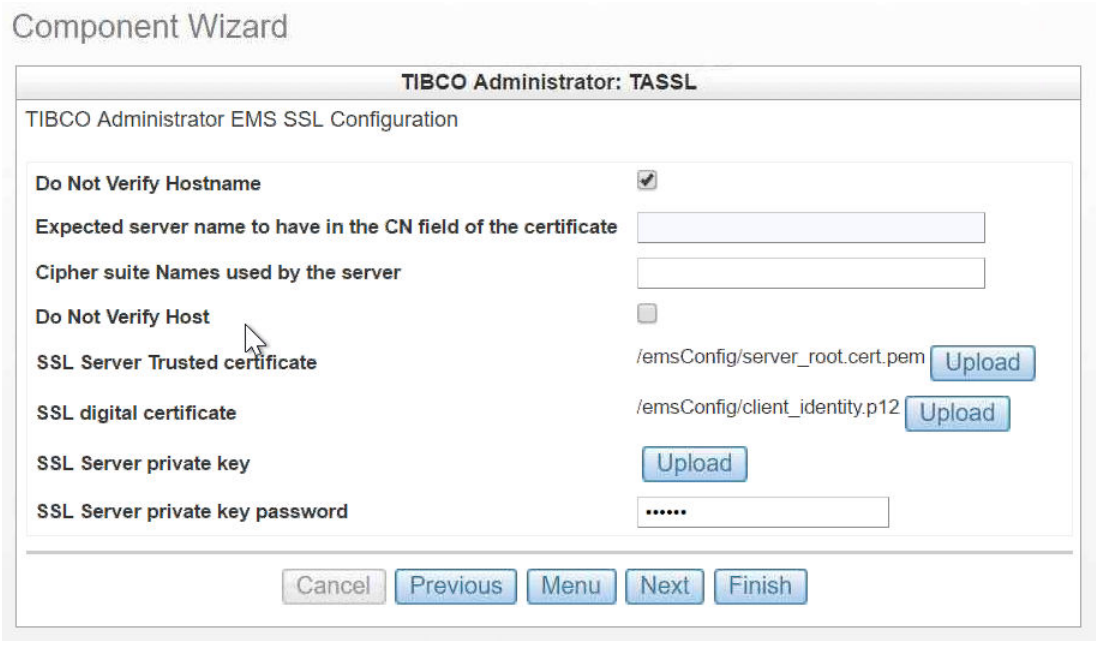

Component Wizard

TIBCO EMS Server: emssl

Upload SSL Key/Certificates for EMS Server

SSL Server digital certificate (Server Identity File)	/ssl/server.cert.pem Upload
SSL Server private key (Server Key File)	/ssl/server.key.pem Upload
SSL Server private key password (Server Password)	••••••
SSL Server Certificate chain member (Server Certificate Issuer)	/ssl/server_root.cert.pem Upload
SSL Server Trusted certificate (Server Trusted Certificate)	/ssl/client_root.cert.pem Upload
Require client certificate verification? (Client need to provide this certificate)	☐
Use Certificate Username (EMS Authorization need to enable)	☐

TIBCO Administrator EMS SSL Configuration


Component Wizard

TIBCO Administrator: TASSL

TIBCO Administrator EMS SSL Configuration

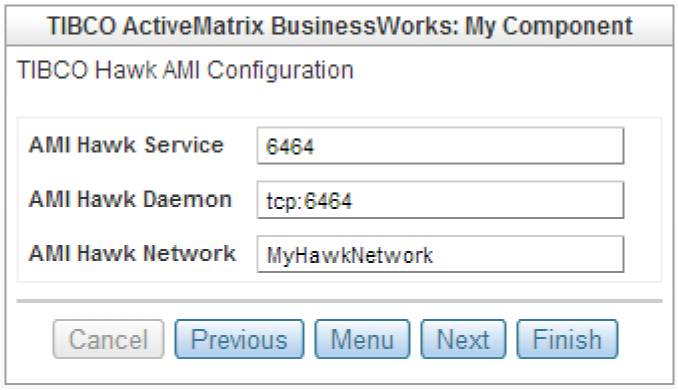
Do Not Verify Hostname	✓
Expected server name to have in the CN field of the certificate	
Cipher suite Names used by the server	
Do Not Verify Host	☐
SSL Server Trusted certificate	/emsConfig/server_root.cert.pem Upload
SSL digital certificate	/emsConfig/client_identity.p12 Upload
SSL Server private key	Upload
SSL Server private key password	••••••

Hawk Application Management Interface (AMI) Configuration (optional)

The TIBCO Hawk AMI Configuration page defines how the BusinessWorks Component uses TIBCO Rendezvous with TIBCO Hawk Microagents (HMA). Even when EMS is used as the primary transport, HMA still uses Rendezvous as the transport.

In the TIBCO Hawk AMI Configuration window, enter values in each field.

TIBCO Hawk AMI Configuration


TIBCO ActiveMatrix BusinessWorks: My Component
TIBCO Hawk AMI Configuration
Cancel
Previous
Menu
Next
Finish

AMI Hawk Service	6464
AMI Hawk Daemon	tcp:6464
AMI Hawk Network	MyHawkNetwork

The configuration parameters in the TIBCO Hawk AMI Configuration window are as described.

AMI Hawk Service

Specifies the TIBCO Hawk port number, for example, TIBCO Rendezvous connects with TIBCO Hawk on the default port `7474`. AMI Hawk Service and AMI Hawk Daemon must be set with valid values together. Setting only one of them will result in an error.

AMI Hawk Daemon

Specifies the location of the TIBCO Hawk Daemon. A value of "`tcp:`*`yyyy`*" would correspond to a local Hawk Daemon where "yyyy" would be the port number. A Hawk Daemon located elsewhere would be specified by a value of the protocol, IP address, and port number: "`tcp:`*`xxx.xxx.xxx.xxx:yyyy`*".

The AMI Hawk Service and the AMI Hawk Daemon ports may be the same or different. By default, different TLMs on the same engine daemon (physical machine) are using the same RV transport (default AMI Hawk Service is `7475`, and the default AMI Hawk Daemon is `tcp:7474`). This setting makes visible all of the deployed applications on each TLM even if some applications are NOT deployed on this particular TLM.

The actual AMI Hawk Service port for the runtime component instance is incremented by an integer according to the engine instance ID. For example, if the user sets the AMI Hawk Service to `6464` and the component is instantiated to run on engine instance 1, then the service is reported in the `hawkagent.cfg` file as `6465`. Then when the component is scaled up to other engine instances the AMI Hawk Service value will be incremented higher by the Enabler automatically.

Generally, AMI Hawk Network is an empty string, and all three fields can be left empty. If all three fields are left empty then the enabler uses the default value"7475" and "tcp:7474".

HTTP Port Settings Management (optional)

The HTTP port management is done for the following BW process starter:

- SOAP/HTTP Event Source
- SOAP/HTTP Service
- HTTP Receiver Activity

To avoid port conflicts on the TIBCO Silver Fabric engines where TIBCO Active Matrix BusinessWorks™ will be running, TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks provides a field named: "HTTP Base value for the HTTP request Activity and SOAP/HTTP Web Service Activity" to set the HTTP base value.

This defines a reserved range of ports that should not be used elsewhere.

Edit the base values of the "*HTTP Base Value...*" and "*HTTP Port Increase Value...*" variables on the HTTP Port Management page.

The HTTP base port value is used to calculate derived port values for Web Services using SOAP over HTTP transport and other HTTP activities as required by your BusinessWorks project.

HTTP Port = "*HTTP Base Value...*" + "*HTTP Port Increase Value...*"* (*HttpActivityNumber*(1,2,3,or more)-1) + *EngineInstanceValue*

For example, you have a BusinessWorks project configured using two HTTP activities. The HTTP activities are running on the Engine instance whose number is 2. The "*HTTP Base Value...*" is the default value: `8200`. The "*HTTP Port IncreaseValue...*" is the default value: `50`.

The first port is set to `8202` which was derived from: `8200 + 50 *(1-1) + 2`

The second port is set to `8252`, derived from: `8200 + 50 *(2-1) + 2`.

You should not put different base values on different BusinessWorks components running on the same Silver® Fabric private cloud. If the values are different, you may encounter port conflicts.

If your project uses more than 50 HTTP ports, the "*HTTP Port Increase Value...*" variable must greater than the number of HTTP ports.

Configure TIBCO Hawk Agent Running Condition

Polling Period (in seconds) for detection of TIBCO Hawk Agent running verification (required)

Enter an integer to specify the number of seconds between periodic verification checks that the TIBCO Hawk Agent is still running.
If the TIBCO Hawk Agent becomes unresponsive to this verification then the process is automatically restarted.

Automatically Restart Silver Fabric Engine if TIBCO Hawk Agent fails to restart N successive times (required)

Enter an integer to specify the number of restart retries for the TIBCO Hawk Agent before the TIBCO Silver Fabric engine is restarted. A successful restart resets the count.

Force Kill of TIBCO Hawk Agent Process by Engine Daemon on Shutdown (if the regular stop is not successful)

If the regular stop does not work, then only use this option. Select this option if you want kill all the TIBCO Hawk Agent processes on shutdown. It stops all those processes those were started by engine daemon.

Uploading Hawk MicroAgent Plugin (optional)

Procedure

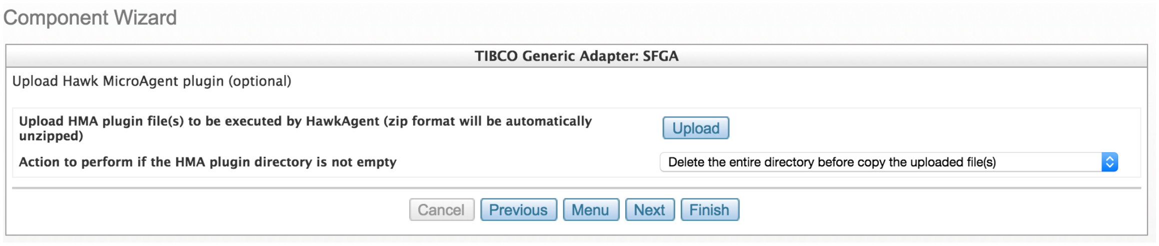
1. Upload Hawk MicroAgent plug-ins to HawkAgent as a .zip file. These Hawk microagents collect information and operate using that information. They execute specific tasks known as methods.

   The `.zip` file gets extracted into the HMA plugin directory.

2. Select one of the following options, if the directory is not empty,:

   - **Delete the entire directory before copying the uploaded files**
   - **Keep all existing files and replace the existing files with the uploaded ones**
   - Delete the entire directory before copying the uploaded files
   - Keep all existing files and replace the existing files with the uploaded ones

Uploading Hawk MicroAgent Plug-in



Component Wizard

TIBCO Generic Adapter: SFGA

Upload Hawk MicroAgent plugin (optional)

Upload HMA plugin file(s) to be executed by HawkAgent (zip format will be automatically unzipped)	Upload
Action to perform if the HMA plugin directory is not empty	Delete the entire directory before copy the uploaded file(s)

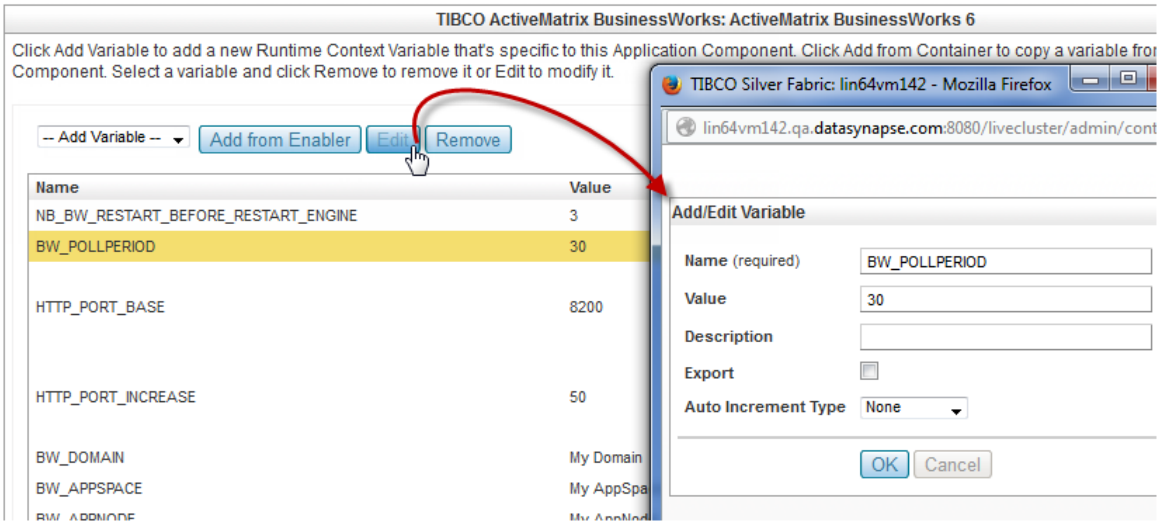
Adding or Editing Runtime Context Variables (optional)

Variables may be added to define runtime specific context variables. **String**, **Environment**, **System**, or **Encrypted** variables may be added.

Procedure

1. Select a variable type from the **Add Variable** drop down if you wish to add or edit a variable.

   Editing a Variable



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

Click Add Variable to add a new Runtime Context Variable that's specific to this Application Component. Click Add from Container to copy a variable fro
Component. Select a variable and click Remove to remove it or Edit to modify it.

Name	Value
NB_BW_RESTART_BEFORE_RESTART_ENGINE	3
BW_POLLPERIOD	30
HTTP_PORT_BASE	8200
HTTP_PORT_INCREASE	50
BW_DOMAIN	My Domain
BW_APPSPACE	My AppSpa
BW_APPNODE	My AppNod

TIBCO Silver Fabric: lin64vm142 - Mozilla Firefox

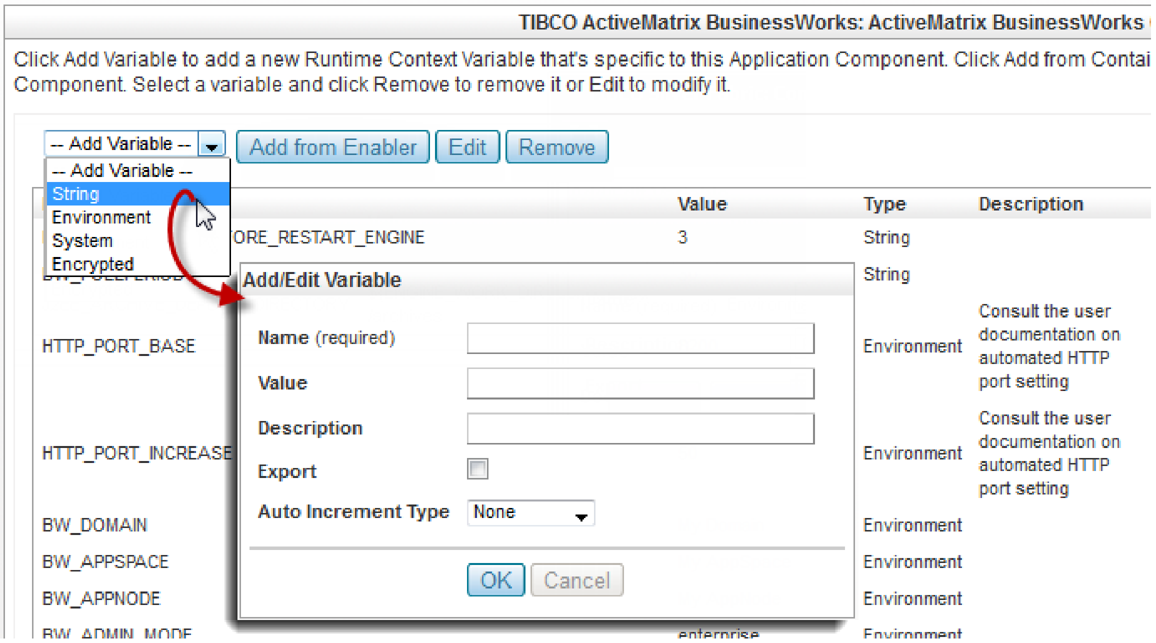
lin64vm142.qa.datasynapse.com:8080/livecluster/admin/cont

Add/Edit Variable

Name (required)	BW_POLLPERIOD
Value	30
Description	
Export	☐
Auto Increment Type	None

You can add a string variable to change pre-existing context variables like: `ARCHIVE_DETECTION_FREQUENCY`.

Add or Edit a String Variable



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks

Click Add Variable to add a new Runtime Context Variable that's specific to this Application Component. Click Add from Contai
Component. Select a variable and click Remove to remove it or Edit to modify it.

	Value	Type	Description
ORE_RESTART_ENGINE	3	String	
		String	
HTTP_PORT_BASE		Environment	Consult the user documentation on automated HTTP port setting
HTTP_PORT_INCREASE		Environment	Consult the user documentation on automated HTTP port setting
BW_DOMAIN		Environment	
BW_APPSPACE		Environment	
BW_APPNODE		Environment	
BW_ADMIN_MODE	enterprise	Environment	

Add/Edit Variable

Name (required)	
Value	
Description	
Export	☐
Auto Increment Type	None

ARCHIVE_DETECTION_FREQUENCY is the periodic interval (default value is 30 seconds) for detection of deployed archives and report to the broker. The broker uses this data to synchronize the archive with scaling rules.

These variables are not exposed in the interface elsewhere, but you can change their values. In this case it would be recommendable to set these variable values higher than the time needed to deploy and start an archive.

Changes are of course optional, because all variables have default values that are usually appropriate for the most common use cases.

Variable values from the Enabler may be changed as well.

2. Use the **Add from Enabler** button to change values of Enabler-specific context variables.

Upload, Add, Customize, or Remove a Content File

If needed, you can upload a content file (for example, the JDBC Jar file for `mysql`) specific to this component from this dialog.

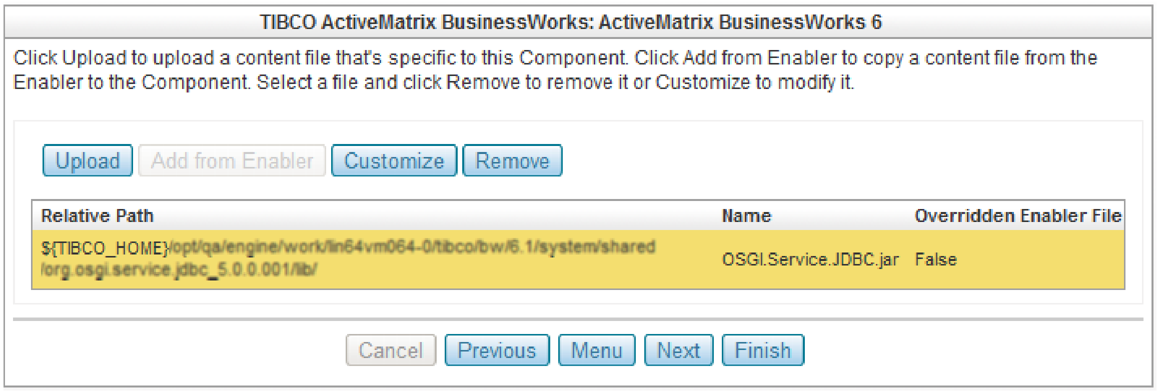
You can perform the following operations:

- **Upload**: Upload new file.
- **Add from Enabler**: Copy a content file from the Enabler to the component.
- **Customize**: Modify the content file.
- **Remove**: Delete the file.

Procedure

- Add files to a relative path associated with the BusinessWorks component that can be required for the component to run according to design.

Uploading Content Files


TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

Click Upload to upload a content file that's specific to this Component. Click Add from Enabler to copy a content file from the Enabler to the Component. Select a file and click Remove to remove it or Customize to modify it.

Relative Path	Name	Overridden Enabler File
${TIBCO_HOME}/opt/qa/engine/work/lin64vm064-0/tibco/bw/6.1/system/shared/org.osgi.service.jdbc_5.0.0.001/lib/	OSGI.Service.JDBC.jar	False

You can customize the domain related properties by clicking **Add from Enabler**. The file `AddMachine.properties` can be selected and loaded in the wizard where you can customize the properties using the Customize button. The changes to the properties are reflected in the `AddMachine.xml` file. This file is used for adding machine to the TIBCO Administrator domain.

Steps to customize the properties are as follows:

1. Select the loaded `AddMachine.properties` file and click **Customize**.
2. Uncomment the property you want to customize, and set the value after "=" sign
3. Click **OK** once you have done the customization.

Add or Remove BusinessWorks Component Statistics

You can add or remove the following statistics on 5.x component for tracking. The following table lists the available BusinessWorks statistics for 5.x components.

BusinessWorks (BW) 5.x Component Statistics

Name	Description
BW Active Processes Count	Total number of active, not paged, processes.
BW Max Processes Count	Maximum number of running process instances.
BW Percentage of Memory Used	Percentage of used memory (used memory divides allocated memory).
BW Memory Free Bytes	Total number of bytes that are not currently in use.
BW Process Count	Total number of running process instances.
BW Number of Processes Aborted	Number of times process instances have been aborted.
BW Number of Processes Swapped	Number of times process instances have been swapped to disk.
BW Number of Processes Suspended	Number of times process instances have been suspended.

Name	Description
BW Average Elapse Execution Time	Average elapsed clock time (in milliseconds) for all successfully completed process instances.
BW Last Execution Time	Execution time (in milliseconds) of most recently completed process instance.
BW Total Number Error	Total number of errors encountered since the process engine was started.
BW Last Number Error	Total number of errors encountered since the last time statistic method was called.
BW Uptime	Elapsed time (in minutes) since the process engine was started.
BW Number of Processes Created	Number of process instances created.
BW Number of Processes Completed	Number of process instances that have been successfully completed.

Adding Allocation Rule Settings

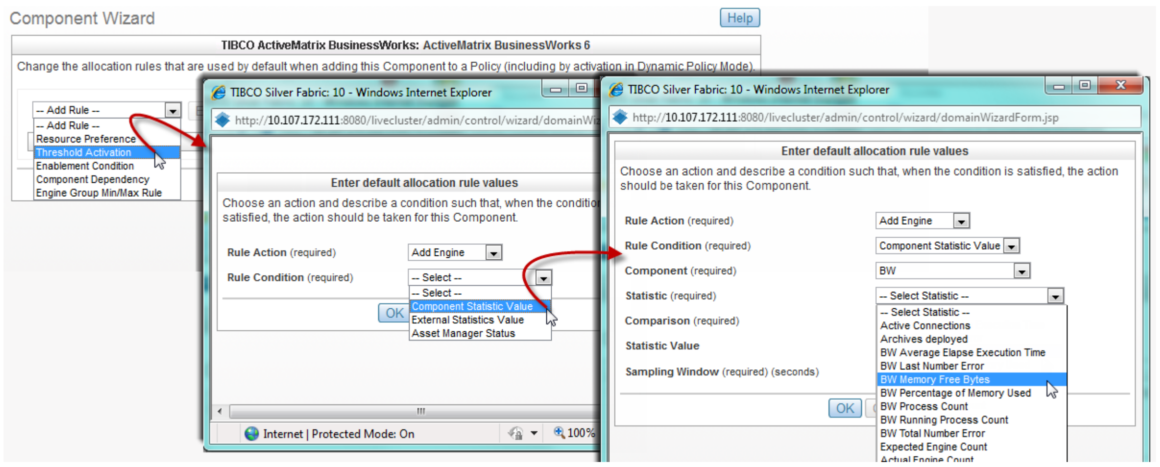
Add rules to specify and set component behavior.

Add rules to do the following actions:

- Specify Resource Preferences
- Set Thresholds for Activation
- Set Enablement Conditions
- Specify component dependency
- Set Engine Group Minimums or Maximums

Each rule selection brings up a slightly different dialog window that allows property selection of a tracked engine or component archive statistic to be evaluated according to a logical operator and a value you specify to define an action. In some cases the Component Archive Statistics may be selected as is shown in the Figure.

Using BW Statistics for Threshold Activation



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

Change the allocation rules that are used by default when adding this Component to a Policy (including by activation in Dynamic Policy Mode).

http://10.107.172.111:8080/livecluster/admin/control/wizard/domainWizardForm.jsp

Enter default allocation rule values

Choose an action and describe a condition such that, when the condition is satisfied, the action should be taken for this Component.

Rule Action (required)	Add Engine
Rule Condition (required)	Component Statistic Value
Component (required)	BW
Statistic (required)	-- Select Statistic --
Comparison (required)	
Statistic Value	
Sampling Window (required) (seconds)	

More information on using statistics for micro-scaling or archive scaling is available in the *TIBCO Silver® Fabric Cloud Administration Guide* and more about Component Archive scaling within a stack is covered in this guide.

Finishing Configuring the Component

All other screens are generic for all Silver® Fabric Enablers. The configuration is optional for BusinessWorks components.

Refer to *TIBCO Silver® Fabric User's Guide* for additional information on these configuration.

Procedure

1. Click **Finish**, and make sure that the component is published to make it available to create a stack.
2. Select **Publish Component** in the Actions list located at the line of the component you just created.

Creating BusinessWorks Components with 6.x Distributions

Using the TIBCO Silver Fabric component wizard, you can create and configure the TIBCO ActiveMatrix BusinessWorks component as follows:

- Configure a TIBCO ActiveMatrix BusinessWorks instance for publishing to TIBCO Silver Fabric Engines.
- Add a TIBCO ActiveMatrix BusinessWorks Environment from the following types:
  - **Local**: customize the domain, appspace, and appnode. Upload BusinessWorks application from the component. You can use the BusinessWorks component without depending on any component.
  - **Enterprise**: use ActiveSpaces or EMS/DB or FTL/DB for persistence and transport. Register with TIBCO Enterprise Administrator and either use dependent TEA component or use standalone TEA server. Upload application from the component, then publish and deploy the BusinessWorks component depending on TEA component or a standalone TEA server. You can deploy the BusinessWorks application and log into the TEA to manage agent, domain, appspace, appnode, and applications.

    Even if you do not upload the BusinessWorks projects, you can still publish the BusinessWorks projects on the machine where BusinessWorks registers to TIBCO Domain using either TIBCO Enterprise Administrator or another administrative interface.

Procedure

1. Enter name and description for the new BusinessWorks component.

   See Specifying the Component Name for more details on configuration options.
2. Enter values for each parameter field as applicable, in the Choose Product distribution page.

   See Choosing the TIBCO Product Distribution Version for more details.
3. Configure the environment of TIBCO ActiveMatrix BusinessWorks.

   See Configuring the Environment for more details on configuration.
4. Enter the values in each parameter field in the Component Wizard page for the ActiveMatrix BusinessWorks environment.

   See Specifying the Domain, AppSpace, AppNode, and APPNode HttpPort Base Configuration for more details.
5. (Optional) Upload a BusinessWorks project.

   See Uploading a BusinessWorks Project for more details.
6. (Optional) Enter the values for the variables on the HTTP Port Management Page.

   See HTTP Port Settings (optional) for more details.
7. (Optional) Enable or disable LDAP Authentication Configuration for Bwagent REST API.

   See LDAP Authentication for more details.
8. Configure the running condition parameters in AppSpace and AppNode Running Condition page.

   See AppNode or bwagent Running Condition Configuration for details on configuration options.
9. (Optional) Configure the runtime parameters for the BusinessWorks component.

   See Review and Edit Runtime Context Variables (optional) for details.
10. Enter values in the fields as applicable to define and set runtime-specific context variables.

    See Adding a String, Environment, System, or Encrypted Variable for more details.

11. Add a string variable to change preexisting context variables.

    See String Variable to Change an Existing Context Variable for more details.
12. (Optional) Upload a content file specific to the BusinessWorks component.

    See Upload, Add, Customize, or Remove a Content File (optional) for more details.
13. Add or remove BusinessWorks Component statistics for tracking.

    See Add or Remove BusinessWorks Component Statistics for available list for 6.x components.
14. The following tasks of the Component Wizard are generic for all Silver® Fabric Enablers. The configuration of these is optional for BusinessWorks component 6.x.

    - Add a component to a policy
    - Add allocation rule settings
    - Apply allocation constraints
    - Configure component options

    Refer to *TIBCO Silver® Fabric User's Guide* for additional information on these configurations
15. Add, edit, or remove ECMA or python scripts from the doc test page.

    See Uploading, Editing, or Removing Script Files for more details.
16. Proceed through the component wizard until you can click **Finish** to save your changes.

    If you abandon a component wizard creation or configuration changes and the Silver Fabric Administrator interface times out your session before you click **Finish**, your changes are not saved.

    After you click **Finish**, make sure that the component is published to make it available to update or create a stack.
17. Publish the component after creating it.

    See Publishing a Component for details on publishing the component.

Specifying the Component Name

Procedure

1. On the TIBCO Silver® Fabric Administration Tool Web UI, select **Stacks** > **Components** .
2. On the **Components** page, in the **Global Actions** list, select **Create New TIBCO ActiveMatrix BusinessWorks Component**.
3. Provide a **Name** and **Description** for the component and then click **Next**.

Choosing the TIBCO Product Distribution Version

After naming the component, choose the distribution to be published by the component. Selection of the distribution significantly changes what pages and configurations appears for setting values in the **Component Wizard**.

Procedure

1. For **TIBCO_ActiveMatrix_BW_distribution**, select the most recent release.
2. For **TIBCO_sunec_distribution**, select a more recently released version.

It is a required field. If you do not see a more recent version, you must install the Elliptical Cryptography distribution.

The html file opens http://public.tibco.com from where you can download the compressed file.

3. For **TIBCO_FTL_distribution**, select the most recent FTL distribution.

The FTL distribution is visible only in **Choose Optional Distribution version** page, only when TIBCO_ActiveMatrix_BW_distribution version is 6.3.2.0.0 or higher

4. Click **Next**.

Configuring the Environment

TIBCO ActiveMatrix BusinessWorks can be run in the Enterprise mode or Local mode.

Procedure

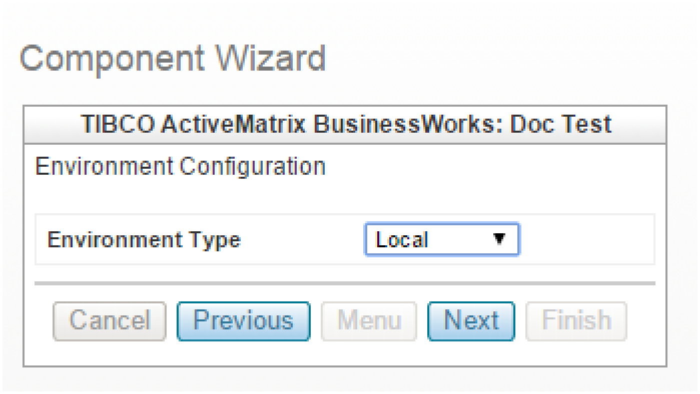
- Choose the environment type: **Enterprise** or **Local.**

  Local Mode

  In **Local mode**, bwadmin modifies the local file system directly instead of delegating the work to a bwagent (no bwagent instance is started). The local mode does not provide data storage and the runtime entities are created in the file system.

  This mode is useful for developers during development and testing cycles.

  Environment Configuration - Local Environment Type



Component Wizard

TIBCO ActiveMatrix BusinessWorks: Doc Test

Environment Configuration

Environment Type: Local

  When you select the local environment type, the options on the component wizard dialog changes so that there is no need to enter any value. You can directly skip to Specifying the Domain, AppSpace, AppNode, and APPNode HttpPort Base Configuration.

  Enterprise Mode

  In **Enterprise mode**, bwadmin communicates with the bwagent. The bwagent can communicate across machines and can be configured to form a bwagent network. Instead of working on the file system directly, bwadmin sends commands to the bwagent. The bwagent dispatches the command to the targeted agent. That agent then completes the command on the local file system as applicable.

a) In **Environment Type**, select **Enterprise**.
b) Select the **Create Domain, AppSpace, AppNode** option to take advantage of automated creation of the first domain, AppSpace, and AppNode for each of your ActiveMatrix BusinessWorks runtime instances on Silver Fabric engines. Clearing the check box disables creation of the domain, AppSpace, and AppNode.
c) In **Domain data persistence and communication transport layer**, select the messaging transport from one of the following options:

- **ActiveSpaces** TIBCO ActiveSpaces offers fast computing by maintaining data in memory with persistent data stored on the local file system. See Using TIBCO ActiveSpaces as a Transport (Option 1) for more details.
- **Database/EMS** provides the option of using an external database for data persistence and TIBCO Enterprise Message Service™ (EMS) for the communication transport layer. See Using Database/EMS as a Transport (Option 2) for more details.
- **Database/FTL** provides the option of using an external database for data persistence and TIBCO FTL ® for the communication transport layer. See Using Database/FTL as Transport for more details.

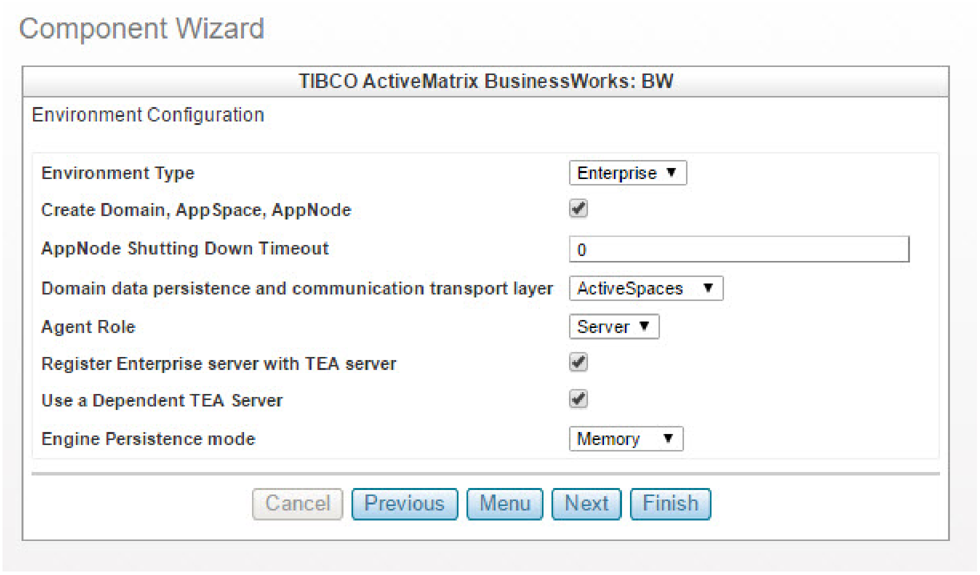
Using TIBCO ActiveSpaces as a Transport (Option 1)

To use ActiveSpaces as the TIBCO ActiveMatrix BusinessWorks transport, perform the following procedure:

- Specify the agent role as either a Server or a Client
- Set the Minimum Seeder Count and Quorum Size

You can make your TIBCO ActiveMatrix Businessworks bwagent process run either as a server or a client. Use one component for each role. However, many implementations instantiate multiple TIBCO ActiveMatrix BusinessWorks bwagent processes working in both the roles.

Environment Configuration - Enterprise Environment Type


Component Wizard
TIBCO ActiveMatrix BusinessWorks: BW
Environment Configuration
Cancel
Previous
Menu
Next
Finish

Environment Type	Enterprise
Create Domain, AppSpace, AppNode	✓
AppNode Shutting Down Timeout	0
Domain data persistence and communication transport layer	ActiveSpaces
Agent Role	Server
Register Enterprise server with TEA server	✓
Use a Dependent TEA Server	✓
Engine Persistence mode	Memory

Procedure

1. In **Environment Type**, ensure that **Enterprise** is selected.
2. Ensure that the **Create Domain, AppSpace, AppNode** option is selected.
3. Specify the **AppNode Shutting Down Timeout**. By using this option you can specify the timeout in minutes after which the AppNode shuts down forcefully, irrespective of the state of the AppNode or the applications. The default value is '0'.
4. In **Domain data persistence and communication transport layer**, select **ActiveSpaces** as the messaging transport.
5. Specify a role for bwagent as: **Server** or **Client.**

Server

When bwagent is configured to run as a server, it creates a file-based data store and provides remote management capability. The number of bwagents that must be configured to run as a server depends on the total number of bwagents in the deployment and the degree of fault tolerance required. Refer to the TIBCO® ActiveMatrix BusinessWorks™ documentation and the comments in bwagent.ini located in *SFBW_HOME*\bw\N.N\config for more information.

Selecting the role of agent as server enables the **Register Enterprise server with TEA server** option in the Environment Configuration dialog box.

Client

When bwagent is configured to run as a client, data is not stored locally and clients are managed remotely by bwagent instances performing in the server role.

6. When the role of agent is server, you can see that the **Register Enterprise server with TEA server** option is selected.

   TIBCO Enterprise Administrator (TEA) server provides an Administrator UI which can be used to create, view, and monitor runtime entities. The `bwagent` interacts with the TIBCO Enterprise Administrator server through a TEA agent. Only one TEA agent can be registered with a `bwagent` at a time. When the TEA agent is registered with `bwagent`, the `bwagent` is displayed in the Administrator UI. By using the Administrator UI, you can manage and monitor runtime entities and perform almost all `bwadmin` administrative tasks.

7. Ensure that the **Use a Dependent TEA Server** option is selected**.**

   Selecting this option creates a TEA component for TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator and also sets the dependency of BusinessWorks component on the TEA component. The TIBCO BusinessWorks component then acquires the URL of the TEA server and automatically registers to the TEA server.

   If this option is not selected, you must first set up and start a TEA server manually and then provide the URL of the running TEA server in the **TIBCO TEA Server Configuration** page in the BusinessWorks component creation wizard.

   You must take advantage of the automatic registration and administrative features provided by TIBCO Silver® Fabric Enabler for TIBCO® Enterprise Administrator.

   When the component bwagent is set to run in the server agent role, select **Register Enterprise Server with TEA Server** and **Use a Dependent TEA Server** to leverage the features provided by TIBCO® Enterprise Administrator (TEA).

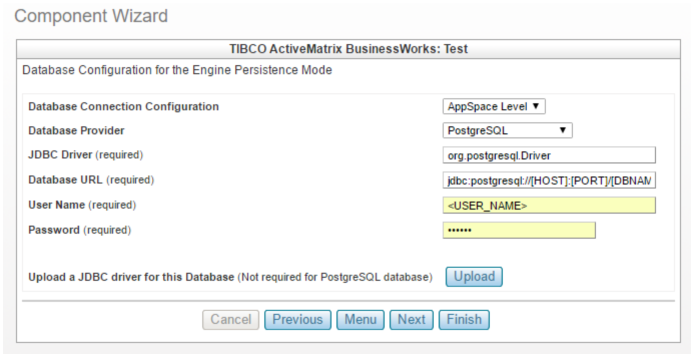
8. When the role of agent is server, add a TEA component to your BusinessWorks stack and add a TEA server dependency to your BW server component. Refer to the TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator documentation for more information.

9. Select the **Engine Persistence mode** from the drop-down.

   - **Memory** This is the default mode and does not require any special configuration. In the memory mode, there is no persistence and the engines are unaware of the existence of each other. As a result, there is no collaboration between engines.

   - **Datastore** If you select this mode, then you must provide the database configuration as this mode uses a database to provide persistence. The database deatils must be configured on the **Database Configuration for The Engine Persistence Mode** page that is displayed.

Database Configuration for The Engine Persistence Mode



Component Wizard

TIBCO ActiveMatrix BusinessWorks: Test

Database Configuration for the Engine Persistence Mode

Database Connection Configuration	AppSpace Level ▾
Database Provider	PostgreSQL ▾
JDBC Driver (required)	org.postgresql.Driver
Database URL (required)	jdbc:postgresql://[HOST]:[PORT]/[DBNAM
User Name (required)	<USER_NAME>
Password (required)	••••••
Upload a JDBC driver for this Database (Not required for PostgreSQL database)	Upload

Cancel | Previous | Menu | Next | Finish

There are two **Database Connection Configuration** levels:

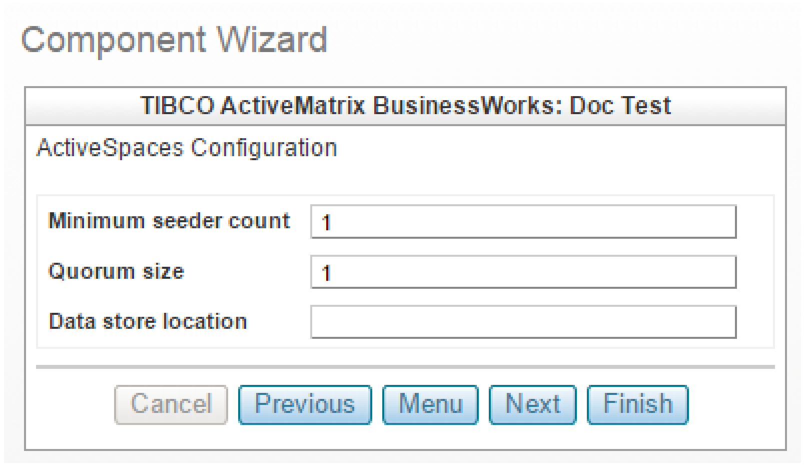
AppSpace and **AppNode Levels**. For more information refer Specifying the Domain, AppSpace, AppNode, and APPNode HttpPort Base Configuration.

- **Group** The group mode uses a database and a group provider to provide persistence and collaboration between the AppNodes. In the group mode, the engines are aware of each other's existence and they can collaborate and work together to enable features such as checkpointing and managed fault tolerance. The Group mode displays **The group provider technology for the engine** property in the **Environment Configuration** page. For more information refer Group Provider Technology for the Engine .
- **FTGroup** The database is not needed when the Engine Persistence mode is FTGroup. Also The **FTGroup** mode displays **The group provider technology for the engine** property in the **Environment Configuration** page. For more information refer Engine Persistence mode FTGroup .

10. Specify **The engine group name (Group_[DomainName]_[AppSpaceName])** property. This is an optional property and it specifies name of the BusinessWorks engine group. If this property is not specified, then the group name defaults to "*Group_<DomainName>_<AppSpaceName>*"
11. Click **Next**.

    When the role of agent is server, the ActiveSpaces Configuration dialog-box is displayed.

    ActiveSpaces Configuration



Component Wizard

TIBCO ActiveMatrix BusinessWorks: Doc Test

ActiveSpaces Configuration

Minimum seeder count	1
Quorum size	1
Data store location	

Cancel | Previous | Menu | Next | Finish

You can configure the minimum seeder count and the quorum for the `bwadmin` cluster to run properly. You can also change the file directory for ActiveSpaces to store persistent data. These settings are not necessary for the BusinessWorks client.

12. Specify **Minimum seeder count**.

It determines the minimum number of bwagents that are configured to run in the **Server** agent role, for the `bwadmin` cluster to function. You must change the default value in a multi machine/multi agent setup.

> As a general guideline the Minimum Seeder Count can be expressed as a function of at least half the total number of BW servers.
>
> `minSeederCount = Floor(n/2) + 1, n>2` where n is the number of bwagents configured to run in the server agent role and `n` must be odd.
>
> Degree of fault tolerance is: n - minSeederCount
>
> For example, if five bwagents are set to run in the server agent role, the minSeederCount=3, and two of the bwagents can be stopped without impacting the bwadmin cluster
>
> Another example, if three bwagents are set to run in the server agent role, the minSeederCount=2, and only one of the bwagents can be stopped without impacting the bwadmin cluster.

13. Specify **Quorum size**.

    It specifies the minimum number of `bwagents`, configured to be run in the server agent role. These `bwagents` must be started to recover the data from the local files when all the `bwagents` which were configured to run as servers were brought down and restarted. The default value has to be changed in a multi machine or multi agent setup.

    > Set the quorum size to the number of bwagents configured to run as servers.
    >
    > For example, if we configure five bwagents to run as servers with a quorum of "5", the first four bwagents do not start until the fifth bwagent is also started.
    >
    > In this example, if one of the agents has to be abandoned for some reason like hardware failure, the quorum size must be changed to "4" before restarting the group of bwagents.

14. Set the **Data store location**.

    To run `bwagent` in the server agent role, a data store location is used. The default path for the data store location is:

    ```
    ${TIBCO_HOME}/bw/6.2/domains/.datastore
    ```

    > The data store location can be changed, but make sure to use "/" as the path separator in the folder path and use *${TIBCO_HOME}* as the prefix to specify the relative location on the TIBCO Silver Fabric engine.

15. Click **Next**.

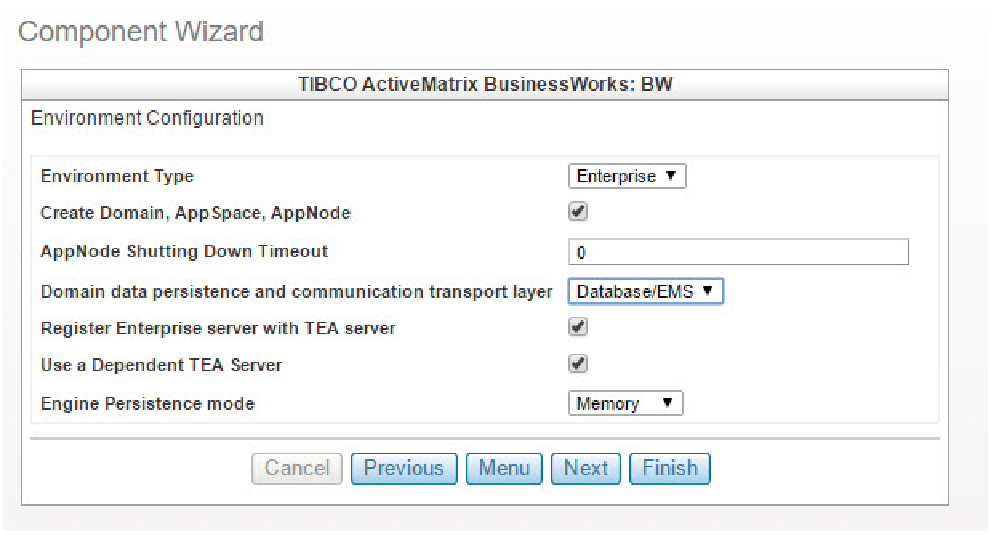
Using Database/EMS as a Transport (Option 2)

Using the Database/EMS option, you can use an external database for data persistence and TIBCO Enterprise Message Service™ (EMS) for the communication transport layer.

Procedure

1. In **Environment Type**, ensure that **Enterprise** option is selected.
2. See that the **Create Domain, AppSpace, AppNode** option is selected.
3. Specify the **AppNode Shutting Down Timeout**. By using this option you can specify the timeout in minutes after which the AppNode shuts down forcefully, irrespective of the state of the AppNode or the applications. The default value is '0'.
4. In **Domain data persistence and communication transport layer**, select **Database/EMS** as the messaging transport.

Environment Configuration



Environment Configuration

Environment Type	Enterprise
Create Domain, AppSpace, AppNode	✓
AppNode Shutting Down Timeout	0
Domain data persistence and communication transport layer	Database/EMS
Register Enterprise server with TEA server	✓
Use a Dependent TEA Server	✓
Engine Persistence mode	Memory

5. Make sure that the **Register Enterprise server with TEA server** option is selected.

   TIBCO Enterprise Administrator (TEA) server provides an Administrator UI which can be used to create, view, and monitor runtime entities. The `bwagent` interacts with the TIBCO Enterprise Administrator server through a TEA agent. Only one TEA agent can be registered with a `bwagent` at a time. When the TEA agent is registered with `bwagent`, the `bwagent` is displayed in the Administrator UI. Using Administrator UI, you can manage and monitor runtime entities and perform almost all `bwadmin` administrative tasks.

6. Ensure that the **Use a Dependent TEA Server** option is selected**.**

   Selecting this option creates a TEA component for TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator and also sets the dependency of BusinessWorks component on the TEA component. TIBCO BusinessWorks component then acquires the URL of TEA server and automatically registers to the TEA server.

   If the option is not selected, you must first set up and start a TEA server manually and then provide the URL of the running TEA server in the **TIBCO TEA Server Configuration** page in the BusinessWorks component creation wizard.

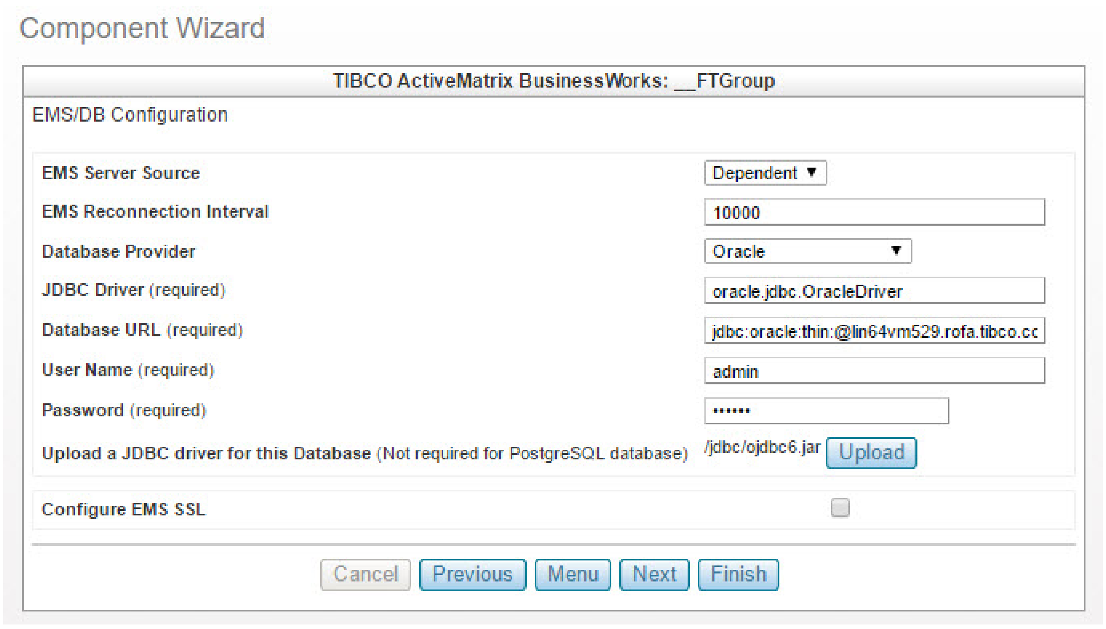
   Make sure that you take advantage of the automatic registration and administrative features provided by TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator.

   When the component bwagent is set to run in the server agent role, select the check boxes **Register Enterprise Server with TEA Server** and **Use a Dependent TEA Server** to leverage the features provided by TIBCO® Enterprise Administrator (TEA). Add a TEA component to your BusinessWorks stack and add a TEA server dependency to your BW server component. Refer to the *TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator* documentation for more information.

7. Select the **Engine Persistence mode** from the drop-down.

   - **Memory** This is the default mode and does not require any special configuration.
   - **Datastore** If you select this mode, then you must provide the database configuration on the **Database Configuration for The Engine Persistence Mode** page that appears.

Database Configuration for The Engine Persistence Mode


Component Wizard

TIBCO ActiveMatrix BusinessWorks: __FTGroup

EMS/DB Configuration

EMS Server Source	Dependent
EMS Reconnection Interval	10000
Database Provider	Oracle
JDBC Driver (required)	oracle.jdbc.OracleDriver
Database URL (required)	jdbc:oracle:thin:@lin64vm529.rofa.tibco.cc
User Name (required)	admin
Password (required)	••••••
Upload a JDBC driver for this Database (Not required for PostgreSQL database)	/jdbc/ojdbc6.jar
Configure EMS SSL	☐

There are two **Database Connection Configuration** levels:

AppSpace and **AppNode Levels**.For more information refer Specifying the Domain, AppSpace, AppNode, and APPNode HttpPort Base Configuration.

- **Group** The Group mode displays **The group provider technology for the engine** property in the **Environment Configuration** page. For more information refer Group Provider Technology for the Engine .
- **FTGroup** The database is not needed when the Engine Persistence mode is FTGroup. Also The **FTGroup** mode displays **The group provider technology for the engine** property in the **Environment Configuration** page. For more information refer Engine Persistence mode FTGroup .

8. Specify **The engine group name (Group_[DomainName]_[AppSpaceName])** property. This is an optional property and it specifies name of the BusinessWorks engine group. If this property is not specified, then the group name defaults to "*Group_<DomainName>_<AppSpaceName>*"
9. Click **Next**.

Result

When your **Domain data persistence and communication transport layer** is **Database/EMS**, then you can specify the **EMS Reconnection Interval** The **EMS Reconnection Interval** specifies the interval, the bwagent tries to reconnect with the EMS server in case the connection is lost. The default time interval is 10 seconds, or 10000 milliseconds.

Using Database/FTL as a Transport: (Option 3)

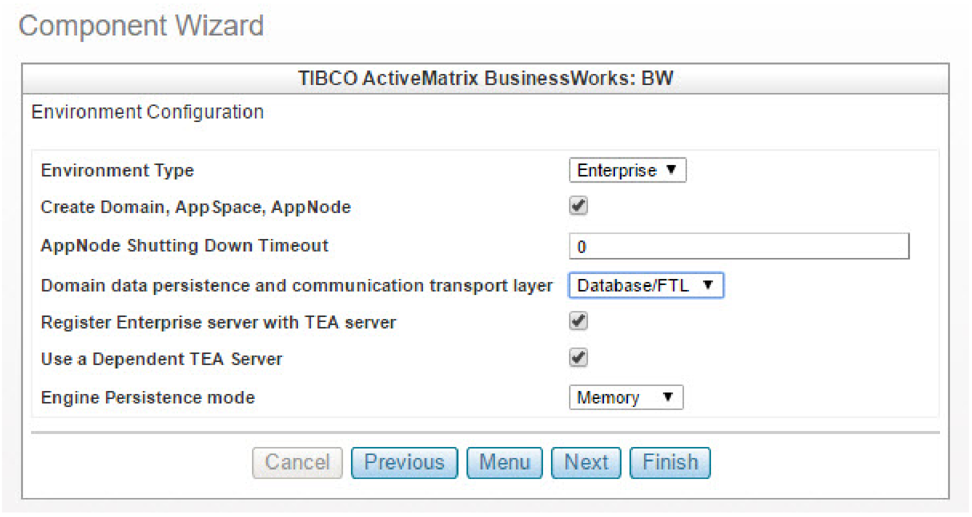
By using the Database/FTL option, you can use an external database for data persistence and TIBCO FTL for the communication transport layer.

Procedure

1. In **Environment Type**, ensure that **Enterprise** option is selected.
2. See that the **Create Domain, AppSpace, AppNode** option is selected. Specify the **AppNode Shutting Down Timeout**. By using this option you can specify the timeout in minutes after which the AppNode shuts down forcefully, irrespective of the state of the AppNode or the applications. The default value is '0'.

3. In **Domain data persistence and communication transport layer**, select **Database/FTL** as the messaging transport.

   Environment Configuration



Environment Configuration

Environment Type	Enterprise ▼
Create Domain, AppSpace, AppNode	✓
AppNode Shutting Down Timeout	0
Domain data persistence and communication transport layer	Database/FTL ▼
Register Enterprise server with TEA server	✓
Use a Dependent TEA Server	✓
Engine Persistence mode	Memory ▼

4. Ensure that the **Register Enterprise server with TEA server** option is selected.

   TIBCO Enterprise Administrator (TEA) server provides an Administrator UI which can be used to create, view, and monitor runtime entities. The `bwagent` interacts with the TIBCO Enterprise Administrator server through a TEA agent. Only one TEA agent can be registered with a `bwagent` at a time. When the TEA agent is registered with `bwagent`, the `bwagent` is displayed in the Administrator UI. Using Administrator UI, you can manage and monitor runtime entities and perform almost all `bwadmin` administrative tasks.

5. Ensure that the **Use a Dependent TEA Server** option is selected**.**

   Selecting this option creates a TEA component for TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator and also sets the dependency of BusinessWorks component on the TEA component. TIBCO BusinessWorks component then acquires the URL of TEA server and automatically registers to the TEA server.

   If the option is not selected, you must first set up and start a TEA server manually and then provide the URL of the running TEA server in the **TIBCO TEA Server Configuration** page in the BusinessWorks component creation wizard.

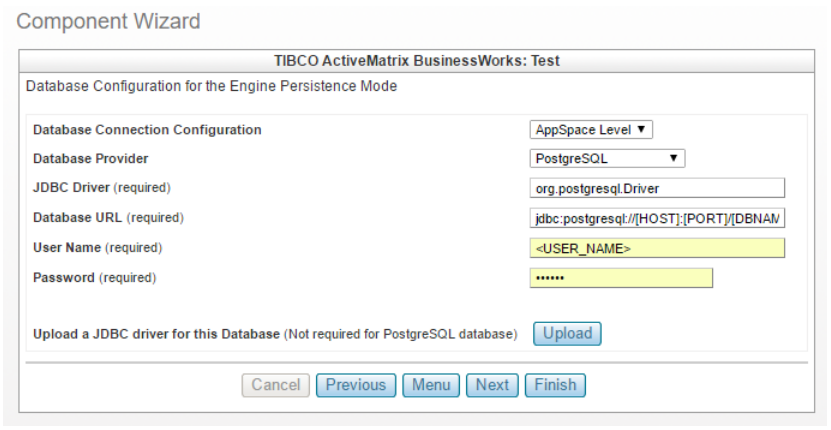
   Make sure that you take advantage of the automatic registration and administrative features provided by TIBCO Silver® Fabric Enabler for TIBCO® Enterprise Administrator.

   When the component bwagent is set to run in the server agent role, select the check boxes **Register Enterprise Server with TEA Server** and **Use a Dependent TEA Server** to leverage the features provided by TIBCO® Enterprise Administrator (TEA). Add a TEA component to your BusinessWorks stack and add a TEA server dependency to your BW server component. Refer to the TIBCO Silver Fabric Enabler for TIBCO Enterprise Administrator documentation for more information.

6. Select the **Engine Persistence mode** from the drop-down.

   - **Memory** This is the default mode and does not require any special configuration.
   - **Datastore** If you select this mode, then you need to provide the database configuration on the **Database Configuration for The Engine Persistence Mode** page that is displayed.

Database Configuration for The Engine Persistence Mode



Database Configuration for the Engine Persistence Mode

Database Connection Configuration	AppSpace Level
Database Provider	PostgreSQL
JDBC Driver (required)	org.postgresql.Driver
Database URL (required)	jdbc:postgresql://[HOST]:[PORT]/[DBNAM
User Name (required)	<USER_NAME>
Password (required)	••••••
Upload a JDBC driver for this Database (Not required for PostgreSQL database)	Upload

There are two **Database Connection Configuration** levels:

AppSpace and **AppNode Levels**.For more information refer Specifying the Domain, AppSpace, AppNode, and APPNode HttpPort Base Configuration.

- **Group** The Group mode displays **The group provider technology for the engine** property in the **Environment Configuration** page. For more information refer Group Provider Technology for the Engine .
- **FTGroup** The database is not needed when the Engine Persistence mode is FTGroup. Also, the **FTGroup** mode displays **The group provider technology for the engine** property in the **Environment Configuration** page. For more information refer Engine Persistence mode FTGroup .

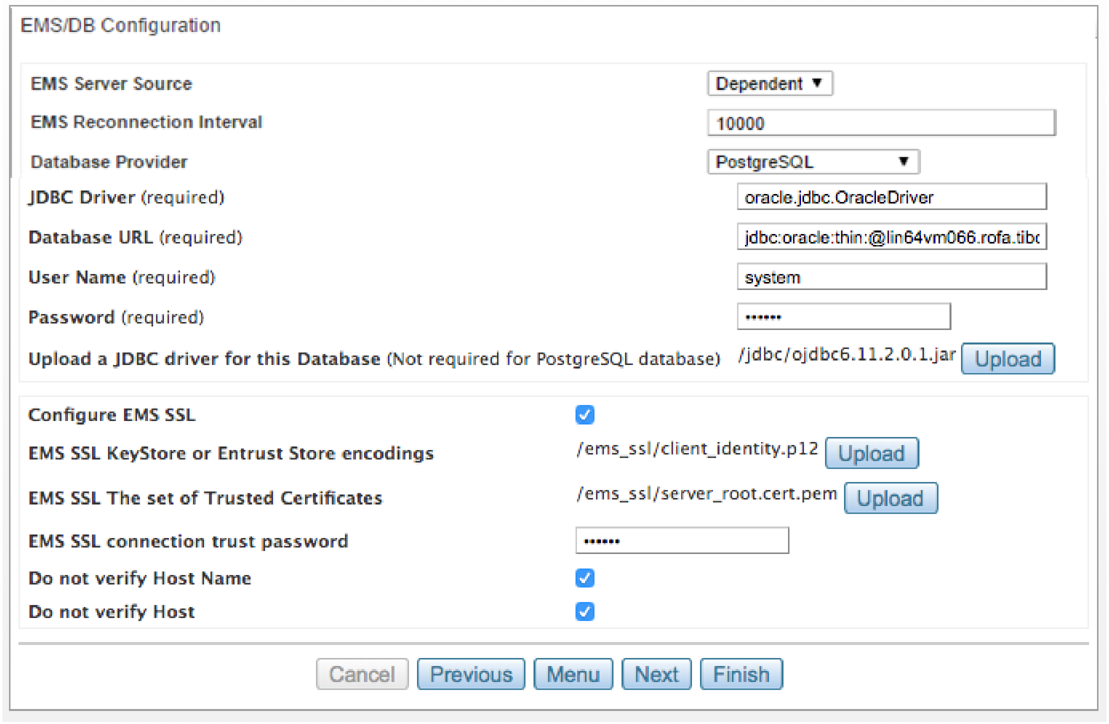
7. Specify **The engine group name (Group_[DomainName]_[AppSpaceName])** property. This is an optional property and it specifies name of the BusinessWorks engine group. If this property is not specified, the group name defaults to "*Group_<DomainName>_<AppSpaceName>*"
8. Click **Next**.

Group Provider Technology for the Engine

On the enviornment configuration page if you select the **Engine Persistence mode** as **Group** or **FTGroup**, then you need to select the **The group provider technology for the engine** from the drop-down: **EMS** or **FTL** .

- If you select **EMS** as the group provider technology then the **EMS as the Group Provider Configuration** page is displayed.

EMS as the Group Provider Configuration



EMS/DB Configuration

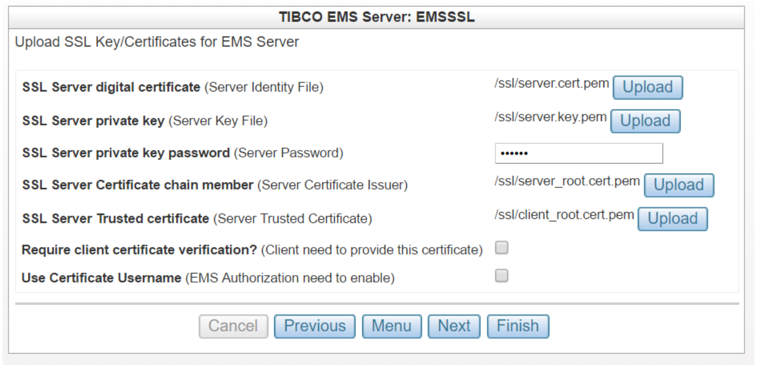
EMS Server Source	Dependent ▼
EMS Reconnection Interval	10000
Database Provider	PostgreSQL ▼
JDBC Driver (required)	oracle.jdbc.OracleDriver
Database URL (required)	jdbc:oracle:thin:@lin64vm066.rofa.tibc
User Name (required)	system
Password (required)	••••••
Upload a JDBC driver for this Database (Not required for PostgreSQL database)	/jdbc/ojdbc6.11.2.0.1.jar Upload
Configure EMS SSL	✓
EMS SSL KeyStore or Entrust Store encodings	/ems_ssl/client_identity.p12 Upload
EMS SSL The set of Trusted Certificates	/ems_ssl/server_root.cert.pem Upload
EMS SSL connection trust password	••••••
Do not verify Host Name	✓
Do not verify Host	✓

If you select EMS Server Source as **Dependent**, then a TIBCO EMS dependent component is needed. The **EMS Reconnection Interval** specifies the interval, the bwagent tries to reconnect with the EMS server in case the connection is lost. The default time interval is 10 seconds, or 10000 milliseconds.

If you select the **EMS Server Source** as **Specified** from the drop-down, then you must specify the **EMS Server URL** , **EMS User Name**, **EMS Password**.

Also, if you select the **Configure EMS SSL** check box, then you upload **EMS SSL KeyStore or Entrust Store encodings**, **EMS SSL The set of Trusted Certificates**, and provide values for **EMS SSL connection trust password**, **Do not verify Host Name**, **Do not verify Host**

TIBCO ActiveMatrix BusinessWorks Certificates

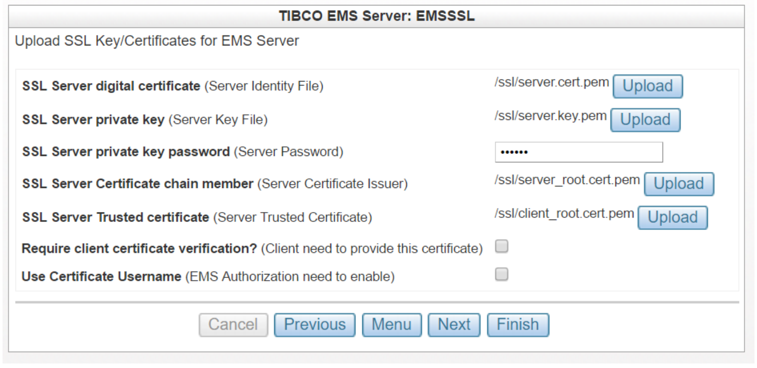


TIBCO EMS Server: EMSSSL

Upload SSL Key/Certificates for EMS Server

SSL Server digital certificate (Server Identity File)	/ssl/server.cert.pem Upload
SSL Server private key (Server Key File)	/ssl/server.key.pem Upload
SSL Server private key password (Server Password)	••••••
SSL Server Certificate chain member (Server Certificate Issuer)	/ssl/server_root.cert.pem Upload
SSL Server Trusted certificate (Server Trusted Certificate)	/ssl/client_root.cert.pem Upload
Require client certificate verification? (Client need to provide this certificate)	☐
Use Certificate Username (EMS Authorization need to enable)	☐

TIBCO EMS Certificates



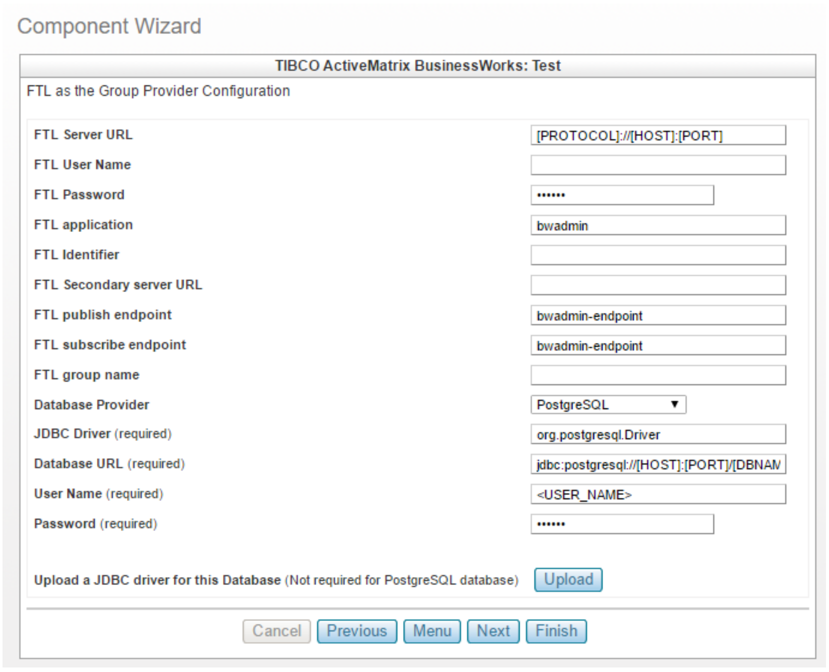
TIBCO EMS Server: EMSSSL

Upload SSL Key/Certificates for EMS Server

SSL Server digital certificate (Server Identity File)	/ssl/server.cert.pem Upload
SSL Server private key (Server Key File)	/ssl/server.key.pem Upload
SSL Server private key password (Server Password)	••••••
SSL Server Certificate chain member (Server Certificate Issuer)	/ssl/server_root.cert.pem Upload
SSL Server Trusted certificate (Server Trusted Certificate)	/ssl/client_root.cert.pem Upload
Require client certificate verification? (Client need to provide this certificate)	☐
Use Certificate Username (EMS Authorization need to enable)	☐

- If you select **FTL** as the group provider technology then the **FTL as the Group Provider Configuration** page is displayed.

FTL as the Group Provider Configuration



Component Wizard

TIBCO ActiveMatrix BusinessWorks: Test

FTL as the Group Provider Configuration

FTL Server URL	[PROTOCOL]://[HOST]:[PORT]
FTL User Name	
FTL Password	••••••
FTL application	bwadmin
FTL Identifier	
FTL Secondary server URL	
FTL publish endpoint	bwadmin-endpoint
FTL subscribe endpoint	bwadmin-endpoint
FTL group name	
Database Provider	PostgreSQL ▼
JDBC Driver (required)	org.postgresql.Driver
Database URL (required)	jdbc:postgresql://[HOST]:[PORT]/[DBNAM
User Name (required)	<USER_NAME>
Password (required)	••••••
Upload a JDBC driver for this Database (Not required for PostgreSQL database)	Upload

Procedure

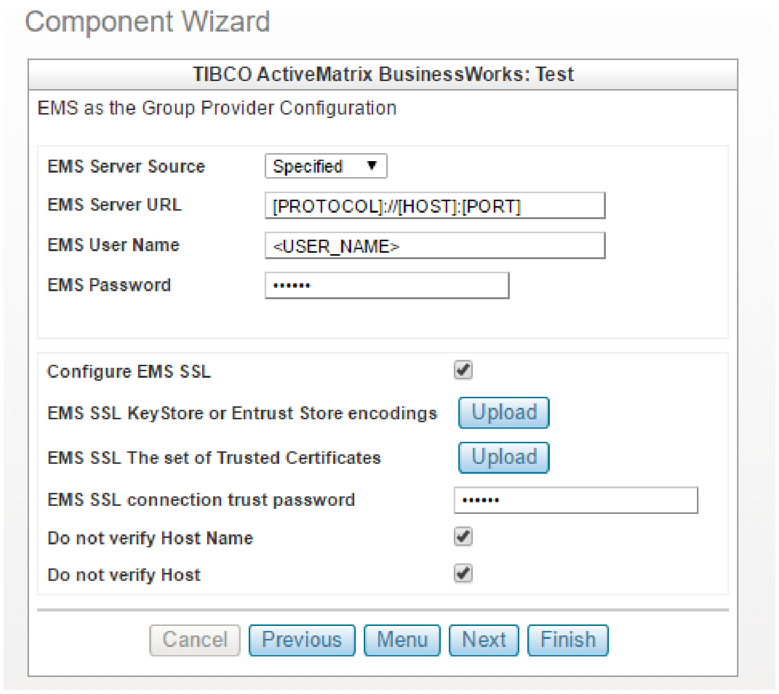
1. Specify details such as **FTL Server URL**, **FTL User Name**, **FTL Password**, and so on to use the new FTL server.
2. For the **Database Provider** option, select PostgreSQL, MySQL, Oracle, Microsoft SQL Server or DB2.
3. In the **Database URL** field, enter the URL path of an external database.
4. In the **User Name** field, enter the user name for the database.
5. In the **Password** field, enter the password for the database.
6. Unless the database provider is PostgreSQL, click **Upload** to upload a JDBC driver for the database.
7. Click **Next**.

Engine Persistence mode FTGroup

The database is not needed when the **Engine Persistence mode** is **FTGroup**.

- If you select **EMS** as the group provider technology then the **EMS as the Group Provider Configuration** page is displayed.

EMS as the Group Provider Configuration


Component Wizard

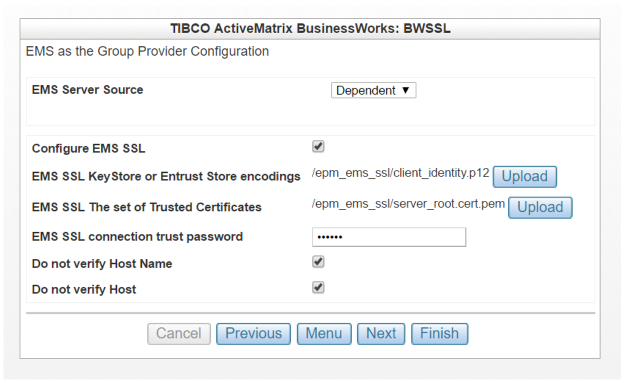
TIBCO ActiveMatrix BusinessWorks: Test

EMS as the Group Provider Configuration

EMS Server Source	Specified
EMS Server URL	[PROTOCOL]://[HOST]:[PORT]
EMS User Name	<USER_NAME>
EMS Password	••••••
Configure EMS SSL	✓
EMS SSL KeyStore or Entrust Store encodings	Upload
EMS SSL The set of Trusted Certificates	Upload
EMS SSL connection trust password	••••••
Do not verify Host Name	✓
Do not verify Host	✓

1. If you select the **EMS Server Source** as **Specified** from the drop-down, then you need to specify the **EMS Server URL** , **EMS User Name**, **EMS Password**.
2. If you select the **Configure EMS SSL** check box, then you upload **EMS SSL KeyStore or Entrust Store encodings**, **EMS SSL The set of Trusted Certificates**, and provide values for **EMS SSL connection trust password**, **Do not verify Host Name**, **Do not verify Host** .

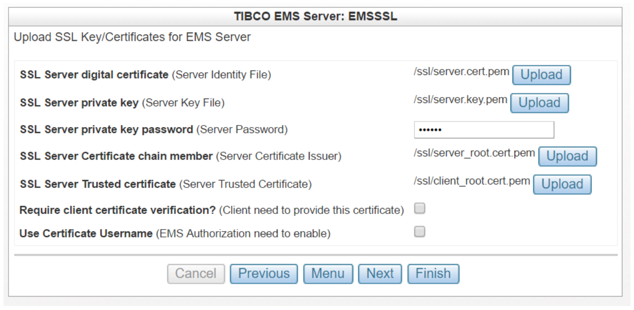
   TIBCO ActiveMatrix BusinessWorks Certificates


TIBCO ActiveMatrix BusinessWorks: BWSSL

EMS as the Group Provider Configuration

EMS Server Source	Dependent
Configure EMS SSL	✓
EMS SSL KeyStore or Entrust Store encodings	/epm_ems_ssl/client_identity.p12 Upload
EMS SSL The set of Trusted Certificates	/epm_ems_ssl/server_root.cert.pem Upload
EMS SSL connection trust password	••••••
Do not verify Host Name	✓
Do not verify Host	✓

TIBCO EMS Certificates



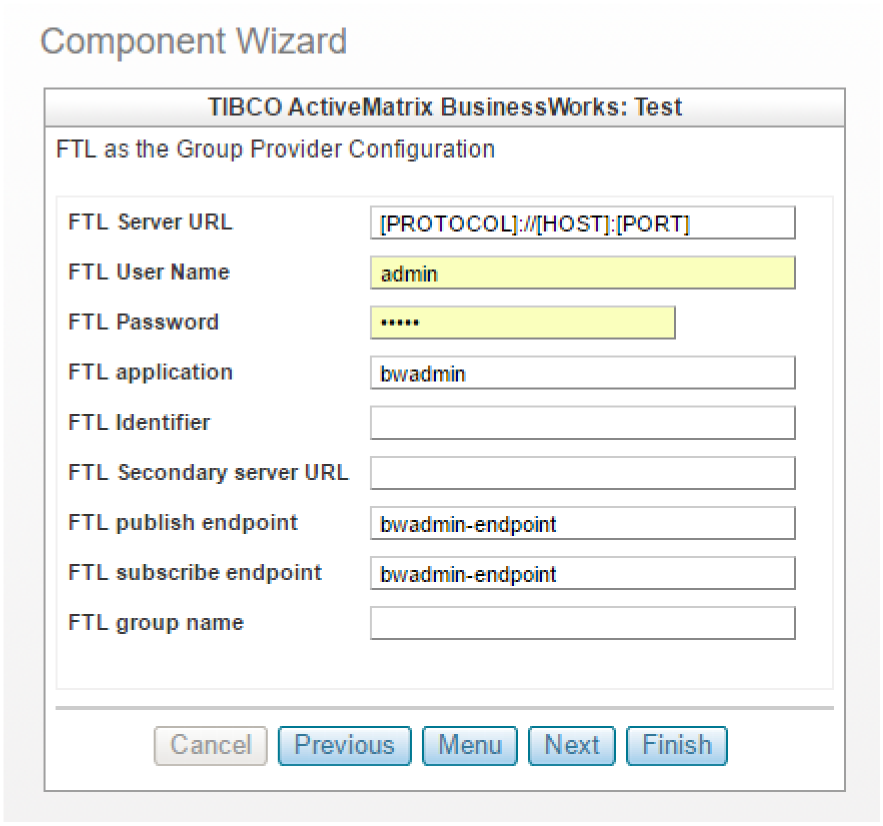
TIBCO EMS Server: EMSSSL

Upload SSL Key/Certificates for EMS Server

SSL Server digital certificate (Server Identity File)	/ssl/server.cert.pem Upload
SSL Server private key (Server Key File)	/ssl/server.key.pem Upload
SSL Server private key password (Server Password)	••••••
SSL Server Certificate chain member (Server Certificate Issuer)	/ssl/server_root.cert.pem Upload
SSL Server Trusted certificate (Server Trusted Certificate)	/ssl/client_root.cert.pem Upload
Require client certificate verification? (Client need to provide this certificate)	☐
Use Certificate Username (EMS Authorization need to enable)	☐

- If you select **FTL** as the group provider technology for the engine, then the **FTL as the Group Provider Configuration** page is displayed.

FTL as the Group Provider Configuration



Component Wizard

TIBCO ActiveMatrix BusinessWorks: Test

FTL as the Group Provider Configuration

FTL Server URL	[PROTOCOL]://[HOST]:[PORT]
FTL User Name	admin
FTL Password	•••••
FTL application	bwadmin
FTL Identifier	
FTL Secondary server URL	
FTL publish endpoint	bwadmin-endpoint
FTL subscribe endpoint	bwadmin-endpoint
FTL group name	

Procedure

1. Specify details such as **FTL Server URL**, **FTL User Name**, **FTL Password**, and so on to use the new FTL server.
2. Click **Next**.

Specifying the Domain, AppSpace, AppNode, and APPNode HttpPort Base Configuration

This component wizard page defines the Domain, AppSpace, and AppNode names for your ActiveMatrix BusinessWorks environment. This page also sets the AppNode HTTP Port Base for your ActiveMatrix Business run time.

When specifying names for the Domain, AppSpace, and AppNode use letters, numbers, hyphens (-) and underscores (_). Other special characters and spaces are not allowed.

Procedure

1. Specify **Domain Name**.

   A BusinessWorks domain is a logical group that provides an isolated environment for applications and their resources to reside. It provides an administrative boundary for an integration project. Each domain can share machines with other domains, but communication between domains is not allowed. Domains include servers that may or may not be distributed over different machines and operating systems.

   If you keep the **Domain Name** field empty or choose to run the component in Local mode, the default domain name is used: *<ComponentName>_Domain.*

   When the Agent Role is set to Client, the client runtime instance uses the domain of ActiveMatrix BusinessWorks running as Server.

   TIBCO ActiveMatrix BusinessWorks running as a Client uses the domain from the TIBCO ActiveMatrix BusinessWorks instance running as a *Server*.

2. Specify **AppSpace Name**.

   A BusinessWorks AppSpace is a virtual pool of AppNodes where an application is deployed. When an application is deployed, the AppSpace starts the application on each of its AppNodes. More AppNodes can be added dynamically to the AppSpace to manage the load-balancing and fault tolerance needs of an application. Multiple applications can be deployed to an AppSpace.

   If you leave the **AppSpace Name** field empty or choose to run the component in Local mode, the default AppSpace name is: *<ComponentName>_AppSpace.*

3. Specify **AppNode Name**.

   A BusinessWorks AppNode is a runtime entity for hosting application modules and libraries. An AppNode represents a physical engine process that is launched when an application starts to run. On the component wizard, one AppNode can be created in an AppSpace, but using the TEA Enabler or other BusinessWorks Administrator interface, you can create more.

   If you keep the **AppNode Name** field empty or run the component in the Local mode, the default AppNode name is: *<ComponentName>_<ComponentInstance>_AppNode*

   - When `ComponentInstance = 0`, the name is *<ComponentName>_AppNode*
   - When the `ComponentInstance > 0`, the name is *<ComponentName>_<ComponentInstance>_AppNode*
   - When a component has multiple instances, appNode names are incremented: *<ComponentName>_<ComponentInstance>_AppNode*

4. In **AppNode HTTP Port Base**, specify a port or select the default.

   The HTTP Port Base value is used as a basis to calculate and assign ports dedicated for specific BusinessWorks applications. This helps to prevent HTTP port conflicts when multiple BusinessWorks applications are deployed to the same machine. Silver Fabric notifies the Virtual Router of the HTTP end points so that the end user can call those endpoints without having to know where the application runs (machine port). The default value is 18200.

5. Specify the number of fault tolerance instances in the **Number of FT Instances** field. The default value is 1.
6. Specify the value for **AppNode HttpPort Increase** field. The default value is 50.
7. Click **Next**.

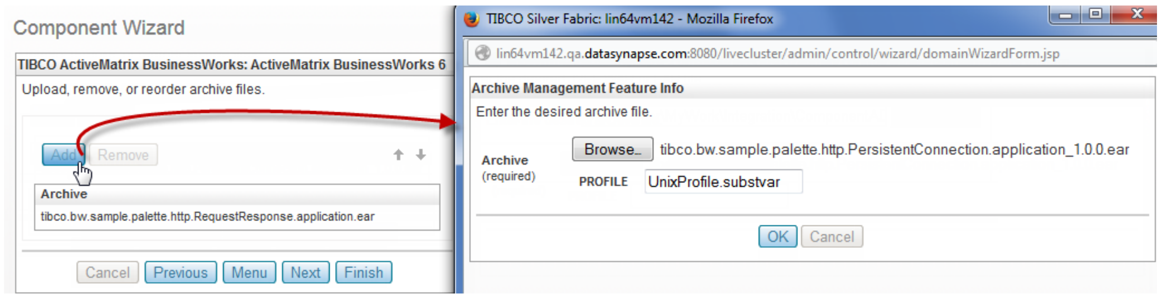
Uploading a BusinessWorks Project

If you want TIBCO Silver Fabric Enabler for BusinessWorks components to publish and run one or more BusinessWorks projects, upload one or more archive files (EAR or ZIP files) as follows:

Procedure

1. Click the **Add** button in the **Upload, remove, or recorder archive files** panel as shown in Uploading Archives figure:

   Uploading Archives



Component Wizard
TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6
Upload, remove, or reorder archive files.
Add
Remove
Archive
Cancel
Previous
Menu
Next
Finish
TIBCO Silver Fabric: lin64vm142 - Mozilla Firefox
Archive Management Feature Info
Enter the desired archive file.
Browse...
Archive (required)
PROFILE
OK
Cancel

tibco.bw.sample.palette.http.RequestResponse.application.ear

lin64vm142.qa.datasynapse.com:8080/livecluster/admin/control/wizard/domainWizardForm.jsp

Archive (required): tibco.bw.sample.palette.http.PersistentConnection.application_1.0.0.ear

PROFILE: UnixProfile.substvar

2. Click the **Browse** button in the **Upload A File** panel to navigate to your EAR or .zip file.

   You can upload one of following archive files:

   - **`.ear` File**: When you upload a BusinessWorks EAR file, it uses the default value of the global variables set in TIBCO Business Studio. The EAR file should have a JAR file and a `META-INF` folder, where the profile is defined so that it can be used to deploy the application.

     – Enter a Profile to be used with the EAR upload.

       When using an EAR file, enter the full file name of an application archive substitution variables file as the value of **PROFILE**. Your EAR file should contain at least one profile definition: `<`*`Profile_Name`*`.substvar` with the application archive substitution variables like: `UnixProfile.substvar`

   - **`.zip` File**: You can create a `.zip` file that contains an EAR file and a profile file named `<`*`Profile_Name`*`>.substavar`. The enabler uses this profile to deploy the application.

     The `.zip` file can contain other files used in the profile along with the `.ear` file, which is used in place of the profile field.

     Do not enter a profile when using a `.zip` file, because the profile (and the contained substitution variables) are automatically picked up from the zipped archive.

   Alternatively, you can deploy applications using an associated TIBCO Enterprise Administrator, the `bwadmin` utility, or by making REST calls.

   Refer to *TIBCO Silver® Fabric Enabler for TIBCO Enterprise Administrator User's Guide* for more information on the TIBCO Enterprise Administrator.

   Refer to Continuous Deployment - Deploy Archives Directly to Endpoints.

3. Click **OK** and perform one of the following steps according to your requirement:

- Repeat the steps to add more application archives to deploy and run.
- Click **Next** to proceed with further configuration of the component.
- Click **Finish** to complete and save configurations to the component.

HTTP Port Settings (optional)

Manage the HTTP ports for the following ActiveMatrix BusinessWorks processes:

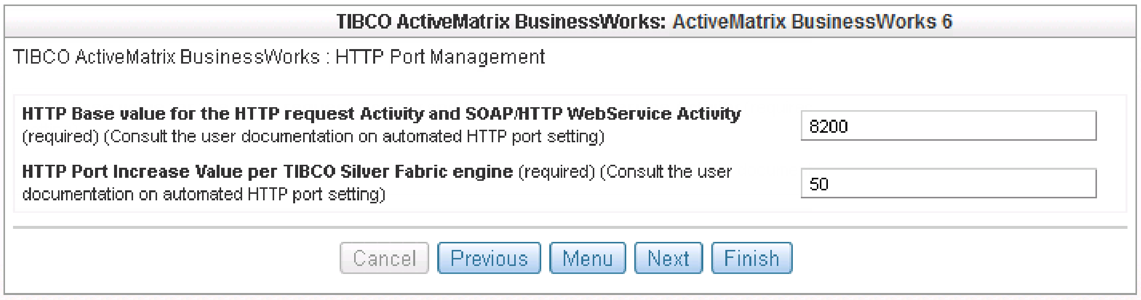
- SOAP/HTTP Event Source
- SOAP/HTTP Service
- HTTP Receiver Activity

To avoid port conflicts on the TIBCO Silver Fabric engines where TIBCO Active Matrix BusinessWorks™ is running, TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks provides a field named: **HTTP Base value for the HTTP request Activity and SOAP/HTTP Web Service Activity** to set the HTTP base value.

This defines a reserved range of ports that should not be used by unrelated processes.

Edit the base values of the "*HTTP Base Value...*" and "*HTTP Port Increase Value...*" variables on the HTTP Port Management page..

The configuration parameters of the HTTP Port Management page are as described.


TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

TIBCO ActiveMatrix BusinessWorks : HTTP Port Management

HTTP Base value for the HTTP request Activity and SOAP/HTTP WebService Activity (required) (Consult the user documentation on automated HTTP port setting): 8200

HTTP Port Increase Value per TIBCO Silver Fabric engine (required) (Consult the user documentation on automated HTTP port setting): 50

The HTTP base port value is used to calculate derived port values for Web Services using SOAP over HTTP transport and other HTTP activities as required by your BusinessWorks project.

HTTP Port = "*HTTP Base Value...*" + "*HTTP Port Increase Value...*"* (*HttpActivityNumber*(1,2,3,or more)-1) + *EngineInstanceValue*

For example, you have a BusinessWorks project configured using two HTTP activities. The HTTP activities are running on the Engine instance whose number is `2`. The "*HTTP Base Value...*" is the default value: `8200`. The "*HTTP Port IncreaseValue...*" is the default value: `50`.

The first port is set to `8202`. That value is derived from: `8200 + 50 *(1-1) + 2`

The second port is set to `8252`, derived from: `8200 + 50 *(2-1) + 2.`

You should **not** put different base values on different BusinessWorks components running on the same Silver® Fabric private cloud. If ports are assigned and incremented for different components in different stacks on the same set of engines you can encounter port conflicts.

If your project uses more than 50 HTTP ports, the "*HTTP Port Increase Value...*" variable must be changed to be greater than the number of HTTP ports used.

LDAP Authentication (optional)

An LDAP Authentication shared resource represents configuring the connection to an LDAP server. This connection is used by component implementations to look up names in an LDAP directory server.

LDAP authentication is primarily used for HTTP basic authentication in ActiveMatrix BusinessWorks™ 6.x.

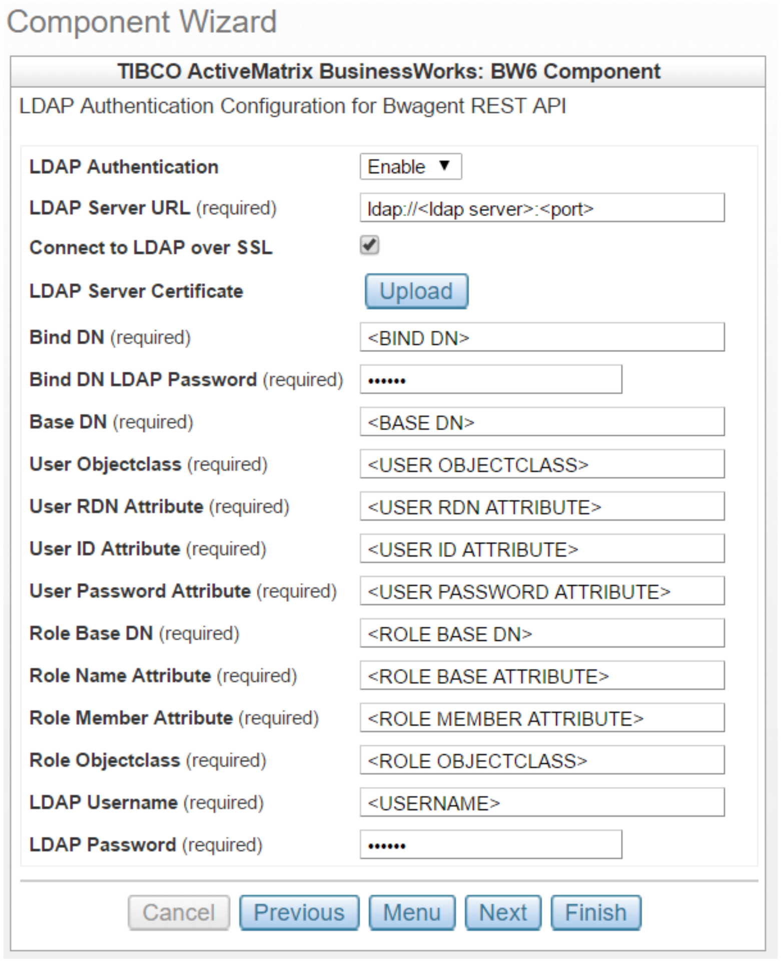
LDAP support for user authentication for bwagent REST API enables authentication of the user when the bwagent runs. This is required because when a bwagent runs, it exposes some webservices and REST API.

Procedure

1. From the **LDAP Authentication** drop-down, select **Enable**.
2. Enter details for **LDAP Server URL**, **Bind DN**, **Bind DN LDAP Password**, **Base DN**, **User Objectclass**, **User RDN Attribute**, **User ID Attribute**, **User Password Attribute**, **Role Base DN**, **Role Name Attribute**, **Role Member Attribute**, **Role Objectclass**, **LDAP Username** and **LDAP Password** . If you select the **Connect to LDAP over SSL** checkbox, then upload the LDAP Sever Certicate by clicking the **Upload** button.
3. Click **Next**

Result

LDAP Authentication Configuration for BWagent REST API


Component Wizard
TIBCO ActiveMatrix BusinessWorks: BW6 Component
LDAP Authentication Configuration for Bwagent REST API
Upload
Cancel
Previous
Menu
Next
Finish

LDAP Authentication	Enable ▼
LDAP Server URL (required)	ldap://<ldap server>:<port>
Connect to LDAP over SSL	✓
LDAP Server Certificate	Upload
Bind DN (required)	<BIND DN>
Bind DN LDAP Password (required)	••••••
Base DN (required)	<BASE DN>
User Objectclass (required)	<USER OBJECTCLASS>
User RDN Attribute (required)	<USER RDN ATTRIBUTE>
User ID Attribute (required)	<USER ID ATTRIBUTE>
User Password Attribute (required)	<USER PASSWORD ATTRIBUTE>
Role Base DN (required)	<ROLE BASE DN>
Role Name Attribute (required)	<ROLE BASE ATTRIBUTE>
Role Member Attribute (required)	<ROLE MEMBER ATTRIBUTE>
Role Objectclass (required)	<ROLE OBJECTCLASS>
LDAP Username (required)	<USERNAME>
LDAP Password (required)	••••••

AppNode or bwagent Running Condition Configuration

When the environment is set to run locally (in local mode), a running condition check can be applied to ensure that the AppNode process is responsive.

When in Enterprise mode, running condition applies to the bwagent process.

The following are the configuration parameters in the AppSapce and AppNode Running Condition page:

Polling Period (in seconds) for detection of TIBCO ActiveMatrix BusinessWorks AppNode or bwagent running verification (required)

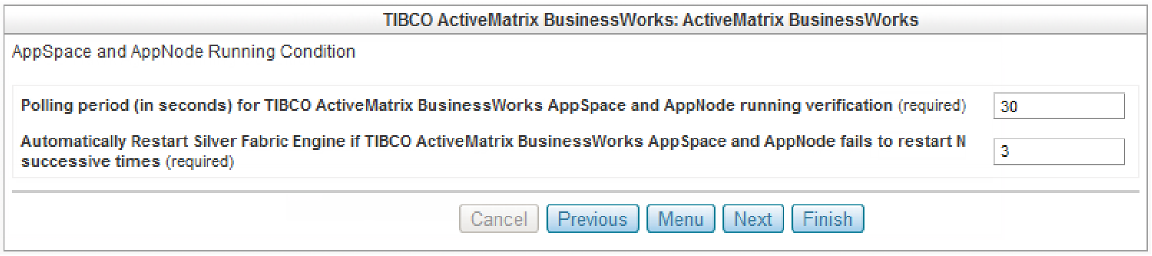
Enter an integer to specify the number of seconds between periodic verification checks that the AppNode or `bwagent` are still running.
If the AppNode or `bwagent` become unresponsive to this verification then the process is automatically restarted.

Automatically Restart Silver Fabric Engine if TIBCO ActiveMatrix BusinessWorks AppSpace and AppNode fails to restart N successive times (required)

Enter an integer to specify the number of restart retries for the TIBCO ActiveMatrix BusinessWorks application before the TIBCO Silver Fabric Engine will be restarted. A successful restart will reset the count.

Setting the value to "0" disables the feature.

AppNode Running Condition



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks

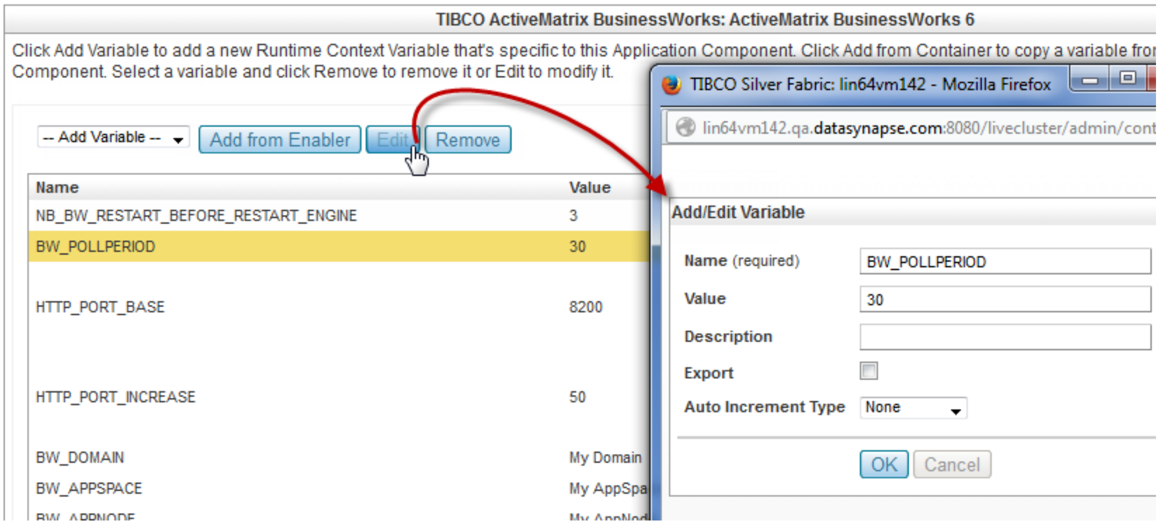
AppSpace and AppNode Running Condition

Polling period (in seconds) for TIBCO ActiveMatrix BusinessWorks AppSpace and AppNode running verification (required)	30
Automatically Restart Silver Fabric Engine if TIBCO ActiveMatrix BusinessWorks AppSpace and AppNode fails to restart N successive times (required)	3

Review and Edit Runtime Context Variables (optional)

Review variables and their attributes or change those values to refine your runtime implementation. Select any row and click **Edit** to modify that variable.

Select and Edit a Variable



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

Click Add Variable to add a new Runtime Context Variable that's specific to this Application Component. Click Add from Container to copy a variable fro
Component. Select a variable and click Remove to remove it or Edit to modify it.

Name	Value
NB_BW_RESTART_BEFORE_RESTART_ENGINE	3
BW_POLLPERIOD	30
HTTP_PORT_BASE	8200
HTTP_PORT_INCREASE	50
BW_DOMAIN	My Domain
BW_APPSPACE	My AppSpa
BW_APPNODE	My AppNod

TIBCO Silver Fabric: lin64vm142 - Mozilla Firefox

lin64vm142.qa.datasynapse.com:8080/livecluster/admin/cont

Add/Edit Variable

Name (required)	BW_POLLPERIOD
Value	30
Description	
Export	☐
Auto Increment Type	None

Most of the runtime context variables have corresponding component wizard UI for setting values. The table clarifies what variables are described by UI settings and it provides a description of function for those that are not described elsewhere.

Runtime Context Variables and Component Wizard UI

Runtime Context Variable	Description
`NB_BW_RESTART_BEFORE_RESTART_ENGINE`	
`BW_POLLPERIOD`	
`HTTP_PORT_BASE`	The port number to listen for incoming HTTP requests
`HTTP_PORT_INCREASE`	Specifies to increment the HTTP port number
`HAWK_PLUGIN_CONFIG_FILE`	The location where TIBCO Hawk Plug-in configuration files exist
`ACTION_TO_PERFORM_FOR_HAWK_FILE`	The actions to be performed on Hawk files
`BW_DOMAIN`	A domain is a logical group that provides an isolated environment for applications and their resources. Runtime entities such as AppSpaces and AppNodes are contained within a domain
`BW_APPSPACE`	An AppSpace is a virtual pool of AppNodes where an application is deployed
`BW_APPNODE`	An AppNode is created under an AppSpace. An AppNode is a runtime entity for hosting application modules and libraries
`BW_ADMIN_MODE`	In the Admin mode, you can manage and monitor runtime entities
`APPNODE_HTTPPORT_BASE`	The port number on which AppNode listens for incoming HTTP requests
`BW_CREATE_ENTITIES`	
`BW_TRANSPORT`	A transport used for communication in BusinessWorks
`BW_ENTERPRISE_AGENT_ROLE`	The role of an agent in the BusinessWorks Enterprise version
`WITH_TEASERVER`	
`USE_DEPEND_TEASERVER`	
`TEA_SERVER_URL`	The URL of the TEA server

Runtime Context Variable	Description
`MIN_SEEDER_COUNT`	The minimum seeder count value for a specified space in TIBCO ActiveSpaces persistence and transport layer
`QUORUM_SIZE`	The size of the quorum in TIBCO ActiveSpaces persistence and transport layer. The minimum number of seeders for recovery
`DATASTORE_LOCATION`	The location of a data store in TIBCO ActiveSpaces persistence and transport layer
`BW6_EMS_SERVER_SOURCE`	The source file location of EMS Server
`BW6_EMS_SERVER_URL`	The URL to connect to the EMS server `Example: tcp://localhost:7222`
`BW6_EMS_SERVER_USERNAME`	The user name to authenticate to the EMS server. The default is admin
`BW6_DB_PROVIDER`	The database provider: postgresql, mysql, oracle, mssql or db2
`BW6_JDBC_DRIVER`	The JDBC driver
`BW6_DB_URL`	The URL to connect to the database `Example: jdbc:postgresql://localhost:5432/ bwadmindb`
`BW6_DB_USERNAME`	The user name to authenticate the database for ActiveMatrix BusinessWorks version 6.x
`BW6_JDBC_DRIVER_FILE`	The location of JDBC driver files for ActiveMatrix BusinessWorks version 6.x
`BW6_EMS_SERVER_PASSWORD`	The password to authenticate to the EMS server. The default is admin
`BW6_DB_PASSWORD`	The password to authenticate the database for ActiveMatrix BusinessWorks version 6.x.
`BW6_JDBC_DRIVER_FILE`	The location of JDBC driver files
`BW_ENTERPRISE_AGENT_HTTP_PORT_BASE`	The port number configured in the HTTP Connector shared resource for ActiveMatrix BusinessWorks Agent for Enterprise version
`BW_ENTERPRISE_TEA_AGENT_HTTP_PORT_BASE`	The port number configured in the HTTP Connector shared resource for TEA Agent for ActiveMatrix BusinessWorks Enterprise version

Runtime Context Variable	Description
`BW_ENTERPRISE_DISCOVERY_PORT_BASE`	The port and interface to be used to discover all the agents in the BusinessWorks Enterprise version. Must be same for all the agents in the network. This is typically set to a comma separated list of listen URLs that form a small subset of agents in the network
`BW_ENTERPRISE_REMOTE_LISTEN_PORT_BASE`	The port and interface of the bwagent for the BusinessWorks Enterprise version that remote clients (bwagents configured as remoteclient) can connect to
`DISCOVER_SERVERS_POOL_PERIOD`	
`DELETE_BW_APPLICATION_CONF_AT_SHUTDOWN`	Removes all the configuration files at shutdown
`DO_NOT_REDEPLOY_EAR_FILE_AT_STARTUP`	Specifies that EAR file should not be deployed gain at startup
`BW_ENTERPRISE_AGENT_HTTP_PORT_BASE`	The port number on which BW Enterprise Agent listens for incoming HTTP requests
`BW_ENTERPRISE_TEA_AGENT_HTTP_PORT_BASE`	The port number on which TEA Agent listens for incoming HTTP requests
`MANAGED_PROCESS_HAWK_AGENT_ENABLED`	
`MANAGED_PROCESS_SERVICE_ENABLED`	
`TestPro_PROFILE`	
`BW6_FTL_REALMSERVER_URL`	Set The FTL realm server. Example: `BW6_FTL_REALMSERVER_URL = http://localhost:8070`
`BW6_FTL_REALMSERVER_USERNAME`	Set the FTL user name.
`BW6_FTL_REALMSERVER_PASSWORD`	Set the FTL password.
`BW6_FTL_APPLICATION`	Set the application name. Example: `BW6_FTL_APPLICATION=bwadmin`
`BW6_FTL_ENDPOINT`	Set the FTL endpoint. Example: `BW6_FTL_ENDPOINT=bwadmin-endpoint`
`BW6_FTL_SECONDARY`	Set the secondary realm server
`BW6_FTL_IDENTIFIER`	Set the FTL identifier
`BW6_FTL_DATAFORMAT`	Set the FTL data format. Example: `BW6_FTL_DATAFORMAT=bw-format`

Runtime Context Variable	Description
`BW6_FTL_INBOX`	Set the FTL inbox. Example: `BW6_FTL_INBOX =bw-inbox`
`EMS_RECONNECTION_INTERVAL`	Set the reconnection interval, in milliseconds. It defines how often the bwagent tries to reconnect with the EMS server if the connection is lost. The default time interval is 10 seconds, or 10000 milliseconds.
`APPNODE_SHUTTING_DOWN_TIMEOUT`	Set the timeout in minutes after which the AppNode and all the applications running on that appnode are shutdown forcefully, irrespective of the state of the AppNode or the applications. Default value is '0'

Adding a String, Environment, System, or Encrypted Variable

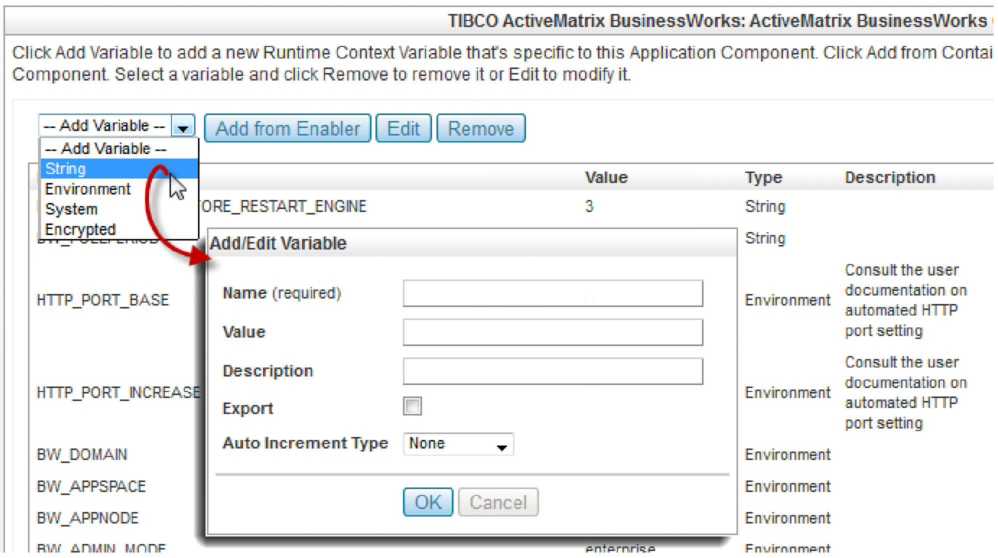
The string, environment, system, or encrypted variables may be added to the component to define and set runtime-specific context variables.

For TIBCO Silver Fabric mediated publishing, select a variable type from the **Add Variable** drop-down list or **Add from Enabler**, to use a variable from a selected Enabler.

Procedure

1. Click the **Add Variable** selector.
2. Select the variable type from the list:
    - String
    - Environment
    - System
    - Encrypted

    Add or Edit a String Variable



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks

Click Add Variable to add a new Runtime Context Variable that's specific to this Application Component. Click Add from Contai
Component. Select a variable and click Remove to remove it or Edit to modify it.

	Value	Type	Description
ORE_RESTART_ENGINE	3	String	
		String	
HTTP_PORT_BASE		Environment	Consult the user documentation on automated HTTP port setting
HTTP_PORT_INCREASE		Environment	Consult the user documentation on automated HTTP port setting
BW_DOMAIN		Environment	
BW_APPSPACE		Environment	
BW_APPNODE		Environment	
BW_ADMIN_MODE	enterprise	Environment	

Add/Edit Variable

Name (required)	
Value	
Description	
Export	☐
Auto Increment Type	None

3. Click **OK**.
4. (Optional) Variable values from an Enabler may be added to run time as well. Click the **Add from Enabler** button to add Enabler-specific context variables.

   After you have added any runtime context variable, select the variable (selected row is highlighted) and click **Edit** to change its attributes. Remove selected rows.

String Variable to Change an Existing Context Variable

You can add a string variable to change preexisting context variables. Some of these variables are not exposed in the interface elsewhere, but you can change their values.

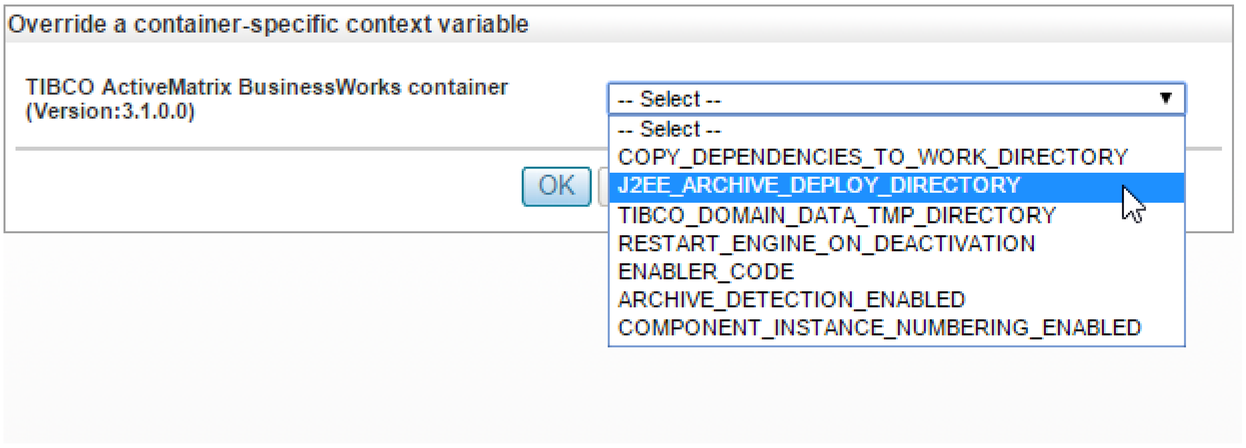
For example, the context variable, ARCHIVE_DETECTION_FREQUENCY, is the periodic interval for detection of deployed archives. The default value for this interval is 30 seconds. The broker uses this data to synchronize the archive with scaling rules.

For this example, if your archive takes longer than 30 seconds to start and respond, then you would set the value higher than the time needed to deploy and start that archive.

Changes to variables are optional and implementation dependent, because all variables have default values that are usually appropriate for the most common use cases.

Variable values from the Enabler can be changed as well. Use the **Add from Enabler** button to change values of Enabler-specific context variables.

Add from Enabler


Override a container-specific context variable

TIBCO ActiveMatrix BusinessWorks container (Version:3.1.0.0)

-- Select --

-- Select --
COPY_DEPENDENCIES_TO_WORK_DIRECTORY
J2EE_ARCHIVE_DEPLOY_DIRECTORY
TIBCO_DOMAIN_DATA_TMP_DIRECTORY
RESTART_ENGINE_ON_DEACTIVATION
ENABLER_CODE
ARCHIVE_DETECTION_ENABLED
COMPONENT_INSTANCE_NUMBERING_ENABLED

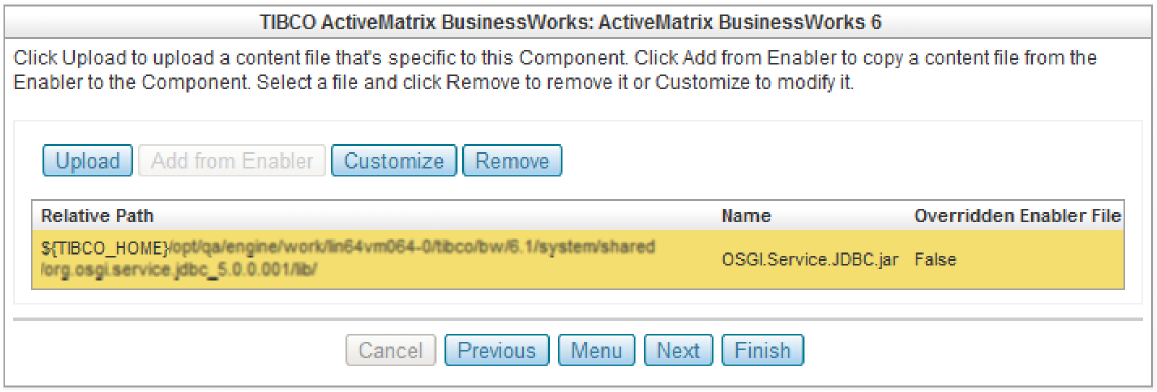
Uploading a Content File (optional)

Procedure

1. Upload a content file (for example, the JDBC Jar file for mysql) specific to this component from this dialog. You can perform the following operations:

   - **Upload**: Upload a new file.
   - **Add from Enabler**: Copy a content file from the Enabler to the component.
   - **Customize**: Modify the content file.
   - **Remove**: Delete the file.
2. Add files to a relative path associated with the BusinessWorks component that can be required for the component to run according to design.

Uploading Content Files


TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

Click Upload to upload a content file that's specific to this Component. Click Add from Enabler to copy a content file from the Enabler to the Component. Select a file and click Remove to remove it or Customize to modify it.

Relative Path	Name	Overridden Enabler File
${TIBCO_HOME}/opt/qa/engine/work/lin64vm064-0/tibco/bw/6.1/system/shared/org.osgi.service.jdbc_5.0.0.001/lib/	OSGI.Service.JDBC.jar	False

Add or Remove BusinessWorks Component Statistics

You can add or remove the following statistics on component created in version 6.x for tracking:

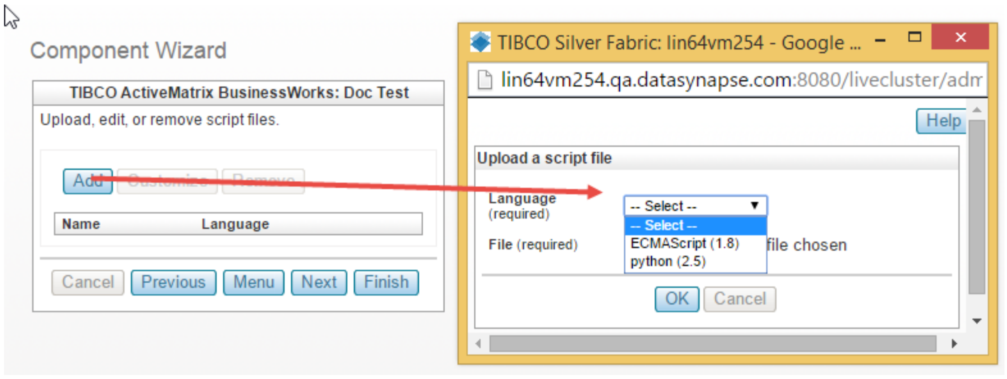
- **BW Running Process Count in an application**: provides count of the number of running process instances in an application.
- **BW Running Process Count**: provides count of the total number of running process instances.

Uploading, Editing, or Removing Script Files

Procedure

- Add, edit, or remove ECMA or python scripts from the Upload, Edit or Remove Script files page.

  Script Files



Publishing a Component

Procedure

1. Select **Publish Component** in the **Actions** list located in the same row as the component you wish to publish.
2. Click the **Actions** icon located at the left of the line of the component you just created or changed.
3. Select **Publish Component** from the menu of actions presented and confirm your action.

Changing the Component Enabler

You can upgrade a component created with TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks release 2.6, 3.0, 3.1, 3.2.

Back up your engines and consider timing for when you wish to perform a component restart so brief operational downtime has minimal impact.

Procedure

1. Using the TIBCO Silver Fabric Administrator Components page identify the BusinessWorks component you wish to upgrade from release 2.6.x, 3.0.x, 3.1.x, or 3.2x The **Enabler Version** column helps to identify out-of-date enablers.

   Click the component **Actions** menu icon on the row for the component you will update and choose **Change Enabler**. Select the enabler version upgrade target and click **OK**.

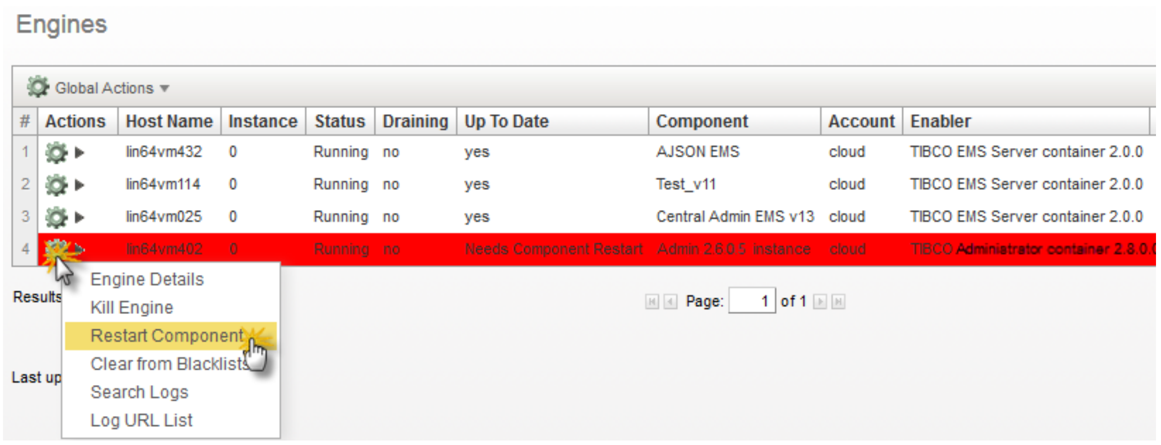
   You cannot downgrade a component.

2. Click the component **Actions** menu icon again, click **Publish Changes**, and then click **OK** to publish the selected component.

3. Switch to the **Engines** page and those stacks that used the upgraded component will appear in red if they require a component restart.

   Click the **Engine Actions** menu icon that "Needs a Component Restart", then click **Restart Component**.

   Restarting the Component



Engines

#	Actions	Host Name	Instance	Status	Draining	Up To Date	Component	Account	Enabler
1		lin64vm432	0	Running	no	yes	AJSON EMS	cloud	TIBCO EMS Server container 2.0.0
2		lin64vm114	0	Running	no	yes	Test_v11	cloud	TIBCO EMS Server container 2.0.0
3		lin64vm025	0	Running	no	yes	Central Admin EMS v13	cloud	TIBCO EMS Server container 2.0.0
4		lin64vm402	0	Running	no	Needs Component Restart	Admin 2.6.0.5 instance	cloud	TIBCO Administrator container 2.8.0.

The expected result of the component restart are determined by the values of the parameters `DELETE_BW_APPLICATION_CONF_AT_SHUTDOWN` and `REDEPLOY_EAR_FILE_AT_STARTUP` set during the basic configuration.

Stacks in Silver Fabric

For 5.x Components

Components are deployed within a stack. The TIBCO ActiveMatrix BusinessWorks™ 5.x component(s) depends upon an instance of the TIBCO Administrator component, and both must be present in the stack. BW component inherits of all specific variable to setup the domain from TIBCO Administrator.

Any number of components could be included in your stack depending on your implementation, but at the minimum a BusinessWorks stack must have a TIBCO Administrator component and an instance of the TIBCO ActiveMatrix BusinessWorks™ component. However, if you want to start and stop component individually, you can put one component per stack. Once you have created components, you can create a stack so they are published to a Silver Fabric Engine as a unit.

Each *TIBCO_DOMAIN* can have one stack, but each stack can have multiple enablers including multiple TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks. After initially defining a stack, you can still update it by adding or removing BusinessWorks components.

For 6.x Components

Components are usually published and run within a stack. A published TIBCO ActiveMatrix BusinessWorks component can be used in many different ways:

- Publish and run a BusinessWorks component by itself in an ad hoc stack.
- Publish and run a BusinessWorks component by itself in a stack.
- Publish a BusinessWorks component running as a bwagent server with one or many BusinessWorks components running as bwagent Clients using the same domain space.
- Publish a set of BusinessWorks components with other components created with TIBCO Silver Fabric Enablers for the TIBCO Enterprise Administrator, TIBCO Enterprise Messaging Server, TIBCO Adapter for Databases, and more.
- Use TIBCO ActiveMatrix BusinessWorks component running on a TIBCO Silver Fabric Engine in all the standard supported ways.
- TIBCO ActiveMatrix BusinessWorks clients can be launched or deallocated based on engine metrics, application statistics, and your archive scaling rules.
- Leverage TIBCO Enterprise Administrator to manage domains, AppSpaces, AppNodes, and applications running on your BusinessWorks components.

Some restrictions that apply:

- Stacks must be limited to one TIBCO ActiveMatrix BusinessWorks Enterprise Environment running in the server mode when used with a TIBCO Enterprise Administrator component.
- The component wizard supports automated creation of one domain per stack.
- Scaling is accomplished by adding multiple BusinessWorks components configured to run as clients, TIBCO ActiveMatrix BusinessWorks Enterprise environments running with the bwagent in the client mode. BusinessWorks clients share the BusinessWorks server domain and use their own AppSpace and AppNode.

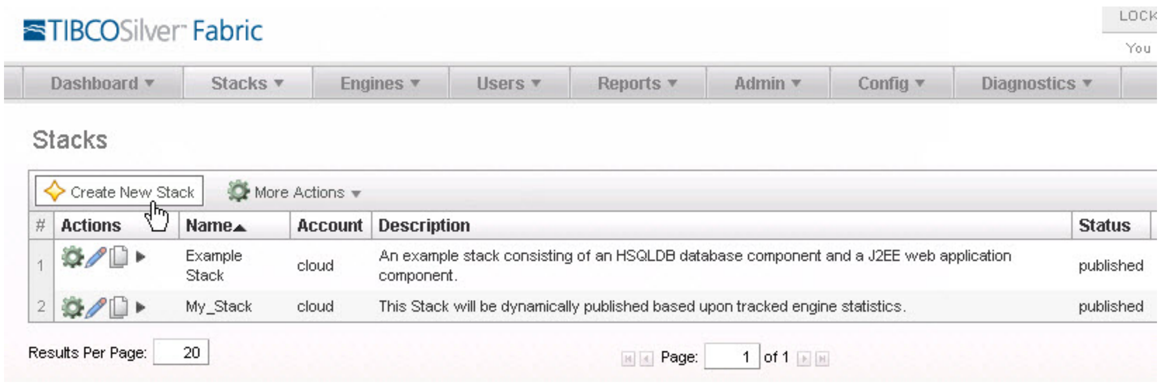
Creating a Stack

Procedure

1. In the TIBCO Silver® Fabric Administration Tool, select **Stacks** > **Stacks** .

2. Select **Create New Stack**.

   Creating Stack



Stacks

#	Actions	Name	Account	Description	Status
1		Example Stack	cloud	An example stack consisting of an HSQLDB database component and a J2EE web application component.	published
2		My_Stack	cloud	This Stack will be dynamically published based upon tracked engine statistics.	published

Results Per Page: 20

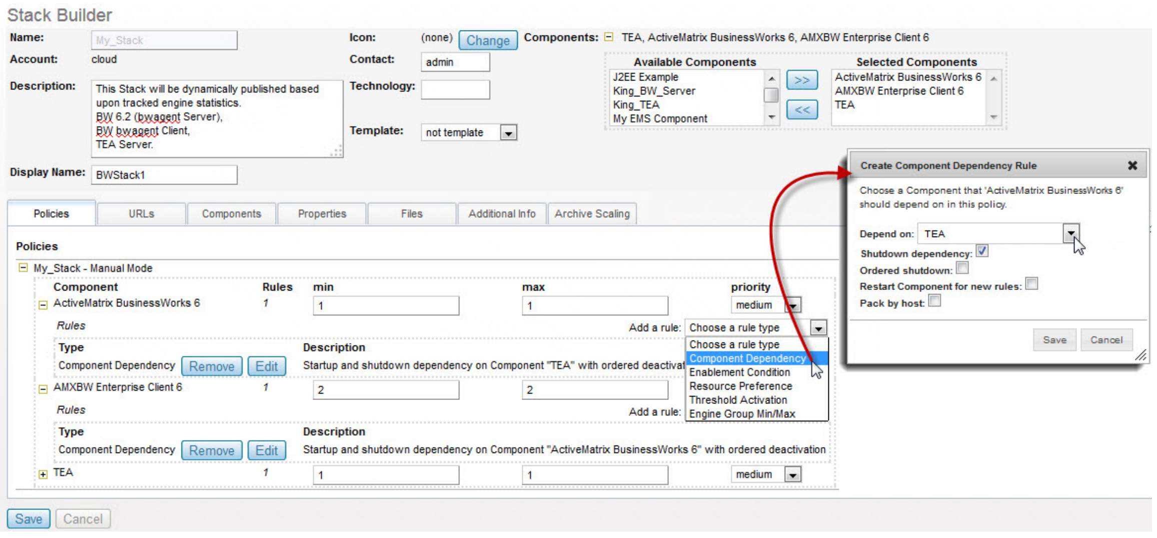
Page: 1 of 1

3. Enter a stack name in the Stack Builder page.
4. In the **Components** area, add the components you want for your stack

   For easy scalability one should add one component with TIBCO ActiveMatrix BusinessWorks configured to run as an Enterprise Environment with the Agent (bwagent) running as a server, and one component TIBCO ActiveMatrix BusinessWorks configured to run as an Enterprise Environment with the Agent (bwagent) running as a Client. The BW Client component can be scaled up to run on more engines or de-allocated based on application demand and your archive scaling rules.

5. In the Policies area, expand the component you just added to view the details of the component.

   Stack Builder page - Adding a Component Dependency



Name: My_Stack
Account: cloud
Description: This Stack will be dynamically published based upon tracked engine statistics. BW 6.2 (bwagent Server), BW bwagent Client, TEA Server.
Display Name: BWStack1
Icon: (none)
Contact: admin
Technology:
Template: not template
Components: TEA, ActiveMatrix BusinessWorks 6, AMXBW Enterprise Client 6

Available Components	Selected Components
J2EE Example	ActiveMatrix BusinessWorks 6
King_BW_Server	AMXBW Enterprise Client 6
King_TEA	TEA
My EMS Component	

Policies

My_Stack - Manual Mode

Component	Rules	min	max	priority
ActiveMatrix BusinessWorks 6	1	1	1	medium
AMXBW Enterprise Client 6	1	2	2	
TEA	1	1	1	medium

ActiveMatrix BusinessWorks 6 — Rules — Add a rule: Choose a rule type

Type	Description
Component Dependency	Startup and shutdown dependency on Component "TEA" with ordered deactivat

Choose a rule type / Component Dependency / Enablement Condition / Resource Preference / Threshold Activation / Engine Group Min/Max

AMXBW Enterprise Client 6 — Rules — Add a rule:

Type	Description
Component Dependency	Startup and shutdown dependency on Component "ActiveMatrix BusinessWorks 6" with ordered deactivation

Create Component Dependency Rule

Choose a Component that 'ActiveMatrix BusinessWorks 6' should depend on in this policy.

Depend on: TEA
Shutdown dependency: ✓
Ordered shutdown: ☐
Restart Component for new rules: ☐
Pack by host: ☐

Dependency Requirements

The dependency requirements of components created in 5.x and 6.x versions differ.

For 5.x Version

Each BusinessWorks component, must have a component dependency set on one TIBCO Administrator component. You must set this component dependency for each of your BusinessWorks components or those components without it will not have enough information for proper deployment.

These steps are required. If you do not set the dependency, TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks will not work.

The BusinessWorks™ component must have the TIBCO Administrator configuration information so that it may publish, undeploy, and communicate with other components (if required) successfully. A connection is supported between a BusinessWorks™ component and one instance of the TIBCO Administrator component - connecting with more than one TIBCO Administrator component is not supported. After setting the dependency, TIBCO BusinessWorks will start after TIBCO Administrator is up and running.

For 6.x Version

If one component requires that another component is running and active before it is started a component dependency must be set to ensure that the dependency is present prior to activation.

For example, a TIBCO ActiveMatrix BusinessWorks component connects with a TIBCO Enterprise Administrator (TEA) component so that it can be monitored and managed. The TIBCO ActiveMatrix BusinessWorks component must have a component dependency set on the TEA component because this connection is established when they are first published and run on TIBCO Silver Fabric.

TIBCO ActiveMatrix BusinessWorks (with bwagent set to run as Client), a BW Client component, must have a dependency set to connect with the only BusinessWorks server component in the stack. The BW Client components can then use the domain created by BW server component.

⚠ These steps are required. If you do not set the dependency, TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks will not work.

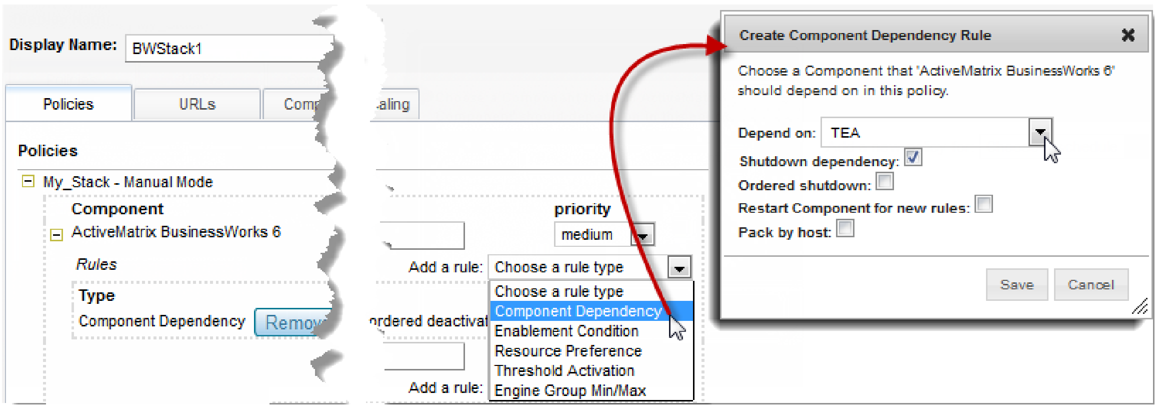
Setting Dependency

Procedure

1. Create a stack with a BusinessWorks component (with bwagent set to run as server) and a TIBCO Enterprise Administrator component.

   Alternatively you can edit an existing stack with a component that needs a dependency rule and select **Add/edit default rule settings** from the menu of the component wizard.
2. In the **Policies** area on the Stack Builder Wizard page, expand the BusinessWorks component.
3. Use the **Add Rule** pull down to select the **Component Dependency** option.

   Setting a Dependency on a TEA Component



Display Name: BWStack1

Create Component Dependency Rule

Choose a Component that 'ActiveMatrix BusinessWorks 6' should depend on in this policy.

Depend on: TEA
Shutdown dependency: ☑
Ordered shutdown: ☐
Restart Component for new rules: ☐
Pack by host: ☐

4. In the **Depend on** field, select the name of the TIBCO Enterprise Administrator component that will manage your BusinessWorks administration tasks.
5. Select any of the following check boxes to control the behavior of the component dependency. Typically Shutdown Dependency and an Ordered Shutdown are desirable.

   Shutdown Dependency

Establishes a requirement for the parent, specified in the **Depend on** field, to be active to keep the dependent component running. When selected the stack stops component dependencies if the parent component becomes unresponsive.

If you use a Database domain, BW continues to work. However, every change in TIBCO Administrator is not taken into account by BusinessWorks even after exporting the changes.

If you run TIBCO Enterprise Administrator in the Fault Tolerant mode, clear the **Shutdown Dependency** check box. Otherwise, all BusinessWorks components will stop if TIBCO Enterprise Administrator stops working.

Ordered Shutdown

Ordered Shutdown provides for a logical, sequential shutdown so that dependent components are shut down first. Ordered shutdown is especially important when the domain is hosted using a file structure instead of a dependent database. When you have an administrative component that uses an external database, order of shutdown is less important.

Restart Component for new rules

If new rules are defined for a component that has already been deployed, it must be restarted for the new changes to be applied. If you wish to manually restart components later to propagate changes leave this box cleared.

Pack by Host

Check the Pack by Host check box to specify that dependent components must run on the same host.

The *TIBCO Silver Fabric User's Guide* has more information on all of these settings.

6. Click **Save** or create another component dependency on the ActiveMatrix BusinessWorks client component that connects with the ActiveMatrix BusinessWorks server.
7. Set a component dependency on ActiveMatrix BusinessWorks client components using the same **Add a rule** drop down menu to set dependence on the single instance of a ActiveMatrix BusinessWorks server component in your stack.
8. **Save** and **Run** your stack.

BusinessWorks Components Scaling and Statistics

If you want BusinessWorks components to scale automatically, you can define rules that add or remove TIBCO Silver® Fabric Engines based on engine statistics.

Data collected are aggregated. The aggregate is used to average raw statistic values by using a source ID. The average is calculated by individually averaging the statistic values for each source ID (for example, for each engine), and then averaging the results across all engines.

If the aggregated value triggers the rule, but the normalized geometric variance across the engines is less than 0.85, then it does not add an engine. Removing engines is not affected by variance.

When an engine is added, it automatically publishes the BusinessWorks archive that was loaded in the component on a new engine. For HTTP activities, for example, web services using SOAP over HTTP transport and HTTP Receiver activities, the engine adds the URL to the load balancer automatically.

You can set up rules on Enablement Condition. You also can set up rules on threshold activation, which is a statistic on the engine itself or other engines.

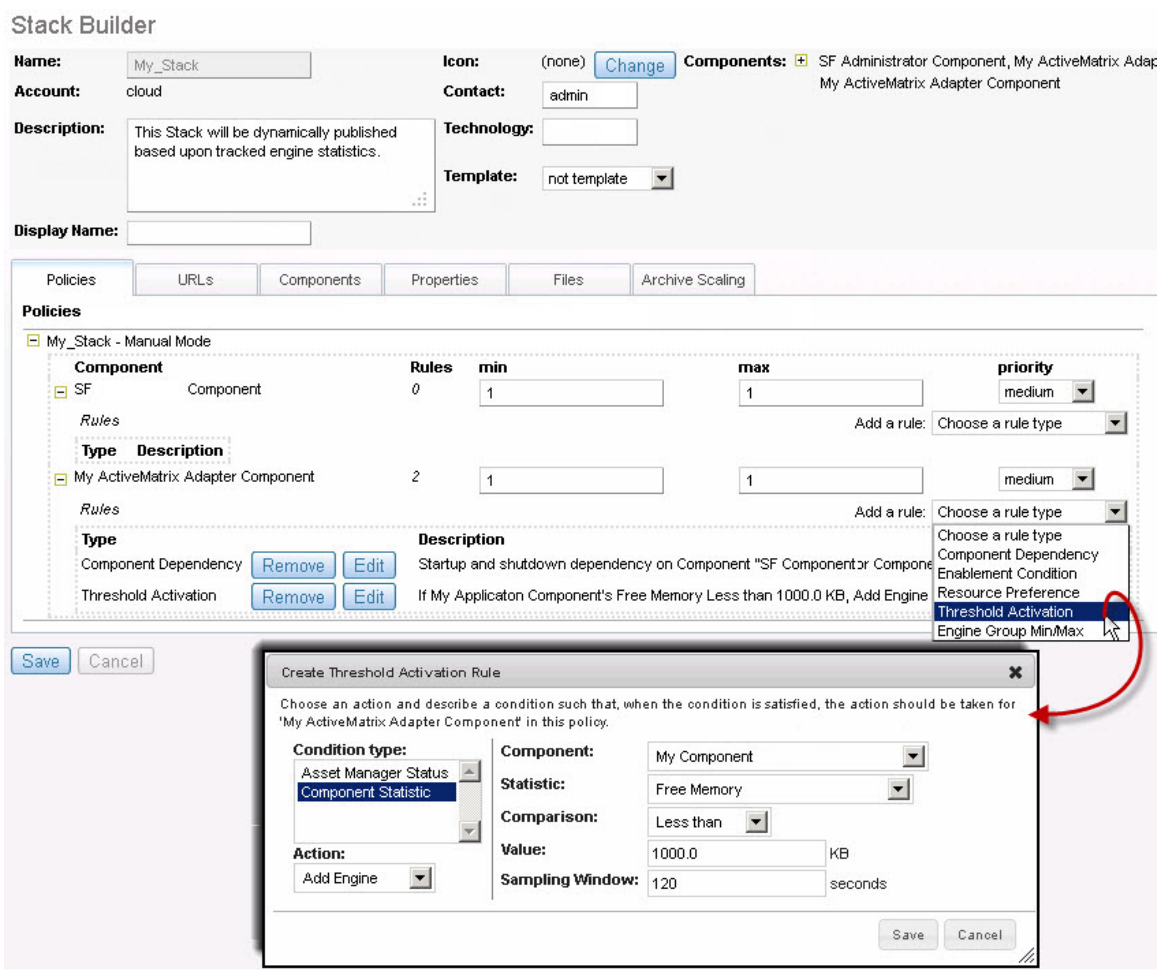
Setting Rules for an Engine

Procedure

1. On the **Policies** tab of the **Stack Builder** page, select the component on which you want to set up rules.
2. Select **Threshold Activation** or **Enablement Condition** in the **Add A Rule** pull-down list to Choose a rule type.
3. If you select **Threshold Activation**, specify the following parameters in the Create Threshold Activation Rule panel:

    - In Condition Type list, select the **Component Statistic** item.
    - In the Action list, select **Add Engine** or **Remove Engine**.
    - In the Component list, select the **Component** where the statistic rules apply.
    - In the Statistics list, select the type of **Statistics**.
    - In the Comparison list, select **Greater Than** or **Less Than**.
    - In the **Value** field, select the value of the statistic when it triggers the rules.
    - In the **Sampling Window** field, set the time interval (in seconds). It specifies how often the statistics is calculated.

Creating Rules



Stack Builder

Name: My_Stack
Account: cloud
Description: This Stack will be dynamically published based upon tracked engine statistics.
Display Name:
Icon: (none) Change
Contact: admin
Technology:
Template: not template
Components: SF Administrator Component, My ActiveMatrix Adap
My ActiveMatrix Adapter Component

Policies

My_Stack - Manual Mode

Component	Rules	min	max	priority
SF Component	0	1	1	medium
My ActiveMatrix Adapter Component	2	1	1	medium

Rules — Add a rule: Choose a rule type

Type	Description
Component Dependency	Startup and shutdown dependency on Component "SF Component" or Compone
Threshold Activation	If My Applicaton Component's Free Memory Less than 1000.0 KB, Add Engine

Create Threshold Activation Rule

Choose an action and describe a condition such that, when the condition is satisfied, the action should be taken for 'My ActiveMatrix Adapter Component' in this policy.

Condition type: Asset Manager Status / Component Statistic (selected)
Action: Add Engine
Component: My Component
Statistic: Free Memory
Comparison: Less than
Value: 1000.0 KB
Sampling Window: 120 seconds

Engine statistics consist of statistic information about the engine, machine (independent of BusinessWorks components), and statistic information gathered from the BusinessWorks Engine.

Creating Archive Scaling Rules

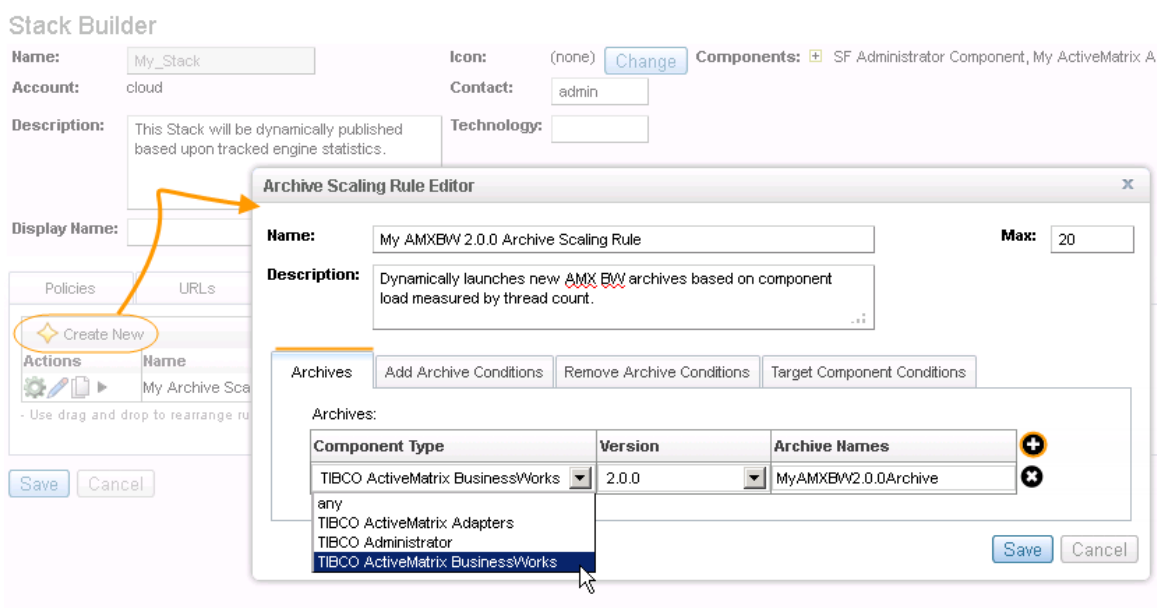
You can define stacks that add or remove component archives based on archive statistics that are monitored for triggering conditions.

Create Archive Scaling Rules to automate creation or removal of new BusinessWorks component archives when rules based on archive statistics meet or exceed thresholds or conditions you have set.

Procedure

1. After adding components to your stack you can create new scaling rules by opening the **Archive Scaling** tab and clicking on **Create New**.
2. Name your new archive scaling rule and give it a description to help you and others quickly identify the purpose and content of your archive scaling rule.
3. Use the **Archives** tab to define what archive is to be added or removed according to the rules you define in the other tabs. Click **Add** icon at the right of the column heading row to add one or more archives (process instances) to be scaled up or down in your stack.

Creating a new Archive Scaling Rule



Stack Builder

Name: My_Stack
Account: cloud
Description: This Stack will be dynamically published based upon tracked engine statistics.
Display Name:
Icon: (none) Change
Contact: admin
Technology:
Components: SF Administrator Component, My ActiveMatrix A

Policies | URLs

Create New

Actions	Name
	My Archive Sca

- Use drag and drop to rearrange ru

Save Cancel

Archive Scaling Rule Editor

Name: My AMXBW 2.0.0 Archive Scaling Rule
Max: 20
Description: Dynamically launches new AMX BW archives based on component load measured by thread count.

Archives | Add Archive Conditions | Remove Archive Conditions | Target Component Conditions

Archives:

Component Type	Version	Archive Names
TIBCO ActiveMatrix BusinessWorks	2.0.0	MyAMXBW2.0.0Archive

any
TIBCO ActiveMatrix Adapters
TIBCO Administrator
TIBCO ActiveMatrix BusinessWorks

Save Cancel

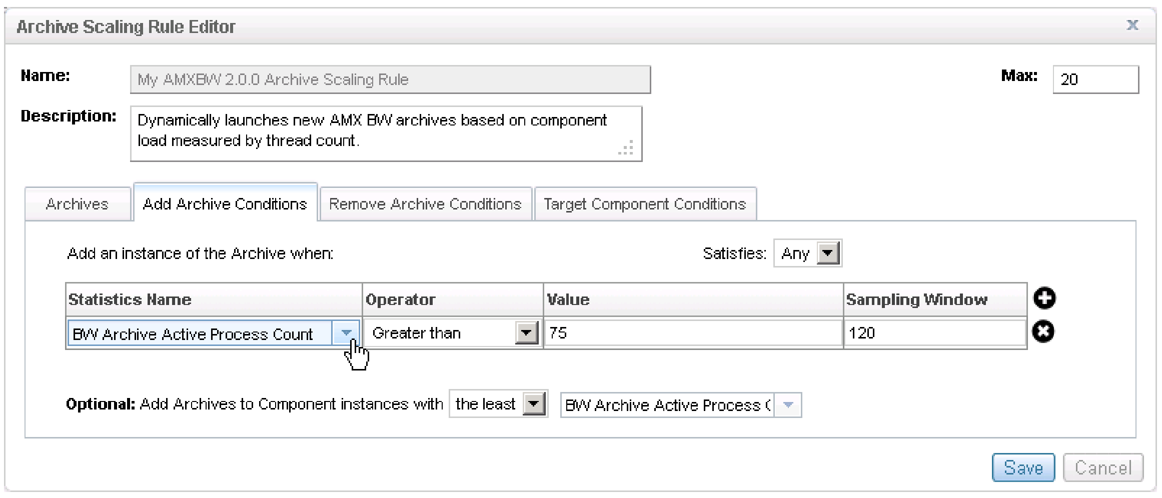
4. Select the BusinessWorks component that was used to upload the application archive file and use the **Archive Names** field to specify what archive will be subject to the rules you set using the other tabs of the Archive Scaling Rule Editor.
5. Use the **Add Archive Conditions** tab and click the **Add** icon to the right of the column heading row to create and define a new Add Archive rule.
6. Select the statistics, the operator, the value, and the sampling window period in seconds to define your condition for adding a new archive.

The sampling window must be a sufficiently large time period (in seconds) to allow aggregated statistics to be collected. If the sampling window is not big enough, there is a chance that no statistics will be reported in a particular time-frame creating an inadvertent trigger condition.

By default allocation statistics like "Expected Engine Count", "Client Count", "Allocating Engine Count", and "Actual Engine Count" are collected every 60 seconds and by default component statistics are collected every 10 seconds.

You can define more than one rule. With more than one rule, set the **Satisfies** field to specify whether all rules must be satisfied or whether any one rule can be satisfied to trigger addition of a new archive instance.

Define Archive Scaling Rule conditions



Archive Scaling Rule Editor

Name: My AMXBW 2.0.0 Archive Scaling Rule
Max: 20
Description: Dynamically launches new AMX BW archives based on component load measured by thread count.

Archives | Add Archive Conditions | Remove Archive Conditions | Target Component Conditions

Add an instance of the Archive when:
Satisfies: Any

Statistics Name	Operator	Value	Sampling Window
BW Archive Active Process Count	Greater than	75	120

Optional: Add Archives to Component instances with the least BW Archive Active Process C

Save Cancel

Optionally you can set a preference for running new archives or new process instances on component instances with favorable usage profiles. Select the statistic that is most relevant to your implementation and you can create new process instances there according to those conditions you defined.

7. Use the **Remove Archive Conditions** tab to release computing resources and remove unused or idle component archives or process instances to scale down your component archives just as you scaled them up according to conditions you define on usage statistics.
8. Use the **Target Component Conditions** tab allows you to restrict the start of new archive instances to those machines that have the same set of resources that you choose. Set a rule or several rules with statistics, operators, and values as you would on the **Add Archive Conditions** tab and further restrict where the new Archive instances can start depending on component instances that have:

   Same Component

   This setting works for all component archive scaling.

   From the set of Components

   This setting works only if your component archive is compatible with the set of components present. For example a BusinessWorks archive can scale on a product adapter component.

   Same Component Type

   The process archive has the possibility of being scaled up on different versions of the product

   Same Enabler

   Components that require the same specific enabler should use this option.

   Same Middleware Version

   This selection ensures that the component archive runs on machines with the appropriate middle ware: TIBCO BusinessWorks, TIBCO TRA, TIBCO Hawk, etc.

   Same Enabler and Middleware Version

   This selection ensures that your component archive scales up successfully, but it is the least restrictive of the target component conditions.

Only **From the Same Set of Components** or **Same Enabler and Same Middleware Version** works 100% of the time.

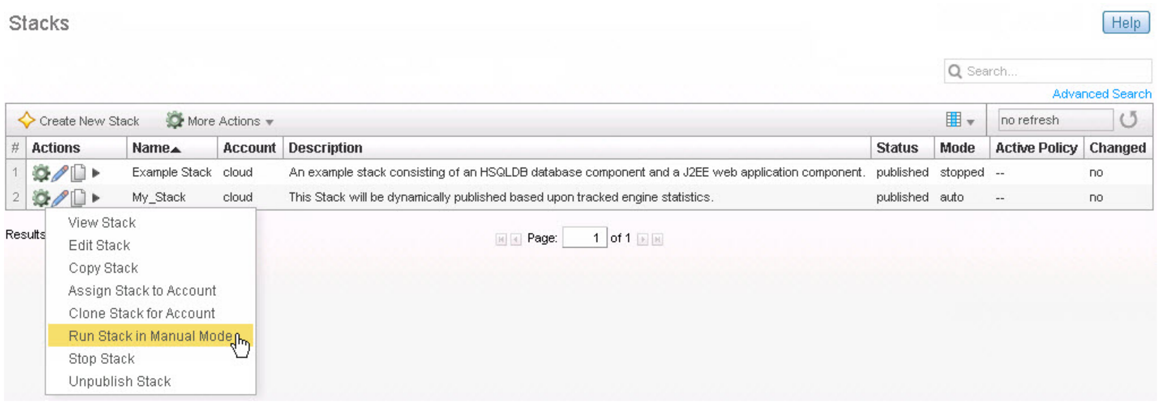
Provided that the component archive has the proper component type, TIBCO Silver Fabric can usually find the correct computing environment for scaling up. For example, for a BusinessWorks component archive, a selection of the "Same Component Type" ensures that the Silver Fabric Broker tries and finds an engine with same BusinessWorks component type on which to run a new BusinessWorks component archive.

Running Stacks

Procedure

1. After you have created your stack, use **Publish Stack**, available from the **Actions** drop-down list, to make it available to run.
2. After publishing, select **Run Stack In Manual Mode** in the **Actions** drop-down list as shown in the following figure to run the stack immediately on available resources.

   Running a Stack



#	Actions	Name	Account	Description	Status	Mode	Active Policy	Changed
1		Example Stack	cloud	An example stack consisting of an HSQLDB database component and a J2EE web application component.	published	stopped	--	no
2		My_Stack	cloud	This Stack will be dynamically published based upon tracked engine statistics.	published	auto	--	no

Alternatively, if the stack is defined with a Policy (schedule), you can **Run** (the) **Stack in Auto Mode**. The stack runs according to the schedule selected for the stack on available resources. For more information on creating and running stacks refer to the TIBCO Silver Fabric documentation.

Updating the TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks Stack

When a stack is published and running, you can still make changes to the stack such as adding other components, changing allocation rules, changing threshold activation rules, or deploying and starting archives on the runtime BusinessWorks Application instantiated on the engine.

Making changes to the stack is as easy as editing, saving, and publishing those changes to any instances that are running. Some changes might require restart of the changed resource, so consult the TIBCO Silver Fabric documentation for best practices prior to making changes to a production system.

Procedure

1. After making any changes to a stack, **Save** the changes
2. Select **Publish Changes** from the Actions list in the main stack page.The specified engines are affected by the changes immediately

   If you want to change a BusinessWorks component, you do not need to stop and restart your entire stack. If you want to deploy, start, stop, and undeploy Business Works project archives here are a couple of ways to accomplish that:

   - **Micro-scaling** - Start and stop BusinessWorks archives based on your defined rules when they are already in your component. For more information refer to: Creating Archive Scaling Rules.
   - **Continuous Deployment** (deploy archives directly to BusinessWorks endpoints) - publish (deploy), unpublish (undeploy), start, or stop BusinessWorks archives without having to change any stacks, components, or BusinessWorks engines. Deploying BusinessWorks application archives via REST and cURL commands is described in the next section.

Continuous Deployment - Deploy Archives Directly to Endpoints

There are situations where deploying archives directly to a running TIBCO ActiveMatrix BusinessWorks component can be useful, such as when an archive needs to be deployed and run on a system that is already running. This is known as continuous deployment.

Archives can be directly deployed to BusinessWorks instances already running on Silver Fabric Engines using the command line interface (CLI), Silver Fabric API, Ant scripts, or an HTTP REST command sent using cURL or a Java client. Refer to the *TIBCO Silver Fabric Cloud Administration Guide* for more information on the CLI or the Silver Fabric API. Archive deployment, undeployment, starting, and stopping application archives via REST are described in further detail here. For details on Ant scripts, refer the chapter “Silver Fabric Ant Tasks” in the *TIBCO Silver Fabric Developers Guide*.

TIBCO Silver Fabric supports many HTTP REST commands to GET, PUT, POST, and DELETE objects and managed resources for use with archive scaling, brokers, components, daemons, enablers, gridlibs, schedules, stacks, and Skyway.

TIBCO Silver Fabric REST Services are documented in the *TIBCO Silver Fabric Administration Tool* at **Admin > REST Services**. They are grouped by resource. Click any method name to see possible parameters if any and example responses.

Continuous deployment is discussed in good detail in the section on *Deploying Archives Directly to Components* in the *TIBCO Silver Fabric Administration Guide*.

Prior to using REST CLI with TIBCO Silver Fabric 5.6 you must change the setting, ***Strict Validation***, at **Config** > **Broker** > **General** to "false".

The TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks adds more REST methods to enable control of ActiveMatrix BusinessWorks archives.

Prior to using REST CLI with TIBCO Silver Fabric 5.6, you must change the Strict Validation setting. For the Configuration of the Broker make the General value as false.

Continuous Deployment Life Cycle

The continuous deployment life cycle has four REST operations that can be executed using cURL methods:

Deploy

Send an archive to a Silver Fabric Engine running an ActiveMatrix BusinessWorks component that meets the criteria specified. The Deploy REST method enables specification of a properties files with criteria dictating where and how the archive should be deployed.

Start

Start an archive that was deployed to an engine.

Stop

Stop an archive that is running on an engine.

Undeploy

Remove an archive from an engine, stopping any running instances of that archive on that engine.

Deploy

ActiveMatrix BusinessWorks application archives may be deployed directly to an appropriate Silver Fabric Engine (TIBCO Logical Machine) by making REST calls. In this document cURL syntax is used to show the REST inputs in a generic form:

For 5.x Version

```
curl -u UserName:Password \
     -X POST \
     -H "Accept:application/json" \
     -H "Content-Type: multipart/form-data" \
     -F "archiveFile=@YourArchiveName.zip" \
     -F "deploymentFile=@YourDeploymentFileName.properties" \
    [-F "LogicalAnd=false"]
     -v "http://YourSFBroker.com:<port>/livecluster/rest/v1/sf/engines/archives"
    [-F "AppName=YourDirABC/YourAppName"]
    [-F "AppSettings.element1.element2=SomeValue"]
    [-F "ArchiveSettings.element1.element2=SomeValue"]
    [-F "Archives=Archive_A,Archive_B,Archive_X"]
    [-F "configurationFile=YourConfigurationfile.xml"]
    [-F "ForceDeploy=true"]
    [-F "FTWeight=1"]
    [-F "GV=globalVariableA=123,globalVarB=SomeString"]
    [-F "InstanceSettings.element1.element2=SomeValue"]
    [-F "NoDeploy=true"]
    [-F "NoStart=true"]
    [-F "VariableProvider=Some_PROVIDER(S)_Name"]
```

For 6.x Version

```
curl -u UserName:Password \
     -X POST \
     -H "Accept:application/json" \
     -H "Content-Type: multipart/form-data" \
     -F "archiveFile=@YourArchiveName.zip" \
     -F "deploymentFile=@YourDeploymentFileName.properties" \
     -v "http://YourSFBroker.com:<port>/livecluster/rest/v1/sf/engines/archives"
    [-F "NoStart=true"]
    [-F "NoStop=true" ]
    [-F "Profile=<Profile_Name>.substvar
    [-F "VariableProvider=Some_PROVIDER_Name"]
```

Where inputs bounded by square brackets are optional; file names, elements, variable names, and any values shown in italics should be substituted by your implementation values.

In a cURL statement all values must be URL-encoded so that special characters like spaces, for example, are converted to "`%20`". Other special characters like forward slashes "`/`" must also be converted to"`%2F`" or their respective URL encoded values so they may be sent and received properly.

Expressions in the generic form cURL expression as mentioned earlier were separated by line breaks to help with readability, but normally a string is submitted in the execution as in the example that follows:

The following is an example cURL archives deploy statement:

```
curl -u admin:admin -X POST -H "Accept:application/json" -H "Content-
Type: multipart/form-data" -v http://MySFBroker.com:8080/livecluster/rest/v1/sf/
engines/archives -F "archiveFile=@MyProcessOrder.ear" -F "AppName=MyOrders/
MyProcOrder" -F "deploymentFile=@MyDeployCriteria.properties"
```

Where the user specified by `-u` must have Silver Fabric administrator level permissions, and the form-data fields (`-F "property=value"`) may be specified in any order. See Mandatory form-data fields for more details about form-data fields (`-F`).

Mandatory form-data fields

Mandatory form-data contain the files necessary for the deployment (`-F` option).

`archiveFile`

Specifies your ActiveMatrix BusinessWorks archive file (`.zip`, `.par`, or `.ear` file) to upload to the Silver Fabric Broker which will then publishe the archive to the appropriate Silver Fabric engine. Multiple application archives may be deployed in a single archive `.zip` or `.ear` file. You can deploy and run them all (default behavior) or you can selectively run a list of archives by specifying that list with the `Archives` form-data field. Archives may also be uploaded using the component wizard.

`deploymentFile`

Specifies the properties file that defines the endpoint selection criteria described as follows in more detail.

```
     -F "deploymentFile=@YourDeploymentFileName.properties"
```

`LogicalAnd` (optional)

By default **all** criteria specified in the deployment properties file must be satisfied for deployment to an application endpoint, but that can be toggled to mean **any** (meaning logical OR) of the deployment properties criteria by setting: `-F "LogicalAnd=false"`

`-v`

Specifies the target of the cURL POST execution and asks for a verbose response. The cURL -v expression should specify the appropriate Silver Fabric directory. For the default installation, that expression would look like the following:

```
 -v "http://YourSilverFabricBrokerName.com:<port>/livecluster/rest/v1/sf/engines/
archives"
```

Where the default http <port> is 8080 and optional form-data fields can specify other continuous deployment behavior.

`AppName`

Specifies the directory location where the application archives are deployed and what the application is named.

Where your archive application deploys and what it is called depends on what you specify with `AppName`. For example, when you use TIBCO Designer to create an `.ear` file with a name such as *MyAppArchive*, varying the `AppName` specification gives following behavior:

- If the `AppName` form-data field is not specified, *MyAppArchive* is deployed at the top level of the directory.

- If `-F "AppName=`***`A`***`"` is submitted in the curl request, *MyAppArchive* is renamed *A* and deployed at the top level.
- If `-F "AppName=`***`A/`***`"` is sent, the directory folder *A* is used or it is created and *MyAppArchive* is deployed within that subdirectory.
- If `-F "AppName=`***`A/B`***`"` is sent, the subdirectory *A* is used or created, and *MyAppArchive* is deployed there and renamed *B*.
- If `-F "AppName=`***`A/B/`***`"` is sent, the folder *A* with a subfolder *B* is used or created and *MyAppArchive* is published within the subfolder *B*.

The full application name is derived from the `AppName` directory location and the application archive name as it is deployed.

Optional form-data fields do not need to be specified unless you want to change the default behavior. By default, all archives in the archive file are deployed and started unless an archive of the same name is already deployed and started, in which case that archive is allowed to run without interruption or replacement.

`AppSettings` (optional)

Specifies settings that your application uses when deployed.

All deployment configuration cURL expressions takes the form:

```
-F "AppSettings.element1.element2=SomeValue"
```

Some examples:

```
-F "AppSettings.localRepoInstance.encoding=UTF-8"
-F "AppSettings.description=This%20application%20deployment%20is%20for%20validation
%20testing."
```

Where the element is one of the following (plus any subordinate *element2* where applicable):

- **description**: a string describing the application.
- **contact**: a string to name the person responsible for the deployment.
- **maxDeploymentRevision**: specifies the default number of application revisions to keep in the revision history for each deployed application. Leave the value at -1 to keep all revisions by default.
- **localRepoInstance**: for Enabler installed components and application archives installed with continuous deployment, a local file (or directory of files) is used as the deployment repository instance.

  When deploying applications your domain is automatically configured to establish a local application repository managed by TIBCO Runtime Agent. This helps to ensure proper functionality of deployed applications when using Fast TLM restart and HTTP discovery.

- **encoding**: specifies encoding for the repository instance. If this element is not specified, the encoding for the admin server is used. If the admin server is not available, the default for this element is ISO8859-1.

All TIBCO components working in the same domain must always use the same encoding for intercommunication.

`Archives` (optional)

Form-data parameter that specifies a comma delimited list of archives that are to be deployed within the `.zip` file that are to be deployed. If an `archives` list is omitted, all archives in the application archive package are deployed. For example:

```
-F "Archives=Archive_A,Archive_B,Archive_X"
```

`ArchiveSettings` (optional)

Form-data parameter specifies settings for the archive.

- `enabled`: `true` or `false`. 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 when you wish to deploy an application that includes a service, for which you do not have the required software. A deployment configuration cURL expressions takes the form:

  ```
  -F "ArchiveSettings.enabled=true"
  ```
- `av`: Specify values for archive runtime variables with a comma-separated string with each key value pair joined by an equal (=) sign. For example:

  ```
  -F "ArchiveSettings.av=Deployment=T2.HTTP_GET-Tomcat,Domain=Mine"
  ```

  Refer to the appendix of the *TIBCO Runtime Agent Scripting Deployment User's Guide* for a full list of Archive Settings properties, parameters, descriptions, and usage. Not all elements and properties are supported for use by the TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks. The following is a first attempt at listing them all.
- `hbInterval`: heartbeat interval. The heartbeat interval determines the time (in milliseconds) between heartbeat messages.
- `activationInterval`: activation interval. 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.
- `preparationDelay`: preparation interval. This field is used to specify a delay before the master engine restarts.
- `ruleBases`: rule bases for the archive
  - `ruleBase.uri`: location of the rulebase file. Example syntax:

    ```
    -F "ArchiveSettings.ruleBases.ruleBase.uri=someValue"
    ```
  - `ruleBase.data`: Rulebase content. Do not change this setting.
- `failureEvents.failureEvent`:
  - `failureType`: *ANY, FIRST, SECOND, Subsequent*

    If an event is used, type must be specified.

    Example syntax:

    ```
    -F "ArchiveSettings.failureEvents.failureEvent.failureType=ANY"
    ```
  - `restart`: true or false. If true, the service instance is restarted upon failure.
  - `description`: information that describes this operation.
  - `alertAction.performPolicy`: the policy to perform. Once:generates an alert only for the first occurrence. Always:generates an alert for each occurrence.
  - `alertAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If false, the action is not called.
  - `alertAction.level`: *High, Medium, Low*
  - `alertAction.message`: The message that displays when this alert is triggered.
- `failureEvents.emailAction`:
  - `performPolicy`: the policy to perform. Once:generates an alert only for the first occurrence. Always: generates an alert for each occurrence.
  - `enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.

  - `message`: the message to send.
  - `to`: a comma-separated list of email addresses to which the messages are sent.
  - `cc`: a comma-separated list of email addresses to which copies of the messages are sent.
  - `subject`: the subject of the email message.
  - `SMTPServer`: The mail server (SMTP server) to use to send the message. Specify the host name or the host IP address.
- `failureEvent.customAction`:
  - `performPolicy`: the policy to perform.
    - `Once`: generates an alert only for the first occurrence.
    - `Always`: generates an alert for each occurrence.
  - `enabled: true or false`. If true, the action occurs when conditions for the action are true. If false, the action is not called.
  - `command`: specify the script to execute. Script files are highly recommended.
  - `arguments`: the list of arguments for the command
- `suspendProcessEvents.suspendProcessEvent`:
  - `restart: true or false`. If true, the service instance is restarted upon failure.
  - `description`: information that describes this operation.
  - `alertAction.performPolicy`: the policy to perform.
    - `Once`: generates an alert only for the first occurrence.
    - `Always`: generates an alert for each occurrence.
  - `alertAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.
  - `alertAction.level`: High, Medium, Low
  - `alertAction.message`: the message that displays when this alert is triggered.
  - `emailAction.performPolicy`: the policy to perform. Once:generates an alert only for the first occurrence. Always*:* generates an alert for each occurrence.
  - `emailAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.
  - `emailAction.message`: the message to send.
  - `emailAction.to`: a comma-separated list of email addresses, to which the messages are sent.
  - `emailAction.cc`: a comma-separated list of email addresses, to which copies of the messages are sent.
  - `emailAction.subject`: the subject of the email message.
  - `emailAction.sMTPServer`: the mail server (SMTP server) to use to send the message. Specify the host name or the host IP address.
  - `customAction.performPolicy`: the policy to perform.
    - `Once`: generates an alert only for the first occurrence.

    - `Always`: generates an alert for each occurrence.
  - `customAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.
  - `customAction.command`: specify the script to execute. Script files are highly recommended.
  - `customAction.arguments`: the list of arguments for the command
- `logEvents.logEvent`:
  - `->restart: true or false`. If `true`, the service instance is restarted upon failure.

    Example syntax:

    ```
    -F "ArchiveSettings.logEvents.logEvent.restart=true"
    ```
  - `match`: The string in the log file to match.
  - `description`: information that describes this operation.
  - `alertAction.performPolicy`: the policy to perform.
    - `Once`: generates an alert only for the first occurrence.
    - `Always`: generates an alert for each occurrence.
  - `alertAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.
  - `alertAction.level`: *High, Medium, Low*
  - `alertAction.message`: The message that displays when this alert is triggered.
  - `emailAction.performPolicy`: the policy to perform.
    - `Once`: generates an alert only for the first occurrence.
    - `Always`: generates an alert for each occurrence.
  - `emailAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.
  - `emailAction.message`: the message to send.
  - `emailAction.to`: a comma-separated list of email addresses, to which the messages are sent.
  - `emailAction.cc`: a comma-separated list of email addresses, to which copies of the messages are sent.
  - `emailAction.subject`: the subject of the email message
  - `emailAction.sMTPServer`: The mail server (SMTP server) to use to send the message. Specify the host name or the host IP address.
  - `customAction.performPolicy`: the policy to perform.
    - `Once`: generates an alert only for the first occurrence.
    - `Always`: generates an alert for each occurrence.
  - `customAction.enabled: true or false`. If `true`, the action occurs when conditions for the action are true. If `false`, the action is not called.
  - `customAction.command`: specify the script to execute. Script files are highly recommended.
  - `customAction.arguments`: the list of arguments for the command
- `failureCount`: The value in this field defines how many restarts should be attempted before resetting the error counter to 0.

- `failureInterval`: The value in this field defines how much time should expire before resetting the error counter to 0.
- `bwprocess`:
  - `name`: the name of the BusinessWorks process
  - `enabled: true or false`. Only enabled processes are deployed.
  - `maxjob`: specifies the maximum number of process instances that can concurrently be loaded into memory.
  - `activation`: the use activation limit
  - `flowlimit`: specifies the maximum number of currently running process instances to start before suspending the process starter.

`InstanceSettings` - (optional)

Some syntax examples:

```
-F "InstanceSettings.initHeapSize=64"
-F "InstanceSettings.maxHeapSize=512"
-F "InstanceSettings.threadStackSize=512"
```

- `description`: specify any pertinent information about the binding.
- `contact`: name the person responsible for this application instance.
- `startOnBoot`: when true, the service instance starts when the computer is restarted. The default value is false.
- `enableVerbose`: when true, sets the enabler for verbose tracking for service instances. The default value is false.
- `maxLogFileSize`: sets the maximum size (in kilobytes) that a log file can reach before the engine switches to the next log file.
- `maxLogFileCount:` specifies the maximum number of log files to use. When the maximum number of log files have been written, the engine begins writing to the first log file again.
- `threadCount`: specifies the number of threads to use to execute process instances. The number of threads determines how many process instances can execute concurrently.
- `prepandClassPath`: the values supplied here are prepended to your CLASSPATH environment variable.
- `appendClassPath`: the items you supply here are appended to your CLASSPATH environment variable.
- `initHeapSize`: specifies the initial size (in MB) for the JVM used for the process engine. The default is 32 MB.
- `maxHeapSize`: specifies the maximum size (in MB) for the JVM used for the process engine. The default is 256 MB.
- `threadStackSize`: specifies the size of the thread stack. The default is 256 KB.
- `runAsNT`: when set to true the service is run 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.
- `startUpType`: specifies the instance service startup type as either: Automatic, Manual, or Disabled.
- `login`: specifies the login account for the service, if any. The domain name must be specified as well.
- `password`: sets the password for that service, if any.

- `checkpoint`: when set to true, the process engine waits for all jobs to finish (up to the maximum timeout) before shutting down the engine, rather than remove jobs at their next checkpoint.
- `timeout`: 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.
- `iv` : this element uses a comma-separated string with name-value pairs with each key value pair joined by an equal (=) sign.

`configurationFile` (optional)

Form-data parameter used to include an XML configuration file created to modify archive properties if needed. Example syntax:

```
-F "configurationFile=YourConfigurationfile.xml"
```

Where your XML configuration file should use the same format as an enabler or component level `configure.xml` file with the outermost XML element as follows:

```
<archiveConfig name="YourArchiveName">
   ...
</archiveConfig>
For more information on writing an archive configuration file, see the "Using the
Silver Fabric SDK" chapter of Silver Fabric Developer's Guide.
```

`ForceDeploy` (optional)

Redeploy, forces a stop and overwrite of a pre-existing archive or set of archives with the same name. By default ForceDeploy is set to false and so a second deployment does not overwrite a pre-existing deployment of the same name. If there is a change of the archive file then ForceDeploy should be set to *true* so that the new application archive is redeployed. If ForceDeploy is used with -F Archives specifying a comma delimited list, then only those archives are stopped, undeployed, and redeployed.

```
-F "ForceDeploy=true"
```

`FTWeight` (optional)

Sets the relative FT (fault tolerance) weight for the enabler binding of the deployed archives. BusinessWorks archives can be deployed on multiple machines to support fault tolerance and load balancing. Setting different FTWeight values for different application endpoints will require invoking the REST service at least twice with different deployment properties criteria, once for each endpoint.

`VariableProvider` (optional)

Specifies a variable provider to set global variables for applications deployed with REST. The variable provider is a Java Class extension compiled into a JAR and loaded into a Silver Fabric directory with an appropriate XML so that it may be called by REST during application deployment. It supports multiple VariableProviders at the same time. You can add multiple variable provider names with a comma separated list. If you use the global variable name in both variable providers, the latest in the list is used.

```
-F "VariableProvider=Some_PROVIDER_Name"
```

`NoStart -` (optional)

The default value is `false`, meaning that the archives are both deployed and started by default. If NoStart is set to `true`, the application is deployed but not started.

```
-F "NoStart=true"
```

`NoStop`(optional)

The default value is `false`, meaning that running applications are stopped when a new archive is deployed. If NoStop is set to `true`, the application is not stopped when deployed. This option works for 5.x versions of BusinessWorks.

```
-F "NoStop=true".
```

Creating a variable provider for use by a REST call

Procedure

1. Create a class that extends `com.datasynapse.fabric.broker.userartifact.variable.AbstractVariableProvider` and override the methods as needed. For example:

   ```
   package com.tibco.sf.providers;
   import java.util.Properties;
   import
   com.datasynapse.fabric.broker.userartifact.variable.AbstractVariableProvider;
   public class My_Var_Provider extends AbstractVariableProvider
   {
     private static final long serialVersionUID = 1L;
     @Override
     public Properties getVariables()
     {
       Properties p = new Properties();
       p.setProperty("MyApp/vars/name", "SomeString");
       p.setProperty("MyApp/vars/episodic", "true");
       p.setProperty("MyApp/vars/Xduration", "60");
       return p;
     }
     @Override
     public void destroy()
     {}
     @Override
     public void init() throws Exception
     {}
   }
   ```

2. Export a JAR from it and save it to the TIBCO Silver Fabric Broker directory:

   ```
   SILVERFABRIC_HOME/webapps/livecluster/deploy/config/variableProviders
   ```

3. Create an XML file with the properties shown in this sample to associate the new class for TIBCO Silver Fabric.

   ```
   <variableProvider class="com.tibco.sf.providers.My_Var_Provider">
      <property name="name" value="Some_PROVIDER_Name"/>
      <property name="description" value="GV values are in the JAR"/>
      <property name="enabled" value="true"/>
   </variableProvider>
   ```

   The variable provider name used by the REST statement is specified as "name" in the XML file.

   Save the XML file in the same directory as the JAR:

   `SILVERFABRIC_HOME/webapps/livecluster/deploy/config/variableProviders`

   ```
   You can verify what variable providers are ready for use by
   REST invocation on the TIBCO Silver Fabric Administrator >
   Admin > Variables page.
   ```

   NoStart (Optional) The default value is false, meaning that the archives are both deployed and started by default. If *NoStart* is set to true, the application is deployed, but not started. -F "NoStart=true". This option works for 5.x versions of BusinessWorks.

   NoStop (Optional) The default value is false, meaning that running applications are stopped when a new archive is deployed. If NoStop is set to true, the application is not stopped when deployed. -F "NoStop=true". This option works for 5.x versions of BusinessWorks.

Deployment File

When deploying an archive by REST you must include a deployment file, which specifies at least one selection criteria for determining which engine and component receives the deployed archive.

The deployment file is a simple properties text file specified by a form-data field like the following for REST upload with the archive:

```
     -F "deploymentFile=@YourDeploymentFileName.properties"
```

The deployment file contains one or more logical statements with a criteria, a comparator, and a value, delimited by spaces:

```
criteria comparator value
```

Some criteria require an argument to be specified in parentheses:

```
criteria(argument) comparator value
```

For example, what follows is a simple statement to deploy an archive to an engine running a component with a component type that contains *BusinessWorks* as part of the name and which has a domain name of *YourDomain*:

```
ComponentType contains BusinessWorks
ImportedVariable(TIBCO_DOMAIN_NAME) = YourDomain
```

By default all deployment file criteria statements must be satisfied for the deployment to occur, but you can change how the properties file criteria are evaluated to make them logical `OR` statements by using the optional cURL form-data switch: `-F "LogicalAnd=false"`

Supported Criteria

Name	Definition
`ComponentName`	The name of the component.
`ComponentType`	The type of the component.
`EnablerName`	The name of the Enabler running the component.
`EnablerVersion`	The version of the Enabler running the component.
`Account`	The name of the account running the component.

`EngineProperty(name)`	The following named engine properties can be used:
	`Engine Id`
	`Engine GUID`
	`Engine Instance`
	`IP Address`
	`Host Name`
	`Number of CPUs`
	`Total CPU Processing Power`
	`Total Memory (KB)`
	`Free Memory (KB)`
	`Free Disk Space (MB)`
	`OS Platform`
	`OS Version`
	`OS Username`
	`Location`
	`vimTemplate`
	`Group`
	`Description`

Name	Definition
`ActivationInfoProperty(`*`name`*`)`	A named property, such as ClusterName, or HTTP_STATIC_ROUTE_PREFIX within the component's ActivationInfo object. Archives that can be scaled elastically keep a list of ActivationInfo properties and their respective values for failover Broker.
`ImportedVariable(`*`name`*`)`	A variable imported into a component.
`ExportedVariable(`*`name`*`)`	A variable exported from a component.

`Statistic(`*`name`*`)`	Supported BusinessWorks Enabler statistic like:
	`BW Process Count`
	`BW Memory Free Bytes`
	`BW Percentage of Memory Used`
	`BW Average Elapse Execution Time`
	`BW Total Number Error`
	`BW Last Number Error`
	`Total Memory`
	`Free Memory`
	`Free Disk`
	`CPU Utilization`
	`DS CPU Utilization`
	`Expected Engine Count`
	`Actual Engine Count`
	`Allocating Engine Count`

`ArchiveStatistic(`*`statName`*`, `*`archiveName`*`)`	A named statistic for a specified archive.
`DependencyComponent`	A named dependency on a component.
`DependencyEngine(`*`componentName`*`)`	A named dependency on an engine running the named component.

Comparator

Valid comparators include `=`, `!=`, `>`, `<`, `<=`, `>=`, `matches`, `contains`, `!matches`, and `!contains`.

The following is the sample deployment file:

```
# Sample deployment file
ComponentType = "TIBCO ActiveMatrix BusinessWorks"
Statistic(CPU Utilization) < 80
ActivationInfoProperty(ClusterName) matches dev_cluster.*
```

Successful Deployment

REST returns a Status of 200 (OK) when the archive is successfully deployed. A successful REST execution returns the `Engine-Instance` and `Engine-Id` where the archive deployed.

Example response (application/JSON):

```
{
    "result": {
        "name": "Archive Deployment",
        "value": {
            "message": "ENGINE '[1885509966828815621-5 : my_archive.ear]' DEPLOYED",
            "Engine-Instance": "5",
            "Engine-Id": "1885509966828815621"
        }
    },
    "status": 200
}
```

Knowing the `Engine-Id`, `Engine-Instance`, and the full `ArchiveName` allows for invocation of START, STOP, and UNDEPLOY methods to enable full control of the Archive life cycle.

An unsuccessful deployment returns an error of Status 500 or some other appropriate error status depending on the cause.

Continuous Deployment Timeout

Continuous deployment transfers can timeout due to one or more of the following factors:

- Large archive size
- A slow network or high latency
- Long start-up time for archives

If you are encountering timeout issues, you can set a higher socket timeout between the broker and engines. This can be set in the Silver Fabric Administration Tool at **Config** > **Broker** > **Communications** under the section **HTTP Connections** > **Engines**. The Socket Timeout parameter configures the HTTP connections established from brokers to clients and engines. Set the timeout value to the longest of the following three:

- The longest scaling archive download time from the broker to engine
- The longest starting or stopping time for an archive
- The longest undeploy time

Files larger than 1GB should not be deployed using continuous deployment.

Start

Using REST you can also start application archives that are deployed but not started. You must know the `Engine-Id`, `Engine-Instance`, and the full application `ArchiveName`.

The following is a generic cURL example that starts an Archive.

```
curl -u UserName:Password \
     -X POST \
     -H "Accept:application/json" \
     -H "Content-type: multipart/form-data" \
     -v "http://<YourSFBroker.com>:<port>/livecluster/rest/v1/sf/e
ngines/{engine-id}/{engine-instance}/archives/{appname}/start"
[-F "Version={VersionNumber}"]     #supported only for 6.x component
```

Expressions in this generic form cURL expression example were separated by line breaks to help with readability, but normally an unbroken string is submitted when executed.

The value of *`{engine-instance}`* can be obtained from the response to the successful cURL deployment REST execution or the TIBCO Silver Fabric Administrator > **Engines** page > expanding the row to see Engine details.

The Start and Stop both have an optional option to specify the version number.
If the version number is not specified, an attempt to complete the action is made on the deployed application with the same name.

There are also Silver Fabric REST methods to GET the engine-id and engine-instance for all instances running on a daemon. Refer to TIBCO Silver Fabric REST Services documented in the *TIBCO Silver Fabric Administration Tool* at **Admin > REST Services**.

`Version -` (optional) The version form-data field (`-F`) lets you specify the version to be started or stopped. If not specified, the version number is obtained from the deployed application that has the same application name.

Another cURL example that starts an archive

```
curl -u admin:admin
     -X POST
     -H "Accept:application/json"
     -H "Content-Type:multipart/form-data" http://lin64vm205.qa.datasynapse.com:
8080/livecluster/rest/v1/sf/engines/5103224222683646426/3/archives/
tibco.bw.sample.binding.soap.http.BookStore.application/start
     -F "Version=1.0"
```

The value of `{appname}` is usually the application name or the full application archive name.

When the `{appname}` is used, it is URL-encoded in a cURL statement so that spaces are converted to "`%20`" and forward slashes (`/`) are represented by "`%2F`". Likewise, other special characters must be encoded appropriately.

A cURL example (URL encoded) that starts an archive.

The full archive name shown had to be URL encoded:

```
/Silver Fabric/SFBW 22E5/DomainMachineName/processOrder/Orders/MyProcOrder/
Process_Archive.par
```

...to be passed in the cURL statement shown here:

```
curl -u admin:admin -X POST -H "Accept:application/json" -H "Content-Type:multipart/
form-data" -v "http://192.168.72.189:8080/livecluster/rest/v1/sf/engines/
8735490097077982598/1/archives/
tibco.bw.sample.palette.http.PersistentConnection.application/start" -F
"Version=1.0"
curl -u admin:admin -X POST -H "Accept:application/json" -H "Content-Type:
multipart/form-data" -v "http://lin64vm121.qa.datasynapse.com:8080/livecluster/
rest/v1/sf/engines/4649861604167227205/1/archives/%2FSilver%20Fabric%2FSFBW
%2022E5%2FDomainMachineName%2FprocessOrder%2FOrders%2FMyProcOrder/
Process_Archive.par/start"
```

Stop

You can stop application archives that are running on an engine by REST method by submitting a cURL expression to the Silver Fabric Broker. You must know the `Engine-Id`, `Engine-Instance`, and the full application `ArchiveName`.

```
curl -u UserName:Password
     -X POST
     -H "Accept:application/json"
     -H "Content-type: multipart/form-data" \
     -v "http://<YourSFBroker.com>:<port>/livecluster/rest/v1/sf/e
ngines/{engine-id}/{engine-instance}/archives/{appname}/{archive-id}/stop"
```

`Version-` (optional) The version form-data field (`-F`) lets you specify the version to be started or stopped. If not specified, the version number is obtained from the deployed application that has the same application name.

Refer to the **Start** method for a description on how to obtain the values of *`{engine-id}`*, *`{engine-instance}`*, and *`{appname}`*.

The following is cURL example that stops an Archive:

```
curl -u admin:admin
     -X POST -H "Accept:application/json"
     -H "Content-Type:multipart/form-data" http://lin64vm205.qa.datasynapse.com:
8080/livecluster/rest/v1/sf/engines/5103224222683646426/3/archives/
tibco.bw.sample.binding.soap.http.BookStore.application/
tibco.bw.sample.binding.soap.http.BookStore.application/stop
```

Undeploy

You can undeploy application archives that are running on an engine by REST method by submitting a cURL expression to the Silver Fabric Broker. You must know the `Engine-Id`, `Engine-Instance`, and the full application `ArchiveName`.

```
curl -u UserName:Password
     -X POST
     -H "Accept:application/json"
     -H "Content-type: multipart/form-data" \
     -v "http://<YourSFBroker.com>:<port>/livecluster/rest/v1/sf/engines/{engine-
id}/{engine-instance}/archives/{appname}/undeploy"
```

Refer to the Start method for a description on how to obtain the values of *{engine-id}*, *{engine-instance}*, and *{appname}*.

`DeleteApp` - (optional - use with undeploy only) The default value is false and it can be omitted. When the DeleteApp parameter is false, an undeploy archive action leaves the application configurations of global variables and bindings so that they can be used again. The archive and the application are only undeployed and not deleted.

Setting `DeleteApp` to true deletes the application and the associated variable settings from the runtime engine after the archive is undeployed.

The following is an cURL example of an undeploy call

```
curl -u admin:admin
     -X POST
     -H "Accept:application/json"
     -H "Content-Type:multipart/form-data"
     -v "http://lin64vm205.qa.da
tasynapse.com:8080/livecluster/rest/v1/sf/engines/5103224222683646
426/3/archives/tibco.bw.sample.binding.soap.http.BookStore.applica
tion/undeploy"
```

Here the application name is used and there is no need for archiveid.

For more information on REST methods, including parameters and return responses, see the online REST help in the Silver Fabric Administration Tool. For information on using REST services, see the *Silver Fabric Developer's Guide*.

The Archive Management Support Feature

When components are activated, by default all archives are deployed in order, and then started in order. This behavior can be changed by editing the component in the component wizard, and editing the Archive Management Support feature, which has an option that can be cleared to prevent archives from starting when the component is activated.

Ant Scripts

In addition to the administration tool, command-line interface tool, and REST interface, you can also configure components and stacks using a set of Silver Fabric Ant tasks, which are provided for Apache Ant. This alternative provides a method more granular than the CLI, which can be easier to use in some configuration scenarios, and also provides an easy way to configure Silver Fabric from within an IDE with a built-in Ant support.

For more details on installation and tasks, refer chapter "Silver Fabric Ant Scripts" in *TIBCO Silver Fabric Developer's Guide.*

Samples

The Silver Fabric command line interface installation contains a samples directory, which contains example build files using Ant tasks for several enablers. See the enclosed Readme for more information on the examples.

You can also create a sample `build.xml` and `build.properties`, based on any of your published stacks. See "Packaging Stacks For Ant Task Deployment" in *TIBCO Silver Fabric Cloud Administrator's Guide* for more information.

You can use the `fabric-cli.properties` file from the Silver Fabric installation along with the following files:

- build.xml
- build.properties

build.xml

Following is the sample `build.xml` file:

```
<project name="sfbw-build" default="release" basedir="."
xmlns:sf="antlib:com.datasynapse.fabric.ant">
<property file="build.properties"/>
<property file="../../fabric-cli.properties" />
<sf:connection-props brokerurl="http://lin64cdc201.qa.datasynapse.com:8000"
username="admin" password="admin" clientssltrustfile="${DSSSLTrustFile}" />
<target name="release" depends="release-component, release-stack"/>
<target name="clean" depends="clean-stack, clean-component"/>
<target name="release-component">
<echo message="Building ${component.name}" />
<sf:component action="create" name="${component.name}" description="$
{component.description}" type="${component.type}" enablername="${enabler.name}"
enablerversion="${enabler.version}" utility="${utility}" />
<sf:option name="${component.name}" action="replace">
<sf:property name="Department" value="${department}" />
<sf:property name="Location" value="${location}" />
<sf:property name="Partition" value="${partition}" />
<sf:property name="Engine Blacklisting" value="false" />
<sf:property name="Failures Per Day Before Blacklist" value="0" />
<sf:property name="Archive Scale Up Timeout" value="${archive.scale.up.timeout}" />
<sf:property name="Archive Scale Down Timeout" value="$
{archive.scale.down.timeout}" />
<sf:property name="Maximum Deactivation Time" value="${deactivation.timeout}" />
<sf:property name="Maximum Activation Time" value="${activation.timeout}" />
<sf:property name="Maximum Capture Time" value="${maximum.capture.time}" />
<sf:property name="Maximum Instances Per Host" value="${max.instances.per.host}" />
<sf:property name="Separator Tags" value="${separator.tags}" />
<sf:property name="Statistics Collection Frequency" value="$
{stats.collection.frequency}" />
<sf:property name="Activation Delay" value="0" />
<sf:property name="Engine Reservation Expiration" value="$
{engine.reservation.expiration}" />
</sf:option>
```

```
<sf:default-settings name="${component.name}" action="update">
<sf:property name="Default Min Engines" value="${min}" />
<sf:property name="Default Max Engines" value="${max}" />
<sf:property name="Default Priority" value="${priority}" />
</sf:default-settings>
<sf:feature name="${component.name}" action="add" feature="Application Logging
Support" />
<sf:feature name="${component.name}" action="update" feature="Application Logging
Support">
<sf:property name="Archive Application Logs" value="${archive.logs}" />
<sf:property name="Checkpoint Frequency In Seconds" value="$
{checkpoint.frequency}" />
<sf:property name="Log File Pattern" value="${log.file.pattern}" />
</sf:feature>
<sf:feature name="${component.name}" action="add" feature="HTTP Support" />
<sf:feature name="${component.name}" action="update" feature="HTTP Support">
<sf:property name="HTTP Enabled" value="${http.enabled}" />
<sf:property name="HTTPS Enabled" value="${https.enabled}" />
<sf:property name="Routing Prefix" value="${routing.prefix}" />
<sf:property name="Route Directly To Endpoints" value="${routing.direct}" />
</sf:feature>
<sf:feature name="${component.name}" action="add" feature="Archive Management
Support" />
<sf:feature name="${component.name}" action="update" feature="Archive Management
Support">
<sf:property name="Start Archives On Activation" value="true" />
</sf:feature>
<sf:publish type="component" name="${component.name}" />
</target>
<target name="release-stack">
<echo message="Building ${stack.name}" />
<sf:stack action="create" name="${stack.name}" description="${stack.description}"/>
<sf:stack-component action="add" name="${stack.name}" components="$
{component.name}" />
<sf:stack-policy action="update" name="${stack.name}">
<sf:policy manualpolicy="true">
<sf:compalloc component="${component.name}" min="${min}" max="${max}" priority="$
{priority}">
<sf:allocationrule type="Resource Preference">
<sf:property name="propertyName" value="os" />
<sf:property name="propertyValue" value="linux" />
<sf:property name="operator" value="contains" />
<sf:property name="affinityPos" value="2" />
</sf:allocationrule>
</sf:compalloc>
</sf:policy>
</sf:stack-policy>
<sf:publish type="stack" name="${stack.name}" />
</target>
<target name="clean-stack">
<echo message="Cleaning ${stack.name}" />
<sf:unpublish type="stack" name="${stack.name}" onerror="ignore" />
<sf:remove type="stack" name="${stack.name}" onerror="ignore" />
</target>
</project>
```

build.properties

Following is the sample of `build.properties` file:

```
# stack properties
stack.name=Example Stack for ${component.name}
stack.description=Example Stack for ${component.name} built by Ant
# component properties
component.name=SFBW Example Component
component.description=SFBW Example Component built by Ant
component.type=TIBCO ActiveMatrix BusinessWorks:3.2.0
enabler.name=TIBCO ActiveMatrix BusinessWorks container
enabler.version=3.2.0.0
utility=false
# load balancing
```

```
routing.direct=false
routing.prefix=${cluster.name}
# logging
archive.logs=true
log.file.pattern=\
/../domaindata/logs/domainutility.log,\
/../domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/Hawk.log,\
/../domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/tsm.log,\
/../domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/msghma.log,\
/../domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/ApplicationManagement.log,\
/../domaindata/tra/${TIBCO_DOMAIN_NAME}/application/logs/.*\.log
checkpoint.frequency=300
# instances
max.instances.per.host=1
min=1
max=4
priority=3
# http
http.enabled=true
https.enabled=true
# build directories
content.dir=content
configure.dir=configure
script.path=
# timeouts
archive.scale.up.timeout=120
archive.scale.down.timeout=120
activation.timeout=300
deactivation.timeout=120
maximum.capture.time=300
stats.collection.frequency=120
engine.reservation.expiration=300
# options
department=
location=
partition=
separator.tags=
```

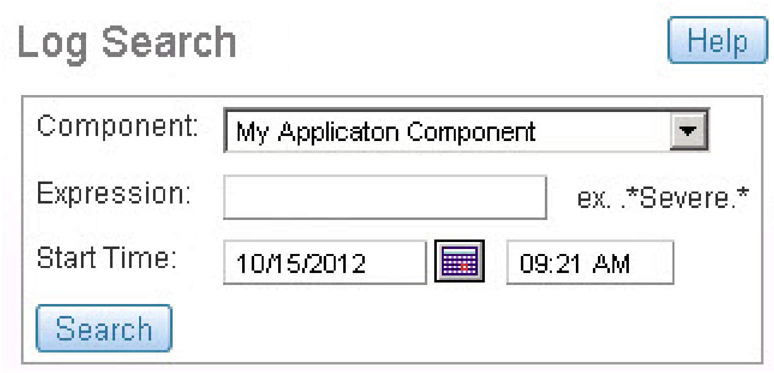
Retrieving Log Files

You can retrieve some BusinessWorks log files from the TIBCO Silver® Fabric Administration Tool.

Procedure

1. In TIBCO Silver® Fabric Administration Tool, select **Engines** > **Log Search** .
2. Select the component from which you want to see the log files, as shown in figure below.
3. Optionally, you can search for a regular expression using the **Expression** field.
4. Select the **Start Time** to see the logs since that time.

Log Files



Component: My Applicaton Component
Expression: ex. .*Severe.*
Start Time: 10/15/2012 09:21 AM

Retained Log Files

The log files generated for 5.x and 6.x components differ as 5.x component use TIBCO Administrator and 6.x components use TIBCO Enterprise Administrator.

Log Files For 5.x Components

In addition to the Engine log file, the following log files are retained:

- TIBCO Hawk Agent Log Files
- TIBCO Administrator Log Files
- BusinessWorks Log Files

TIBCO Hawk Agent Log Files

When TIBCO Administrator runs in the Fault Tolerant mode, all files are not located under the TIBCO Silver® Fabric *$ENGINE_WORK_DIR* directory. TIBCO Hawk® log files do not appear in the TIBCO Silver® Fabric Administrator GUI.

For both TIBCO Administrator components and BusinessWorks components, TIBCO Silver Fabric Enabler for ActiveMatrix BusinessWorks uses TIBCO Hawk and TIBCO Hawk Agent. The following table lists the retained TIBCO Hawk log files:

Retained TIBCO Hawk Agent Log Files

Name	Location	Purpose of the Log
`Hawk.log`	`$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOMAIN/logs`	Log file of the Hawk call in the Hawk Agent.
`tsm.log`	`$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOMAIN/logs`	Log file of the Hawk Agent.
`msghma.log`	`$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOMAIN/logs`	Log file for tibhawkhma.

TIBCO Administrator Log Files

When TIBCO Administrator runs in the Fault Tolerant mode, all files are not located under the TIBCO Silver® Fabric *$ENGINE_WORK_DIR* directory. The TIBCO Administrator log files do not appear in the TIBCO Silver® Fabric Administration Tool.

The following table lists the retained TIBCO Administrator log files:

Retained TIBCO Administrator Log Files

Name	Location	Purpose of the Log
`audit.log`	`$ENGINE_WORK_DIR/domaindata/admin/TIBCO_DOMAIN/logs`	All information about TIBCO Administrator activities.

Name	Location	Purpose of the Log
`tomcat.log`	*`$ENGINE_WORK_DIR`*`/domaindata/admin/`*`TIBCO_DOMAIN`*`/tomcat/logs`	Technical log of TIBCO Administrator that runs on Tomcat.
`domainutility.log`	*`$ENGINE_WORK_DIR`*`/tibcobw/tra/`*`tra_version_2digits`*`/logs`	Log file of the **`domainUtilitty`** command used to create the domain or add the machine. This file is common for all Engines.

BusinessWorks Log Files

The following table lists the retained BusinessWorks log files:

Retained TIBCO BusinessWorks Log Files

Name	Location	Purpose of the Log
`ApplicationManagement.log`	*`$ENGINE_WORK_DIR/`*`domaindata/tra/`*`TIBCO_DOMAIN/`*`logs`	Generated by Appmanage, which publishes BusinessWorks Applications.
`domainutility.log`	*`$ENGINE_WORK_DIR/`*`tibcobw/tra/`*`tra_version_2digits`*`/logs`	Log file of the **`domainUtilitty`** command used to create a domain or add a machine. This file is common for all Engines.
BusinessWorks Application Name, for example, OrderConsolidation-Order_Consolidation.log	*`$ENGINE_WORK_DIR/`*`domaindata/tra/`*`TIBCO_DOMAIN/`*`application/logs`	These are the most important log files, as they are the log files from BusinessWorks and trace all activities that have happened.

Log Files For 6.x Components

The TIBCO Silver Fabric Engine log files are generally a very useful diagnostic tool for finding out what happened on the engine.

The TIBCO Silver Fabric Engine log files are present at the following location:

```
$ENGINE_WORK_DIR/work/<machine-name>-0/log
```

BusinessWorks Log Files

The following table lists the retained BusinessWorks log files:

Retained TIBCO BusinessWorks Log Files

Name	Location	Purpose of the Log
bwagent.log	*`$ENGINE_WORK_DIR`*`/opt/qa/engine/work/`*`<machine_name-0>`*`/tibco/bw/6.2/logs`	Records bwagent activity.

Name	Location	Purpose of the Log
bwadmin.log	*$ENGINE_WORK_DIR*`/opt/qa/engine/work/`*<machine_name-0>*`/tibco/bw/6.2/logs`	Audit record of administrative actions.
`bwappnode.log`	`$ENGINE_WORK_DIR/opt/qa/engine/work/`*<machine_name-0>*`/tibco/bw/6.2/domains/`*<Domain_name>*`/appnodes/`*<AppSpace_Name>*`/`*<AppNode_name>*`/log/bwappnode.log`	Records service events
BusinessWorks Application Name, for example: *MyAppName.log*	When a BW **5.x** Distribution is published: *$ENGINE_WORK_DIR*`/domaindata/tmp/`*Application_Name*`/application/logs` When a BW **6.x** Distribution is published: *$ENGINE_WORK_DIR*`/tibco`	These are the most important log files, as they record all BusinessWorks application activities.

HTTP Virtual Router and Load Balancer

In a private cloud, when you run a BusinessWorksAdapter or TIBCO Administrator instance for 5.x component, you do not know in advance the address of the machine where it will run unless you set resource preferences in the rules.

For better interaction between BusinessWorks activities, an HTTP URL (a web service using SOAP over HTTP transport or an HTTP Receiver activity) is required. The HTTP Load Balancer, an HTTP redirector, fulfills this requirement.

The HTTP virtual router is supported by BusinessWorks 6.x components also.

HTTP Load Balancer for BusinessWorksAdapter Component

The URL to access or invoke the BusinessWorksAdapter archive endpoint (a web service using SOAP over HTTP transport or an HTTP Receiver activity) uses the form:

http://*{BrokerMachineName}*:*{BrokerPort}*/*{TIBCO_ADMIN_DOMAIN}*/*{BW-APPLICATION-PATH.NAME}{GA-APPLICATION-PATH.NAME}*/*{BW_HttpPortName}{GA_HttpPortName}*

Where the URL is composed of the following parts:

- ***{BrokerMachineName}*** - The Machine name or IP Address where you installed the virtual router. The default is the broker machine.
- ***{BrokerPort}*** – Typically this is the port of the Silver® Fabric Administrator GUI. The default value is **8080**.
- ***{TIBCO_ADMIN_DOMAIN}*** – The value of the domain name you entered when you configured TIBCO Enterprise Administrator component.
- *{BW-APPLICATION-PATH.NAME}* – The deployment directory and name of the BusinessWorks application with folders and name delimited by periods. For example if the BusinessWorks application were named *AppName* and was deployed in the TIBCO Admin directory: aaa/bbb/ccc/ then the *{BW-APPLICATION-PATH.NAME}* would be: *aaa.bbb.ccc.AppName*

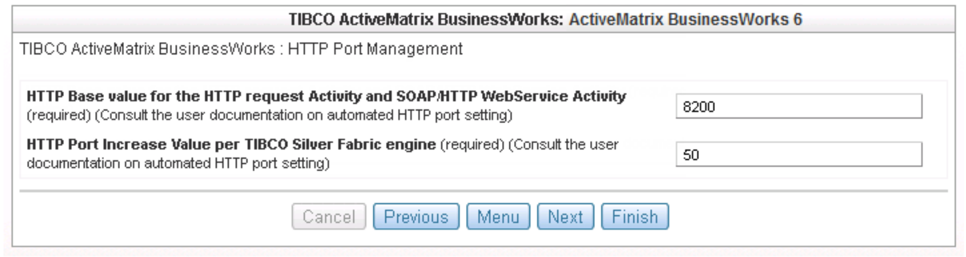
  The *{BW-APPLICATION-PATH.NAME}* may be copied from the **TIBCO Silver Fabric Administrator > Dashboard > Scaled Archives** page.
- *{BW_HttpPortName}* – This global variable, endpoint URI, is defined at archive design-time as the HTTP port created to accept requests. Using TIBCO Designer create the variable (the name is arbitrary) in the global variable group and it should be set as modifiable at the service level to allow the value to be changed at runtime.

  – If deployed from TIBCO Administrator UI or AppManage, it is the default value set in Designer,

  – If deployed from BW Enabler Component (REST, .ear file uploaded) , it is the HTTP port calculated to avoid port conflict.

  The base port value for HTTP request activity and SOAP/HTTP Web service activity may be changed at the component level as was shown previously.

  The TIBCO ActiveMatrix BusinessWorks: HTTP Port Management page



TIBCO ActiveMatrix BusinessWorks: ActiveMatrix BusinessWorks 6

TIBCO ActiveMatrix BusinessWorks : HTTP Port Management

HTTP Base value for the HTTP request Activity and SOAP/HTTP WebService Activity (required) (Consult the user documentation on automated HTTP port setting)	8200
HTTP Port Increase Value per TIBCO Silver Fabric engine (required) (Consult the user documentation on automated HTTP port setting)	50

- ***{GA-APPLICATION-PATH.NAME}*** - the deployment directory and name of the Adapter application with folders and name delimited by periods. For example, if the Adapter application were named *AppName* and was deployed in the TIBCO Admin directory `aaa/bbb/ccc/`, the ***{GA-APPLICATION-PATH.NAME}*** would be: `aaa.bbb.ccc.AppName`

  The ***{GA-APPLICATION-PATH.NAME}*** may be copied from the **TIBCO Silver Fabric Administrator** > **Dashboard** > **Scaled Archives** page.

- ***{GA_HttpPortName}*** - This global variable, endpoint URI, is defined at archive design-time as the HTTP port created to accept requests. Using TIBCO Designer create the variable (the name is arbitrary) in the global variable group and it should be set as modifiable at the service level to allow the value to be changed at run time.

  – If deployed from TIBCO Administrator UI or AppManage, it is the default value set in TIBCO Designer

  – If deployed from GA enabler component (REST, .ear file uploaded) , it is the HTTP port calculated to avoid port conflict

  The base port value for HTTP request activity and SOAP/HTTP Web service activity may be changed at the component level as shown earlier.

For example:

```
http://MyBrokerMachine(IP)Name:8080/MyDomain/{BW-APPLICATION-PATH}{SFGA-APPLICATION-
PATH}/MyArchiveName.par/HTTPVariables/HTTP_Port
```