TIBCO Silver® Fabric Enabler for TIBCO BusinessEvents® User's Guide

Software Release 3.3
June 2017

Two-Second Advantage®

TIBCO®

Important Information

Contents

Figures 6
TIBCO Documentation and Support Services 8
TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Overview 9
Main Functionalities 9
Components 9
TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Stack 10
Deployment Modes 10
BusinessEvents Deployment with TIBCO Administrator 11
Specifying the Component Name 11
Specifying the Deployment Mode: with TIBCO Administrator 11
Selecting the Software Distribution Versions 11
Selecting the Optional Software Distribution Versions 12
Setting the BusinessEvents Basic Configuration 12
Uploading a BusinessEvents Project 15
Hawk Application Management Interface (AMI) Configuration (optional) 16
HTTP Port Management (optional) 17
TIBCO Hawk Agent Running Condition Configuration 18
TIBCO Hawk Plugins File Configuration (optional) 18
Adding or Editing Runtime Context Variables (optional) 19
Uploading a Content File 20
Allocation Rule Settings 20
Edit Configuration File Screen 21
BusinessEvents Deployment in the Standalone mode 22
Specifying the Component Name 22
Specifying the Deployment Mode: Standalone 22
Selecting the Software Distribution Versions 23
Uploading a BusinessEvents Project 23
Creating a Deployment Configuration XML File 24
HTTP Port Management (optional) 24
TIBCO BusinessEvents Engine Running Condition Configuration 26
Adding or Editing Runtime Context Variables (optional) 26
Uploading a Content File 27
Allocation Rule Settings 27
Edit Configuration File Screen 28
BusinessEvents Deployment with TIBCO Enterprise Administrator 29
Specifying the Component Name 29

Specifying the Deployment Mode: with TIBCO Enterprise Administrator 30
Selecting the Software Distribution Versions 30
TIBCO Enterprise Administrator Configuration 30
Using Custom SSH Keys for BusinessEvents TIBCO Enterprise Administrator Agent 32
Using TIBCO Enterprise Administrator Component Dependency 32
Uploading a BusinessEvents Project 33
HTTP Port Management (optional) 33
TIBCO BusinessEvents Engine Running Condition Configuration 35
Adding or Editing Runtime Context Variables (optional) 35
Uploading a Content File 36
Add Allocation Rule Settings 36
Edit Configuration File Screen 37
Creating a Stack 37
Changing the Component Enabler 38
Setting the Dependency Requirements 39
Using Statistics 40
Setting Rules for an Engine 41
TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Statistics 42
Creating Archive Scaling Rules 43
Running a Stack 46
Updating the Stack 46
Continuous Deployment - Deploying Archives Directly to Endpoints 46
Continuous Deployment Life Cycle 47
Deploy 47
Mandatory Form-Data 48
Optional Form-Data 51
Creating a Variable Provider for Use by a REST Call 54
Successful Deployment 56
Continuous Deployment Timeout 56
Start 57
Stop 57
Edit Deployment 58
Import Deployment 58
Delete Deployment 58
Update JVM Property 58
Update Property 59
Update a Global Variable 59
Copy Instance 59
Edit Instance 60

Update System Property 60
Hot Deploy 61
Undeploy 61
The Archive Management Support Feature 61
Ant Scripts 62
Samples 62
build.xml 62
build.properties 68
HTTP Load Balancer for a BusinessEvents Stack 70
Retrieving Log Files 71
Retained Log Files 71

Figures

Specify BusinessEvents Deployment Mode 10
Name and Description of the Component 11
Basic Configuration Page 12
Uploading Archives 15
TIBCO Hawk AMI Configuration 16
HTTP Port Management 17
TIBCO Administrator and Hawk Agent Running Conditions 18
Upload Hawk MicroAgent Plug-in 19
Adding a Runtime Context Variable 19
Uploading Content Files 20
Using BusinessEvents Statistics for Threshold Activation 20
Edit the Configuration File Page 21
Name and Description of the Component 22
Uploading Archives 23
HTTP Port Management 25
TIBCO BusinessEvents Engine Running Condition 26
Adding a Runtime Context Variable 27
Uploading Content Files 27
Using BusinessEvents Statistics for Threshold Activation 28
Edit the Configuration File Page 28
Name and Description of the Component 30
TIBCO Enterprise Administrator Configuration 31
Configure SSH Keys 32
TIBCO Enterprise Administrator Server Settings 32
SSL Certificates 33
Uploading Archives 33
HTTP Port Management 34
TIBCO BusinessEvents Engine Running Condition 35
Adding a Runtime Context Variable 36
Uploading Content Files 36
Edit the Configuration File Page 37
Creating a Stack 38
Stack Builder Page 38
Restart the Component 39
Setting Component Dependency 40
Creating Rules 42
Creating a New Archive Scaling Rule 44
Define Archive Scaling Rule conditions 44

Running a Stack 46
Log Files 71

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 TIBCO BusinessEvents® Installation*
- *TIBCO Silver® Fabric Enabler for TIBCO BusinessEvents® User's Guide*
- *TIBCO Silver® Fabric Enabler for TIBCO BusinessEvents® Release Notes*
- *TIBCO Silver® Fabric Enabler for TIBCO BusinessEvents® Developer's 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

TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Overview

TIBCO Silver® Fabric Enabler for TIBCO BusinessEvents® is a complementary software component. You can publish TIBCO BusinessEvents projects in cloud environments based on TIBCO Silver® Fabric and leverage Silver Fabric capabilities. This accelerates the publishing of TIBCO BusinessEvents projects, enforces its industry best practices, and provides elastic optimization of computing resources.

Main Functionalities

TIBCO Silver Fabric Enabler for TIBCO BusinessEvents provides the following main functionalities:

- You can quickly set up an environment for TIBCO BusinessEvents on multiple machines. TIBCO administrator can then publish TIBCO BusinessEvents projects onto this environment using traditional tools, such as the TIBCO Administrator user interface or its command-line tools.
- You can manage and monitor multiple TIBCO products deployed in an enterprise as TIBCO BusinessEvents now integrates TIBCO Enterprise Administrator.
- You can quickly define, set up, and publish a complete stack to a set of virtual or physical machines based on TIBCO BusinessEvents projects. These actions include installation of this software, creation of TIBCO Domain, starting up TIBCO Administrator Server or starting up TIBCO Enterprise Administrator, and publishing of the TIBCO BusinessEvents projects on one or multiple machines.
- It ensures that all deployments follow a set of required and supported TIBCO practices to implement, load balancing, software updates, and stack updates.
- It collects well-known TIBCO BusinessEvents® metrics from BusinessEvents engines to be exploited in TIBCO Silver® Fabric rules. It scales up and down the number of BusinessEvents engines required to process the workload. Therefore, it provides elasticity and optimization of computing resources.
- Enables creation of a pool of TIBCO Domain Machines (TLM) for scaling up and scaling down TIBCO BusinessEvents engines, and component archives.
- Gathers and reports statistics from each TIBCO BusinessEvents® service instance published by TIBCO Silver® Fabric to support automated rule-based scaling from archive statistics.
- Supports fast TLM restart for engines from shared drives. Planned TLM restarts are streamlined so that large numbers of components with multiple archives each can be restarted quickly.

Components

TIBCO Silver Fabric Enabler for TIBCO BusinessEvents consists of the following components:

- TIBCO BusinessEvents Enabler

  It is used to configure, start, and manage TIBCO BusinessEvents engines and published component archives.
- A set of distributions that includes all the software pieces required to run a TIBCO BusinessEvents environment: TIBCO Runtime Agent, TIBCO Administrator, TIBCO Rendezvous®, TIBCO Hawk®, TIBCO BusinessEvents, Jython, TIBCO Silver® Fabric Enabler for TIBCO® Enterprise Administrator and TIBCO ActiveSpaces® Enterprise Edition.

Also, please refer *TIBCO Silver® Fabric Enabler for TIBCO BusinessEvents® Installation Guide* to know the list of required and optional products.

TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Stack

The TIBCO Silver Fabric Enabler for TIBCO BusinessEvents stack runs inside TIBCO Silver® Fabric to help with component configuration and publishing. Using it you can publish TIBCO BusinessEvents platform and TIBCO BusinessEvents projects.

A TIBCO Silver Fabric Enabler for TIBCO BusinessEvents stack can consist of a TIBCO Administrator component, and one or more TIBCO BusinessEvents components. TIBCO BusinessEvents component can be run in standalone mode without the TIBCO Administrator component.

To build and run a TIBCO Silver Fabric Enabler for TIBCO BusinessEvents stack with a TIBCO Administrator component, perform the following tasks:

- Create and publish one or more TIBCO BusinessEvents components.
- Create a TIBCO BusinessEvents stack for TIBCO Silver Fabric Enabler.

  Refer to Creating a Stack.
- Set a dependency to the TIBCO Administrator component or TIBCO Enterprise Administrator for each TIBCO BusinessEvents component.

  Refer to Setting the Dependency Requirements.
- Optionally, set rules for TIBCO Silver® Fabric engines.

  Refer to Using Statistics.

After completing these tasks, you can run and update a TIBCO Silver Fabric Enabler for TIBCO BusinessEvents stack. See page Running a Stack and Updating the Stack for information on how to run and update a stack.

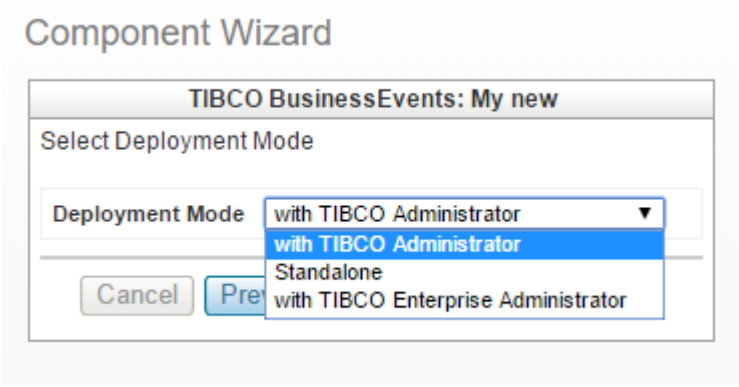
Deployment Modes

There are three deployment modes in which you can deploy your BusinessEvents components.

Select the Deployment Mode:

- with TIBCO Administrator
- Standalone
- with TIBCO Enterprise Administrator

Specify BusinessEvents Deployment Mode



BusinessEvents Deployment with TIBCO Administrator

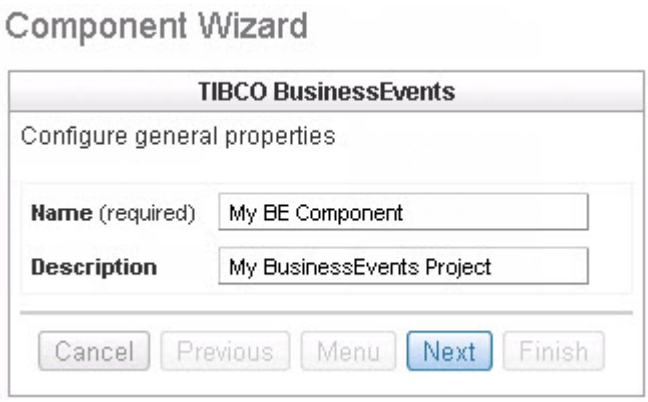
TIBCO Administrator can be used to administer, manage, and monitor BusinessEvents applications. You can use TIBCO Administrator for deploying, hot deploying, undeploying, starting and stopping TIBCO BusinessEvents engines.

Specifying the Component Name

Procedure

1. In the TIBCO Silver® Fabric Administration Tool, select **Stacks** > **Components** .
2. On the Components page, select **Create New TIBCO BusinessEvents Component** in the Global Actions list.
3. If a previous version of this enabler is installed, you can choose the enabler version you want to use for this new component using a dialog. It is good practice to choose the latest enabler version. Click **OK**.
4. Provide a name and description for the component.

   Name and Description of the Component



Specifying the Deployment Mode: with TIBCO Administrator

From the deployment mode drop down, select the deployment mode as **with TIBCO Administrator**.

TIBCO Administrator can be used to administer, manage, and monitor BusinessEvents applications. You can use TIBCO Administrator for deploying, hot deploying, undeploying, starting and stopping TIBCO BusinessEvents engines.

Selecting the Software Distribution Versions

Select the distribution version you want to publish and run.

ActiveSpaces distribution version drop-down is displayed if you choose TIBCO BusinessEvents distribution version of 5.4.1.0.0 or higher. When publishing TIBCO BusinessEvents with supporting applications, select the distribution versions for all the TIBCO products to be installed by the Silver® Fabric Broker.

By default, the latest versions of the distributions are displayed. All versions of the distributions are compatible.

Selecting the Optional Software Distribution Versions

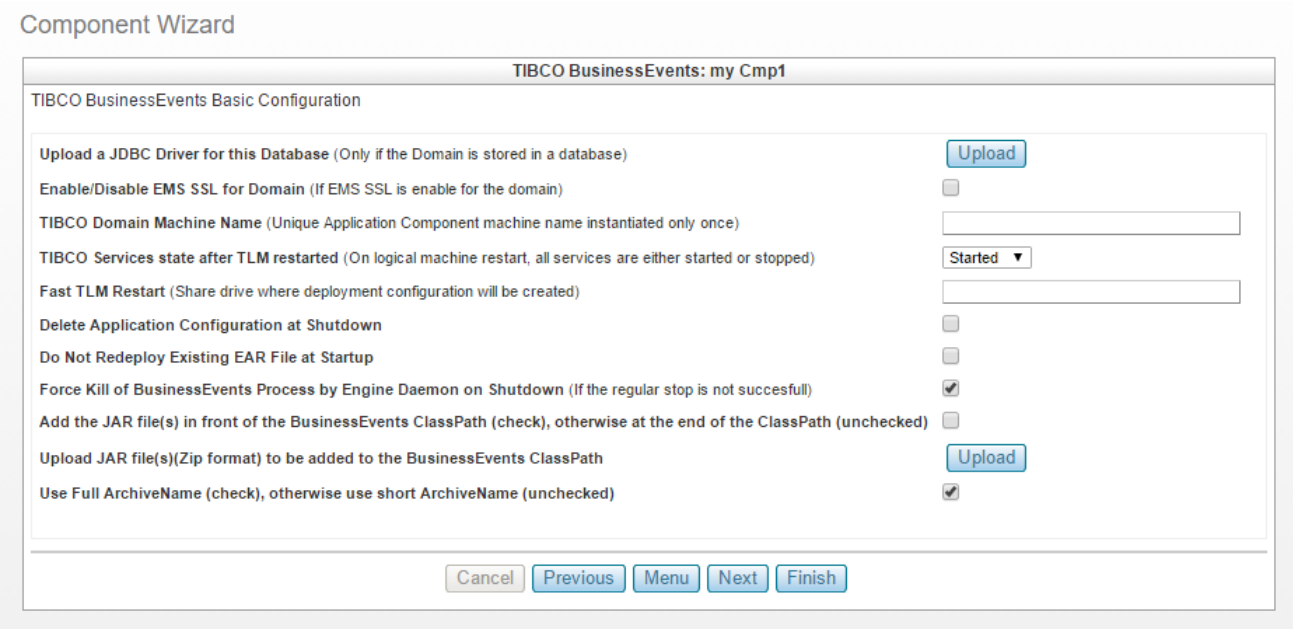
When using TIBCO Runtime Agent 5.9 (or later versions) and using TIBCO Enterprise Messaging Service as the transport, choose the Enterprise Messaging Service distribution in the optional distribution page.

Setting the BusinessEvents Basic Configuration

Using the TIBCO BusinessEvents Basic Configuration page you can define the JDBC driver. You can set a specific TIBCO Domain machine name (the TIBCO Logical Machine name - TLM). You can provide an option to specify the Fast TLM Restart, and uploading of external jar files. You can also prepend/append the path in the ClassPath, and can specify full or short archive names.

Each of the settings are described in the order of their appearance on the page.

Basic Configuration Page



Procedure

1. **Upload a JDBC driver (optional)**

   Specify the JDBC driver to publish with the TIBCO BusinessEvents Distribution 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.

   Your TIBCO BusinessEvents component might have dependency on TIBCO Administrator. Refer to Setting the Dependency Requirements for instructions on how to set that dependency. Disregard setting the dependency if your TIBCO BusinessEvents component is being created to run as a standalone instance.

   Refer to the TIBCO Administrator documentation for more information.

2. **Enable/Disable EMS SSL for Domain (optional)**

   Select this check box to enable EMS SSL for the domain

3. **Set the Domain Machine Name (optional)**

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 any reason.

You can also create a pool of logical machines with specific names that are ready for micro-scaling, scaling up or down BusinessEvents engines according to archive scaling rules defined on tracked statistics.

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.

TIBCO Domain machine name: To activate use of the TIBCO Domain Machine name enter a value in the **TIBCO Domain Machine Name** field. Configuration of the TIBCO Domain Machine Name is optional. The domain machine name 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.

When <*ComponentInstanceNumber*> equals to 0, then TLM machine name is <*Your_TLM_Name*>.

If the **TIBCO Domain Machine Name** field is left blank the component name is used for setting the domain machine name of multiple instantiations.

4. **Set the TIBCO Services state after TLM is restarted**

   TIBCO Services state after TLM restarted: sets the desired services state when the TIBCO Logical Machine is restarted. When the TLM is restarted TIBCO BusinessEvents 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 that are not required to be started for every logical machine change.

5. **Set the Fast TLM Restart shared drive.**

   Fast TIBCO Logical Machine Restart provides for accelerated restart of the engine and redeploying of many archives within reduced time. You can enhanced restart times 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.

   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.
   - Domain configuration changes made in the period between the TLM stop and the TLM restart are not captured.

   Fast TLM Restart: to enable Fast TIBCO Logical Machine Restart, enter the directory path of the shared NFS drive and make sure that the host grants permissions enabling the user who launches the engines to read and write in that location.

   If domain configuration changes made for the period after TLM stop and before TLM restart must be captured, the user must redeploy the modified application.

   For the implementations that use lesser than ten BusinessEvents deployments per component, avoid using Fast TLM Restart to avoid the constraints mentioned previously.

   If you want to force redeployment (synchronization with the domain) once, add a file call 'ForceRedeploy.txt' in:

   ```
   domainDataHome(<domainDataDir>/<DomainName>/<TLMNAME>/<DomainName>
   ```

It redeploys all applications, but no FAST TLM at the startup and then deletes the file. It is a one time action.

6. **Delete Application Configuration at Shutdown**

   Select this option to undeploy and remove all the applications deployed on the

   TIBCO Logical Machine at component shutdown.

   If this check box is selected, enabler deletes all the applications that are deployed on the administrator when the component is restarted. The EAR application attached with component is deployed again, and all the previous data is lost. So if you select this check box and deploy applications through REST call, the applications are deleted and all the data is lost after the component is restarted.

   Use this feature with extreme caution. Do not use this feature if you are not familiar with it.

7. **Do not Redeploy Existing EAR File at Startup**

   Select this option to avoid redeployment of the EAR file whenever the TIBCO Logical Machine restarts.

   You must select this check box after you remove EAR applications from the component, and if you do not want these applications to run again.

   Use this feature with extreme caution. Do not use this feature if you are not familiar with it.

8. **Force Kill of BusinessEvents Process by Engine Daemon at Shutdown**

   Select this option to forcefully kill the BusinessEvents processes at shutdown. Even though BusinessEvents engine is always stopped at shutdown, in case Hawk is down, BusinessEvents engine stops abruptly resulting in orphan engines. To avoid such orphan engines, the engine daemon kills beengine forcefully.

9. **Upload JAR files (in compressed format) to be added to the BusinessEvents ClassPath.**

   Upload individual JAR files to be either appended or prepended to the ClassPath. All uploaded JARs are added 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.

10. **Use the full (or the short) ArchiveName.**

    The ArchiveName is used for archive scaling. You can choose either Full ArchiveNames or Short ArchiveNames depending on your implementation of archives in your stacks.

    Use the full ArchiveName by selecting the box if you reuse archives more than once.

    Use the short ArchiveName, leaving the box cleared, if you do not use the same archives in different applications. The short ArchiveName makes a more convenient name to type, but you cannot reuse the same archive in different TIBCO Administrator applications using the shortened name.

    Within the same stack, full ArchiveNames and short ArchiveNames cannot be mixed.

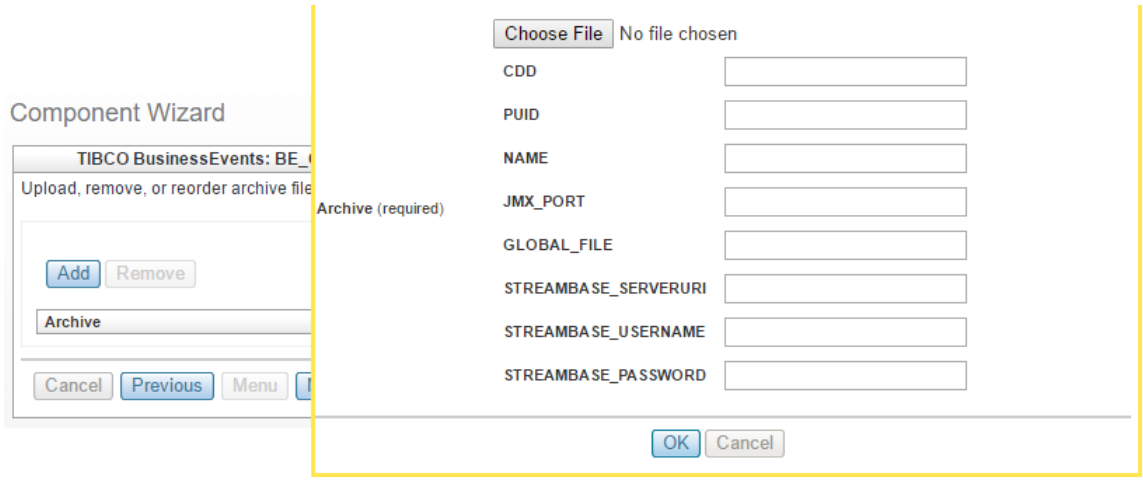
Uploading a BusinessEvents Project

Upload BusinessEvents archives to publish and run BusinessEvents projects. You can upload Enterprise archive (EAR) or files (.zip files) one at a time.

Procedure

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

   Uploading Archives



2. Click the **Browse** button in the Upload A File panel to navigate to the EAR or .zip file
   a) Specify a CDD (Cluster Deployment Descriptor), PUID (Processing Unit ID) file specification as might be required. Also specify NAME, JMX_PORT, GLOBAL_FILE , STREAMBASE_SERVERURL, STREAMBASE_USERNAME, and STREAMBASE_PASSWORD. TIBCO BusinessEvents projects published by TIBCO Silver Fabric support the use of a Cluster Deployment Descriptor (CDD) XML file and a processing unit ID to configure a project EAR for publishing. The value of CDD is the name and relative directory path of that *.cdd file within your compressed archive. The value of PUID is what you set in the *.cdd file or "*default*".

   When CDD and PUID values are entered into the component configuration UI, they overwrite any CDD and PUID values that might have been defined in a deployment configuration XML file is described as follows.

   Also, while uploading the BusinessEvents project archives, parameters such as the name, JMX port parameters and global file can be set.

   b) Click **OK** to proceed.

   You can upload one of two types of archive files:

   - EAR File

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

   - .zip File

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

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

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`".

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 as a substitution to the global variables at the deployment.

When the deployment path is not specified, the uploaded BusinessEvents project EAR and application files can be found in the Application Management directory at the root folder level of TIBCO Administrator.

When the EAR application is deployed to the BusinessEvents run time 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, the uploaded BusinessEvents project EAR and application files could be found in the TIBCO Administrator directory:
`Silver Fabric/<BEComponentName>/<TLM_Name>/<AppName>`

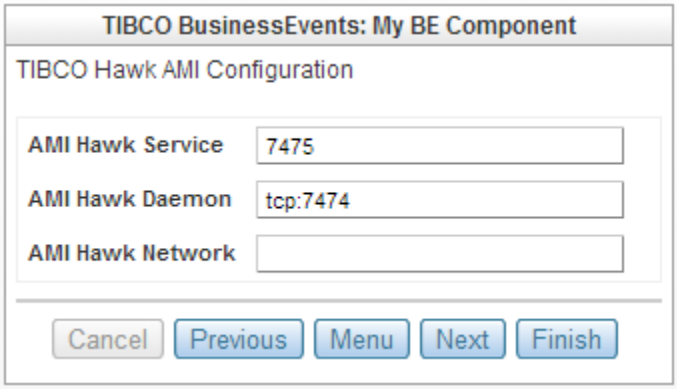
For details about the creation of the *DeploymentConfig*`.xml` file, refer to the *TIBCO Runtime Agent* documentation, *Scripting Deployment User's Guide*.

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

Hawk Application Management Interface (AMI) Configuration (optional)

TIBCO Hawk AMI can be used to monitor BusinessEvents deployments.

TIBCO Hawk AMI Configuration



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, results in an error.

AMI Hawk Daemon

Specifies the location of the TIBCO Hawk Daemon. A value of "tcp:*yyyy*" corresponds to a local Hawk Daemon where "*yyyy*" is the port number. A Hawk Daemon located elsewhere is 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 might be the same or they might be different. By default, different TLMs on the same engine daemon (physical machine) are using the same RV transport (default AMI Hawk Service=7475, and default AMI Hawk Daemon=tcp:7474). This setting enables visibility of 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, the service is reported in the hawkagent.cfg file as 6465. When the component is scaled up to other engine instances the AMI Hawk Service value is incremented higher by the enabler automatically.

Generally, AMI Hawk Network is an empty string.

HTTP Port Management (optional)

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

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

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

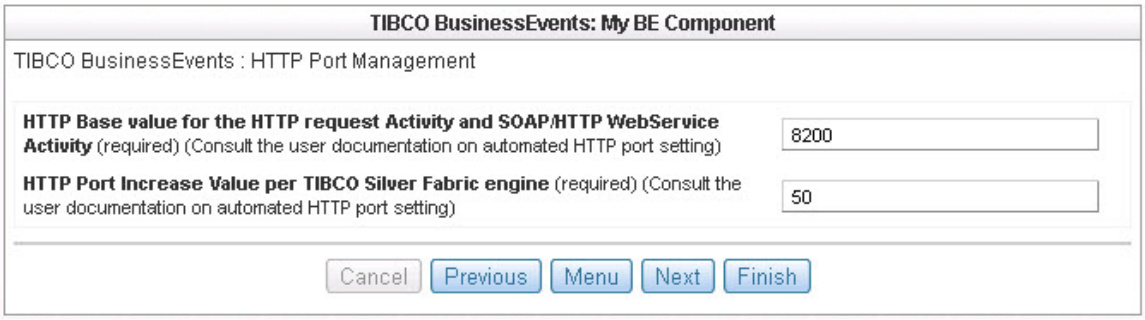
This defines a reserved range of ports that should not be used elsewhere.

If the application that is to be scaled includes an HTTP Port Variable, you must set the HTTP Port Variable to be settable at the service level using TIBCO BusinessEvents® Studio. To make the HTTP Port Variable "service settable" select the **Service Settable** check box in the Global Variable Editor.

Refer to the section on "Working with Global Variables" in the *TIBCO BusinessEvents Developers Guide* for more information.

Edit the base values of the "HTTP Base Value..." and "HTTP Port Increase Value..." variables on the HTTP Port Management page, as shown in the HTTP Port Management figure.

HTTP Port Management



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 BusinessEvents project.

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

For example, you have a BusinessEvents 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 BusinessEvents components running on the same Silver® Fabric private cloud. If the values are different, you might encounter port conflicts.

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

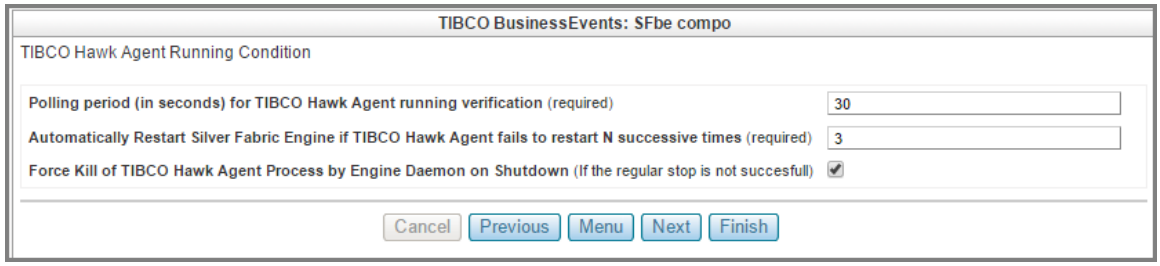
If you choose to deploy TIBCO BusinessEvents as a standalone component on Silver Fabric engines, ensure that you have set ports be sure to avoid conflicts. In standalone mode, global variables such as those for ports (that is port1=*1234* and port2=*5678*) can be set in a file ending in *.cfg suffix in the uploaded compressed archive file.

For more information, refer to the *TIBCO BusinessEvents Administration Guide* - "Building and Deploying EAR Files at the Command Line" : Starting a TIBCO BusinessEvents Engine at the Command Line.

TIBCO Hawk Agent Running Condition Configuration

In the TIBCO Hawk Running condition window, enter values in each field.

TIBCO Administrator and Hawk Agent Running Conditions



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, the process is automatically restarted.

The running condition check might be disabled with a value of 0.

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 you restart the TIBCO Silver Fabric engine enabler. A successful restart resets the count.

The automatic restart might be disabled with a value of 0.

Force kill of TIBCO Hawk agent process by engine daemon on shutdown (If the regular stop is not successful)

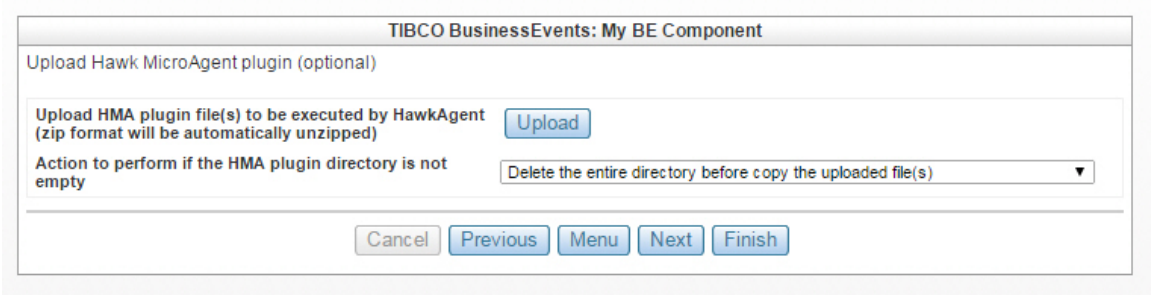
Use this option, only if the regular stop does not work. Select this option if you want to kill all the TIBCO Hawk Agent processes on shutdown. It stops all those processes which were started by the engine daemon.

TIBCO Hawk Plugins File Configuration (optional)

The Hawk Agent can be configured to execute specific Hawk MicroAgent (HMA) plug-ins uploaded in the BusinessEvents enabler component.

Normally, Hawk Agent executes all HMA plug-ins located in the directory defined by the property hma_plugin_dir in the file $DomainData/tra/<domain>/HawkAgent.cfg. Because the default value ($DomainData/tra/<domain>/plugin) cannot be changed by configuration, you can upload plugins to the engine work directory, which are then copied to this directory.

Upload Hawk MicroAgent Plug-in



Upload HMA plugin file to be executed by HawkAgent

Click **Upload** and choose a file to upload. The file uploaded is located in the $EngineWorDir/fabric/hmaplugin directory. It is copied to the hma_plugin_dir between the AddMachineAction() and StartHawkAgentAction() lifecycle events.

Action to perform if the HMA plugin directory is not empty

Choose between "**Delete the entire directory before copy the uploaded file(s)**" or "**Keep all existing files and replace the existing file(s) with the upload one(s)**".

Adding or Editing Runtime Context Variables (optional)

String, **Environment**, **System**, or **Encrypted** variables might be added to the component to define and set runtime specific context variables.

You can add a string variable to change the preexisting context variables such as: `ARCHIVE_DETECTION_FREQUENCY`.

`ARCHIVE_DETECTION_FREQUENCY` is the periodic interval (the 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.

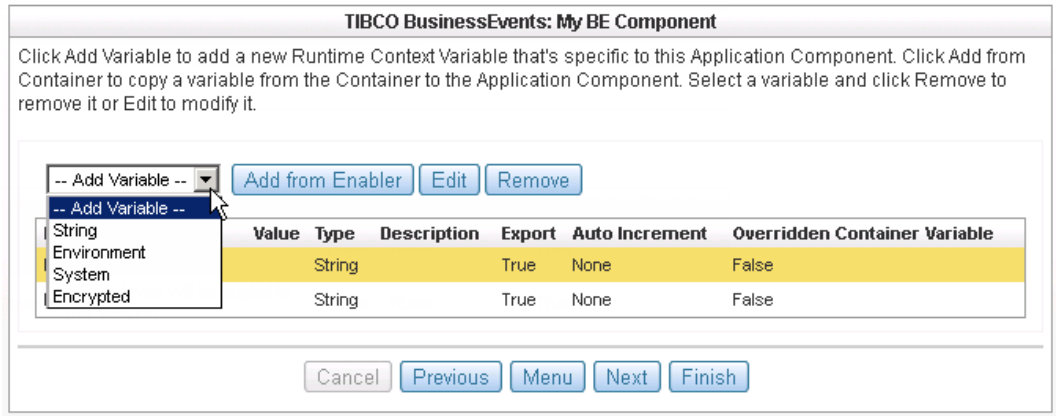
These variables are not exposed in the interface, but you can change their values. In this case, ensure that you set the variable values higher than the time needed to deploy and start an archive.

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

Procedure

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

   Adding a Runtime Context Variable



Variable values from an enabler might be added to the run time as well. Use the **Add from Enabler** button to add enabler-specific context variables.

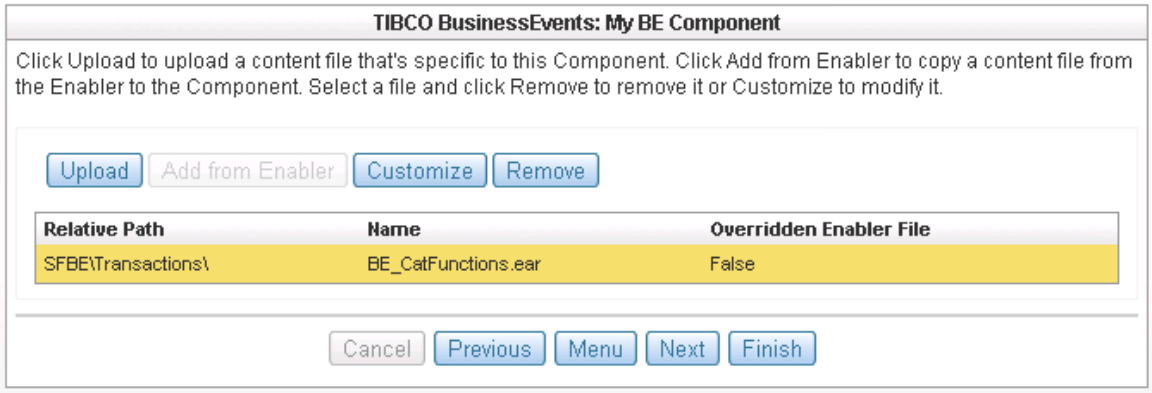
2. After you have added a runtime context variable you can select the variable (selected row is highlighted) and click **Edit** to change its attributes. Selected rows can also be removed.

Uploading a Content File

Content files can be uploaded, added from an enabler, edited with a simple text editor, or removed using the Add/override/customize Enabler and component-specific content files page.

Add files associated with the TIBCO BusinessEvents® component that might be required for the component to be run according to design. An enterprise archive (EAR) file might be uploaded to a specific relative path for runtime component use.

Uploading Content Files

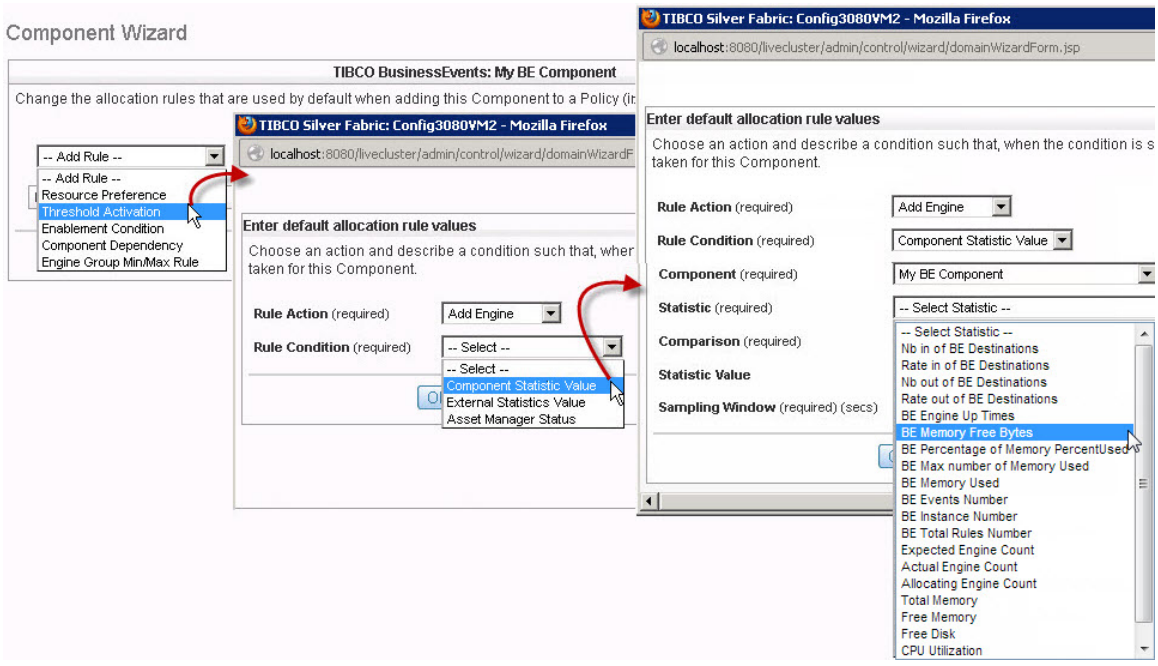


Allocation Rule Settings

Each rule selection brings up a slightly different dialog window. You can select a property of a tracked engine or evaluate the component archive statistic according to a logical operator and a value you specify to define an action using the dialog window.

In some cases the component archive statistics can be selected as shown in the following example.

Using BusinessEvents Statistics for Threshold Activation



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.

Edit Configuration File Screen

Use the Edit the Configuration File page with extreme caution. Do not use the page unless the `configuration.xml` is backed up and specific knowledge about the TIBCO Silver Fabric system is being applied. This interface can be used for more advanced customizations and normally it should be left alone.

For more information, refer "The configure.xml File" section in *TIBCO Silver® Fabric Developer's Guide*.

WARNING! Changes to the configuration.xml can break the installation. Before making any changes to the configuration.xml back it up and secure it. Consult an expert to ensure that any distribution changes are properly made. More information on the use of the `configuration.xml` can be found in the TIBCO Silver Fabric documentation.

Edit the Configuration File Page

As an example, if you want to change the default Java heap size of the JVM in file `hawkagenttra.template` from 256M to 1024M.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
  <containerConfig>
    <configFiles baseDir="${TIBCO_HOME}/tra/5.7/template/domainutility"
include="hawkagenttra.template">
    <regex pattern="java\.heap\.size\.max\s+[0-9]+[a-zA-Z]
+"replacement="java.heap.size.max 1024M" />
    </configFiles>
  </containerConfig>
```

The property, `baseDir`, in the *<configFiles>* element is used to specify the path that includes the file to be updated. It can be modified if needed. For example, if the TIBCO Runtime Administrator (TRA) version was 5.8 instead of 5.7, then the baseDir value is *${TIBCO_HOME}*`/tra/5.8/template/domainutility`.

The property, include, in `<configFiles>` element is used to specify the files to be replaced. It can specify whatever files you want to change. The asterisk wild card is used to represent a string of characters such as: "*.*tra*" to change all of the `.tra` files in *%baseDir%*.

The property, pattern, in the `<regex>` element is used to specify the contents that are to be replaced within the previously specified files. The value of pattern can be a regular expression.

The property, replacement, in `<regex>` element is used to specify the new contents of the node specified by the pattern property value.

The remaining screens of the component wizard are generic for TIBCO Silver® Fabric Enablers. These final configurations are optional for TIBCO BusinessEvents components.

Refer to TIBCO Silver® Fabric User's Guide for more information on the configuration screens.

Click **Finish** to save your changes and then you can publish the component as part of a stack. To do this, select **Publish Component** in the **Actions** drop down list located at the line of the component you just created.

BusinessEvents Deployment in the Standalone mode

TIBCO BusinessEvents can run independently on an engine to maximize use of computing resources.

Depending on whether your BusinessEvents implementation requires connections with external instances of TRA, Hawk, or Rendezvous, you must define JMX Port connections or set Cluster Deployment Descriptor file values, or manually change properties files.

Refer to the TIBCO BusinessEvents documentation for details.

Specifying the Component Name

Procedure

1. In the TIBCO Silver® Fabric Administration Tool, select **Stacks** > **Components** .
2. On the Components page, select **Create New TIBCO BusinessEvents Component** in the Global Actions list.
3. If a previous version of this enabler is installed, you can choose the enabler version you want to use for this new component using a dialog. It is good practice to choose the latest enabler version. Click **OK**.
4. Provide a name and description for the component.

   Name and Description of the Component

Specifying the Deployment Mode: Standalone

To run BusinessEvents as a standalone component, from the deployment mode drop down, select the deployment mode as **Standalone**.

If you configure your BusinessEvents component to be published in standalone mode, the TIBCO Silver Fabric Enabler for TIBCO BusinessEvents component wizard does not display those configuration pages and feature fields that are intended for use with the other TIBCO product distributions. Some of the component wizard pages described in this documentation do not apply to the standalone case and they can be ignored.

Selecting the Software Distribution Versions

Select the distribution version you want to publish and run.

ActiveSpaces distribution version drop-down is displayed if you choose TIBCO BusinessEvents distribution version of 5.4.1.0.0 or higher. When publishing TIBCO BusinessEvents with supporting applications, select the distribution versions for all the TIBCO products to be installed by the Silver® Fabric Broker.

By default, the latest versions of the distributions are displayed. All versions of the distributions are compatible.

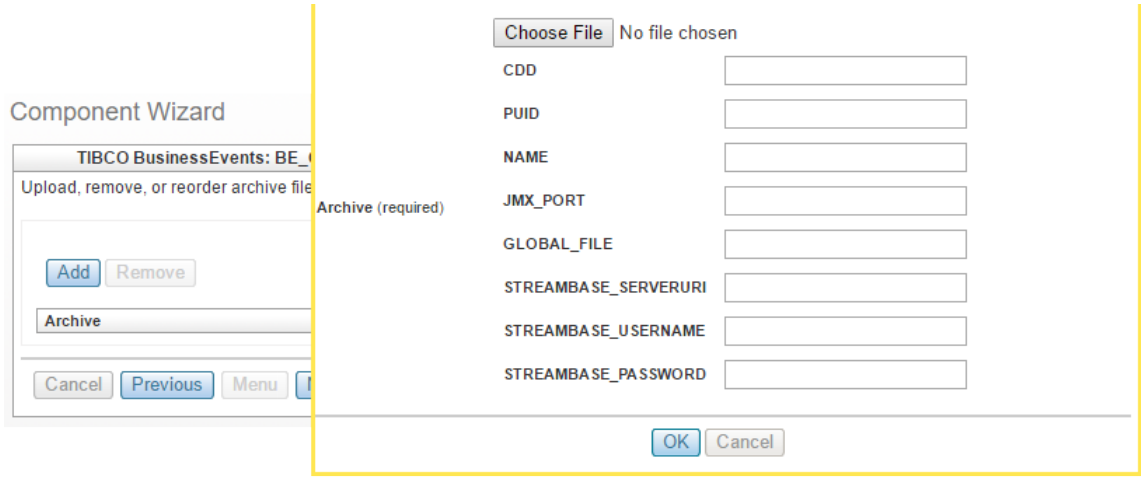
Uploading a BusinessEvents Project

Upload BusinessEvents archives to publish and run BusinessEvents projects. You can upload Enterprise archive (EAR) or files (.zip files) one at a time.

Procedure

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

   Uploading Archives



2. Click the **Browse** button in the Upload A File panel to navigate to the EAR or .zip file
   a) Specify a CDD (Cluster Deployment Descriptor), PUID (Processing Unit ID) file specification as might be required. Also specify NAME, JMX_PORT, GLOBAL_FILE , STREAMBASE_SERVERURL, STREAMBASE_USERNAME, and STREAMBASE_PASSWORD. TIBCO BusinessEvents projects published by TIBCO Silver Fabric support the use of a Cluster Deployment Descriptor (CDD) XML file and a processing unit ID to configure a project EAR for publishing. The value of CDD is the name and relative directory path of that *.cdd file within your compressed archive. The value of PUID is what you set in the *.cdd file or "*default*".

   When CDD and PUID values are entered into the component configuration UI, they overwrite any CDD and PUID values that might have been defined in a deployment configuration XML file is described as follows.

   Also, while uploading the BusinessEvents project archives, parameters such as the name, JMX port parameters and global file can be set.

   b) Click **OK** to proceed.

   You can upload one of two types of archive files:

   - EAR File

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

- .zip File

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

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

  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`".

  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 as a substitution to the global variables at the deployment.

  When the deployment path is not specified, the uploaded BusinessEvents project EAR and application files can be found in the Application Management directory at the root folder level of TIBCO Administrator.

  When the EAR application is deployed to the BusinessEvents run time 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, the uploaded BusinessEvents project EAR and application files could be found in the TIBCO Administrator directory:
  `Silver Fabric/<BEComponentName>/<TLM_Name>/<AppName>`

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 by 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 /tra/*tra_version*/bin, run the following command:

   AppManage –export –ear *EarFile*.ear -out *DeploymentConfig*`.xml`
2. Edit the file and set the values you want for deployment.
3. Create a .zip file with the file *EarFile*.ear and *DeploymentConfig*.xml.

HTTP Port Management (optional)

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

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

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

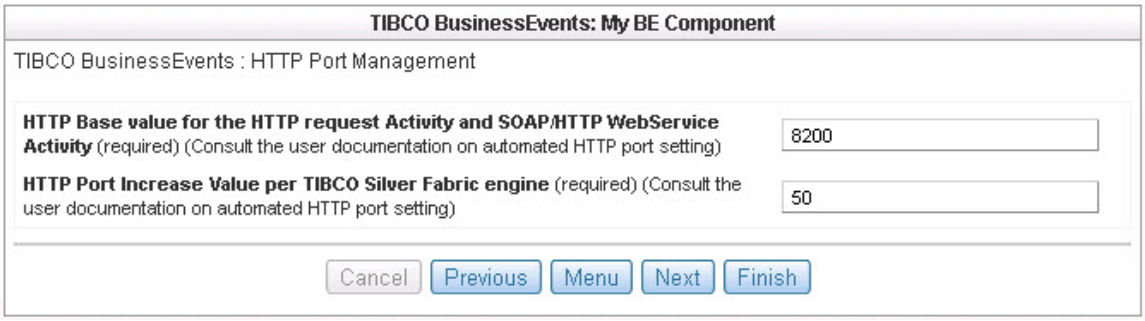
This defines a reserved range of ports that should not be used elsewhere.

If the application that is to be scaled includes an HTTP Port Variable, you must set the HTTP Port Variable to be settable at the service level using TIBCO BusinessEvents® Studio. To make the HTTP Port Variable "service settable" select the **Service Settable** check box in the Global Variable Editor.

Refer to the section on "Working with Global Variables" in the *TIBCO BusinessEvents Developers Guide* for more information.

Edit the base values of the "HTTP Base Value..." and "HTTP Port Increase Value..." variables on the HTTP Port Management page, as shown in the HTTP Port Management figure.

HTTP Port Management



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 BusinessEvents project.

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

For example, you have a BusinessEvents 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 BusinessEvents components running on the same Silver® Fabric private cloud. If the values are different, you might encounter port conflicts.

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

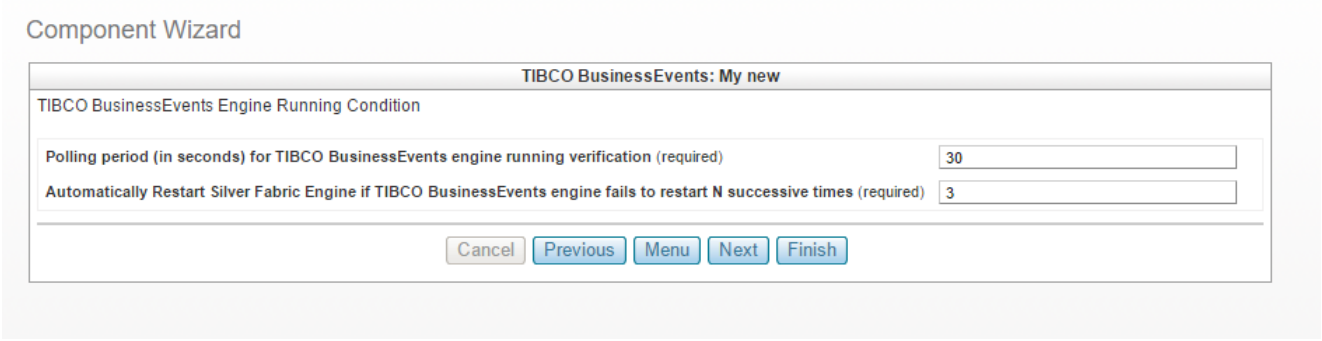
If you choose to deploy TIBCO BusinessEvents as a standalone component on Silver Fabric engines, ensure that you have set ports be sure to avoid conflicts. In standalone mode, global variables such as those for ports (that is port1=*1234* and port2=*5678*) can be set in a file ending in *.cfg suffix in the uploaded compressed archive file.

For more information, refer to the *TIBCO BusinessEvents Administration Guide* - "Building and Deploying EAR Files at the Command Line" : Starting a TIBCO BusinessEvents Engine at the Command Line.

TIBCO BusinessEvents Engine Running Condition Configuration

In the TIBCO BusinessEvents Engine Running condition window, enter values in each field.

TIBCO BusinessEvents Engine Running Condition



Polling Period (in seconds) for TIBCO BusinessEvents Engine Running Condition (required)

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

The running condition check might be disabled with a value of 0.

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

Enter an integer to specify the number of restart retries for the TIBCO BusinessEvents Engine, before you restart the TIBCO Silver Fabric engine enabler. A successful restart resets the count.

The automatic restart might be disabled with a value of 0.

Adding or Editing Runtime Context Variables (optional)

String, **Environment**, **System**, or **Encrypted** variables might be added to the component to define and set runtime specific context variables.

You can add a string variable to change the preexisting context variables such as: `ARCHIVE_DETECTION_FREQUENCY`.

`ARCHIVE_DETECTION_FREQUENCY` is the periodic interval (the 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.

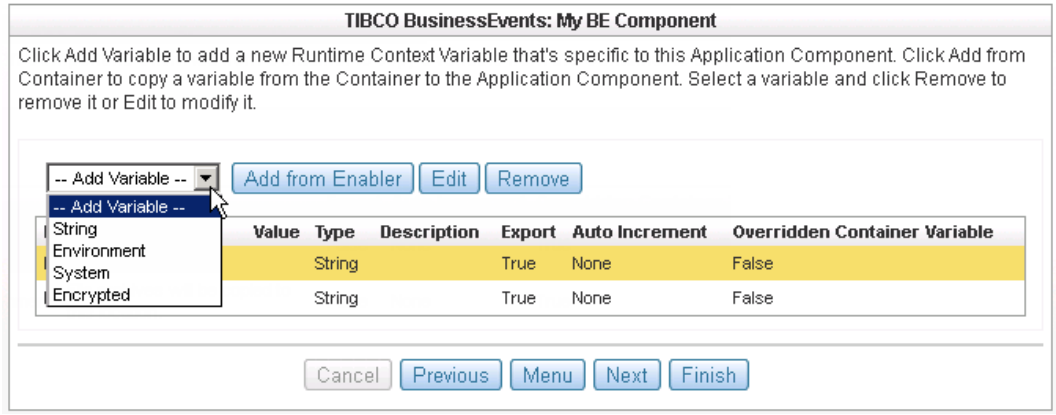
These variables are not exposed in the interface, but you can change their values. In this case, ensure that you set the variable values higher than the time needed to deploy and start an archive.

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

Procedure

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

Adding a Runtime Context Variable



Variable values from an enabler might be added to the run time as well. Use the **Add from Enabler** button to add enabler-specific context variables.

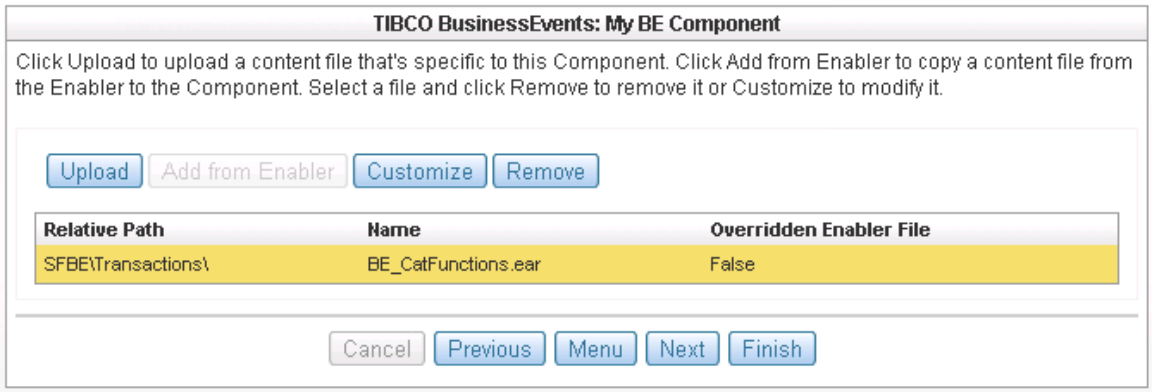
2. After you have added a runtime context variable you can select the variable (selected row is highlighted) and click **Edit** to change its attributes. Selected rows can also be removed.

Uploading a Content File

Content files can be uploaded, added from an enabler, edited with a simple text editor, or removed using the Add/override/customize Enabler and component-specific content files page.

Add files associated with the TIBCO BusinessEvents® component that might be required for the component to be run according to design. An enterprise archive (EAR) file might be uploaded to a specific relative path for runtime component use.

Uploading Content Files

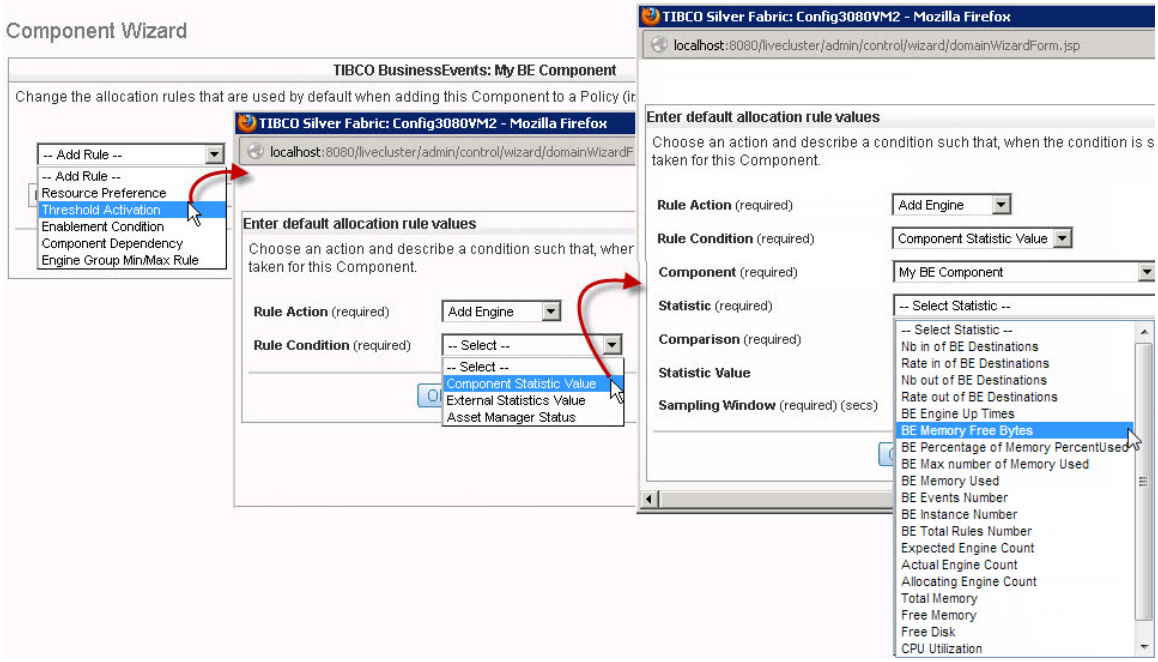


Allocation Rule Settings

Each rule selection brings up a slightly different dialog window. You can select a property of a tracked engine or evaluate the component archive statistic according to a logical operator and a value you specify to define an action using the dialog window.

In some cases the component archive statistics can be selected as shown in the following example.

Using BusinessEvents Statistics for Threshold Activation



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.

Edit Configuration File Screen

Use the Edit the Configuration File page with extreme caution. Do not use the page unless the `configuration.xml` is backed up and specific knowledge about the TIBCO Silver Fabric system is being applied. This interface can be used for more advanced customizations and normally it should be left alone.

For more information, refer "The configure.xml File" section in *TIBCO Silver® Fabric Developer's Guide*.

WARNING! Changes to the configuration.xml can break the installation. Before making any changes to the configuration.xml back it up and secure it. Consult an expert to ensure that any distribution changes are properly made. More information on the use of the `configuration.xml` can be found in the TIBCO Silver Fabric documentation.

Edit the Configuration File Page

As an example, if you want to change the default Java heap size of the JVM in file `hawkagenttra.template` from 256M to 1024M.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
  <containerConfig>
```

```
    <configFiles baseDir="${TIBCO_HOME}/tra/5.7/template/domainutility"
include="hawkagenttra.template">
    <regex pattern="java\.heap\.size\.max\s+[0-9]+[a-zA-Z]
+"replacement="java.heap.size.max 1024M" />
    </configFiles>
  </containerConfig>
```

The property, `baseDir`, in the *<configFiles>* element is used to specify the path that includes the file to be updated. It can be modified if needed. For example, if the TIBCO Runtime Administrator (TRA) version was 5.8 instead of 5.7, then the baseDir value is *${TIBCO_HOME}*`/tra/5.8/template/domainutility`.

The property, include, in `<configFiles>` element is used to specify the files to be replaced. It can specify whatever files you want to change. The asterisk wild card is used to represent a string of characters such as: "*.*tra*" to change all of the `.tra` files in *%baseDir%*.

The property, pattern, in the `<regex>` element is used to specify the contents that are to be replaced within the previously specified files. The value of pattern can be a regular expression.

The property, replacement, in `<regex>` element is used to specify the new contents of the node specified by the pattern property value.

The remaining screens of the component wizard are generic for TIBCO Silver® Fabric Enablers. These final configurations are optional for TIBCO BusinessEvents components.

Refer to TIBCO Silver® Fabric User's Guide for more information on the configuration screens.

Click **Finish** to save your changes and then you can publish the component as part of a stack. To do this, select **Publish Component** in the **Actions** drop down list located at the line of the component you just created.

BusinessEvents Deployment with TIBCO Enterprise Administrator

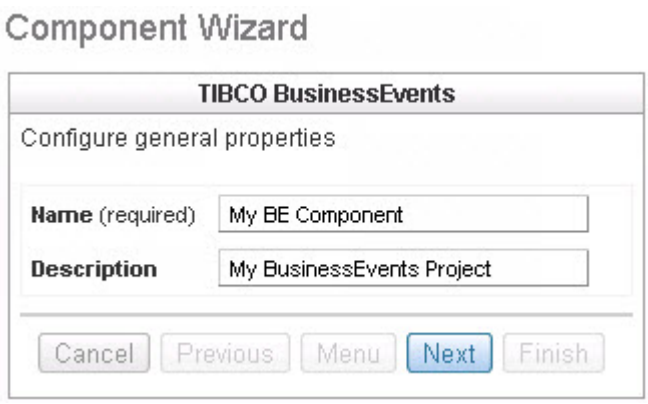
TIBCO Enterprise Administrator server provides an Administrator UI which can be used to create, view, and monitor runtime entities. See section TIBCO Enterprise Administrator Configuration for more details.

Specifying the Component Name

Procedure

1. In the TIBCO Silver® Fabric Administration Tool, select **Stacks** > **Components** .
2. On the Components page, select **Create New TIBCO BusinessEvents Component** in the Global Actions list.
3. If a previous version of this enabler is installed, you can choose the enabler version you want to use for this new component using a dialog. It is good practice to choose the latest enabler version. Click **OK**.
4. Provide a name and description for the component.

Name and Description of the Component



Specifying the Deployment Mode: with TIBCO Enterprise Administrator

From the deployment mode drop down, select the deployment mode as **with TIBCO Enterprise Administrator**.

Selecting the Software Distribution Versions

Select the distribution version you want to publish and run.

ActiveSpaces distribution version drop-down is displayed if you choose TIBCO BusinessEvents distribution version of 5.4.1.0.0 or higher. When publishing TIBCO BusinessEvents with supporting applications, select the distribution versions for all the TIBCO products to be installed by the Silver® Fabric Broker.

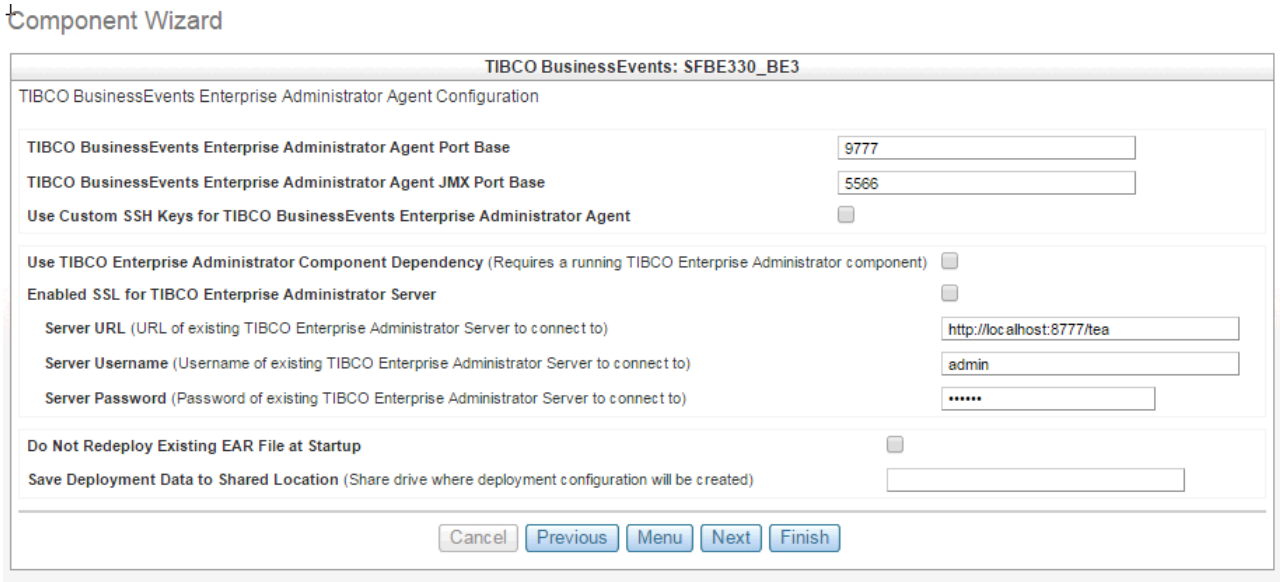
By default, the latest versions of the distributions are displayed. All versions of the distributions are compatible.

TIBCO Enterprise Administrator Configuration

If you configure your component to be deployed using the deployment mode **with TIBCO Enterprise Administrator**, you need to specify the TIBCO Enterprise Administrator configuration fields.

The following screen is displayed when the deployment mode is **with TIBCO Enterprise Administrator**

TIBCO Enterprise Administrator Configuration



- **TIBCO BusinessEvents Enterprise Administrator Agent Port Base** Specifies TIBCO BusinessEvents Enterprise Administrator Agent listening port

  The default value is 9777.

- **TIBCO BusinessEvents Enterprise Administrator Agent JMX Port Base** Specifies the port for JMX connection.

  The default value is 5566.

- **Use custom SSH Keys for BusinessEvents Enterprise Administrator Agent** Select this option to use custom SSH keys for the TIBCO Enterprise Administrator agent.

  To configure the SSH keys for BusinessEvents TIBCO Enterprise Administrator Agent, you need to upload the **Private SSH Key** and provide the **SSH Key Passphrase**.

- **Enabled SSL for TIBCO Enterprise Administrator Server** option is provided if your TIBCO Enterprise Administrator server is running on SSL. To configure this check **Enabled SSL for TIBCO Enterprise Administrator Server** option and clear the **Use TIBCO Enterprise Administrator Component Dependency** check box. Provide the TIBCO Enterprise Administrator server details such as Server URL, Server Username and Server Password. If you select the **Enabled SSL for TIBCO Enterprise Administrator Server** check box, then provide the Server and Client certificates for authentication on the next screen that is displayed.

- **Do not Redeploy Existing EAR File at Startup**

  Select this option to avoid redeployment of the EAR file whenever the BusinessEvents component restarts.

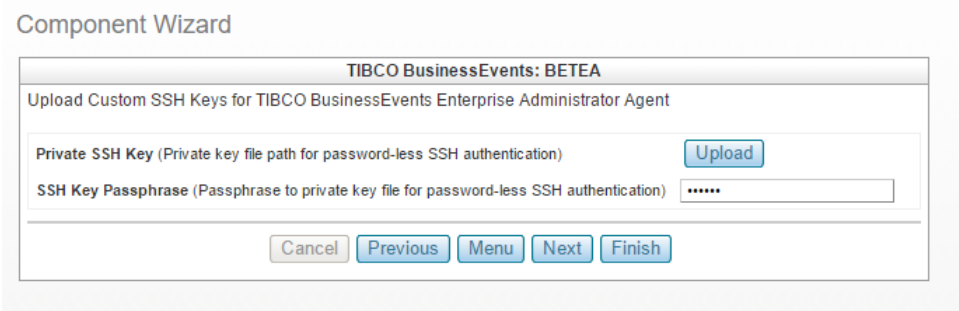
  ⚠ You must select this check box after you remove EAR applications from the component, and if you do not want these applications to run again.

  Use this feature with extreme caution. Do not use this feature if you are not familiar with it.

- **Save Deployment Data to Shared Location** Mention the share drive the deployment configuration is to be created.

Using Custom SSH Keys for BusinessEvents TIBCO Enterprise Administrator Agent

Configure SSH Keys

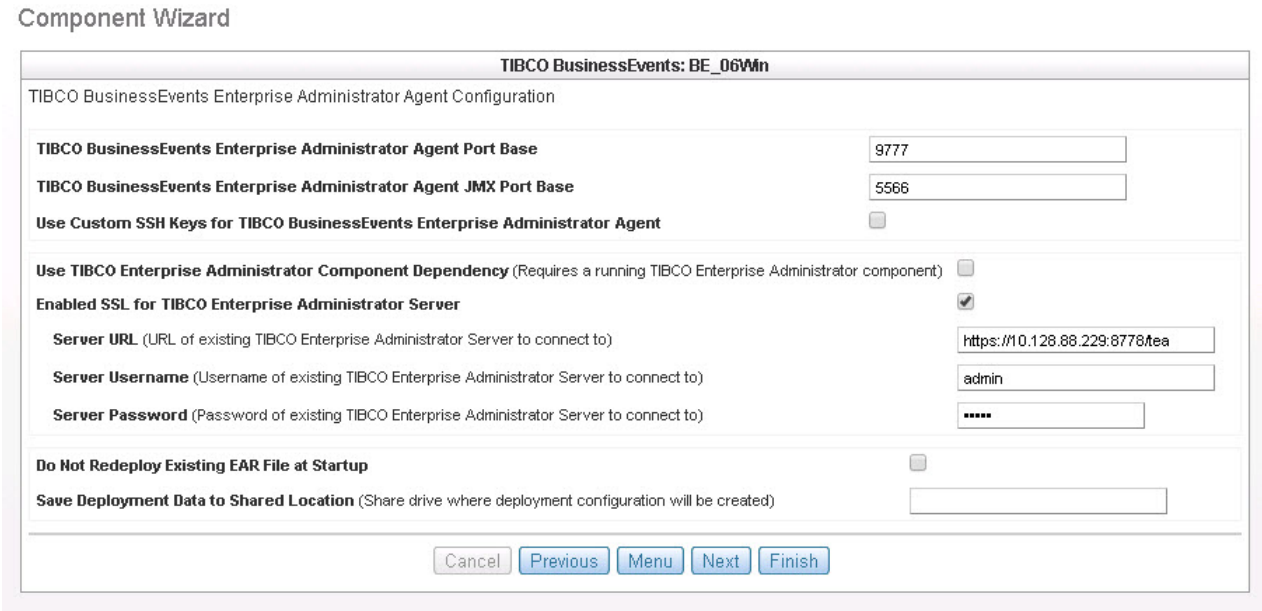


Using TIBCO Enterprise Administrator Component Dependency

The **Use TIBCO Enterprise Administrator Component Dependency** check box is selected by default. If you clear the check box, (when no TIBCO Enterprise Administrator component is used), provide the TIBCO Enterprise Administrator Server settings such as **Server URL**, **Server Username** and **Server Password**.

The **Enabled SSL for TIBCO Enterprise Administrator Server** option is provided if your TIBCO Enterprise Administrator server is running on SSL. To configure this check **Enabled SSL for TIBCO Enterprise Administrator Server** option and clear the **Use TIBCO Enterprise Administrator Component Dependency** check box. Provide the TIBCO Enterprise Administrator server details such as Server URL, Server Username and Server Password.

TIBCO Enterprise Administrator Server Settings



If you select the **Enabled SSL for TIBCO Enterprise Administrator Server** check box, provide the Server and Client certificates for authentication on the next screen that is displayed.

SSL Certificates

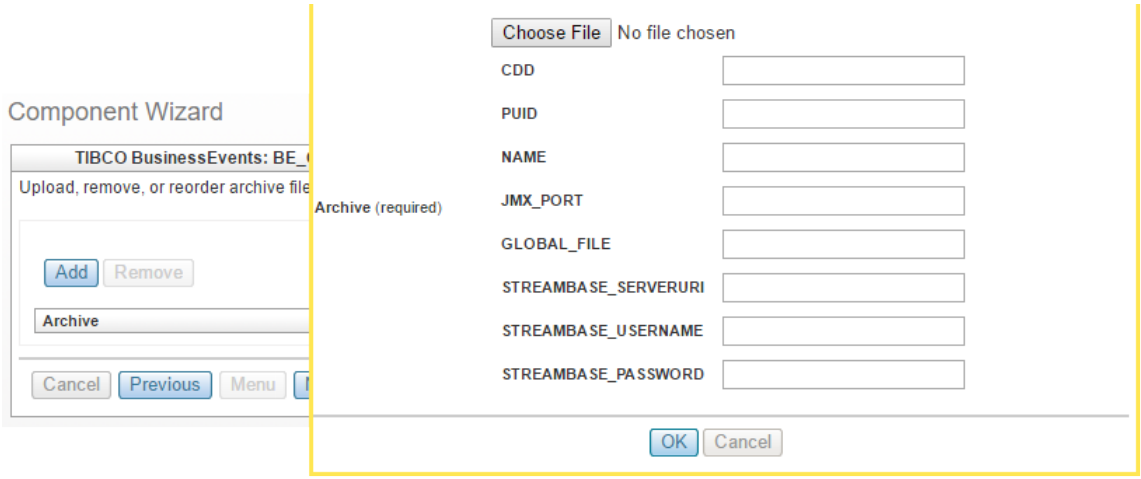
Uploading a BusinessEvents Project

Upload BusinessEvents archives to publish and run BusinessEvents projects. You can upload Enterprise archive (EAR) or files (.zip files) one at a time.

Procedure

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

   Uploading Archives



2. Click the **Browse** button in the "Upload A File" panel to navigate to the EAR or .zip file
   a) Specify a CDD (Cluster Deployment Descriptor), PUID (Processing Unit ID) file specification as might be required. Also specify NAME, JMX_PORT, GLOBAL_FILE, STREAMBASE_SERVERURL, STREAMBASE_USERNAME, and STREAMBASE_PASSWORD. (If you are are not using StreamBase, the last three fields must remain blank). TIBCO BusinessEvents projects published by TIBCO Silver Fabric support the use of a Cluster Deployment Descriptor (CDD) XML file and a processing unit ID to configure a project EAR for publishing. The value of CDD is the name and relative directory path of that *.cdd file within your zipped archive. The value of PUID is what you set in the *.cdd file or "*default*".

   Also, while uploading the BusinessEvents project archives, parameters such as the name, JMX port parameters and global file can be set.

   b) Click **OK** to proceed.

HTTP Port Management (optional)

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

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

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

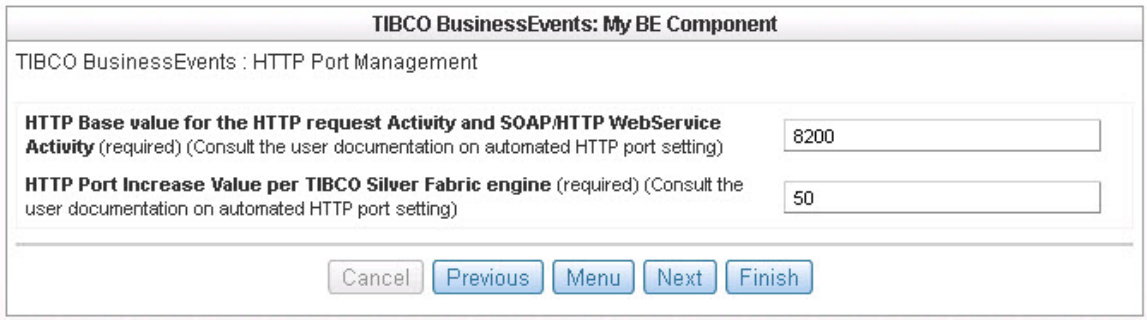
This defines a reserved range of ports that should not be used elsewhere.

If the application that is to be scaled includes an HTTP Port Variable, you must set the HTTP Port Variable to be settable at the service level using TIBCO BusinessEvents® Studio. To make the HTTP Port Variable "service settable" select the **Service Settable** check box in the Global Variable Editor.

Refer to the section on "Working with Global Variables" in the *TIBCO BusinessEvents Developers Guide* for more information.

Edit the base values of the "HTTP Base Value..." and "HTTP Port Increase Value..." variables on the HTTP Port Management page, as shown in the HTTP Port Management figure.

HTTP Port Management



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 BusinessEvents project.

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

For example, you have a BusinessEvents 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 BusinessEvents components running on the same Silver® Fabric private cloud. If the values are different, you might encounter port conflicts.

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

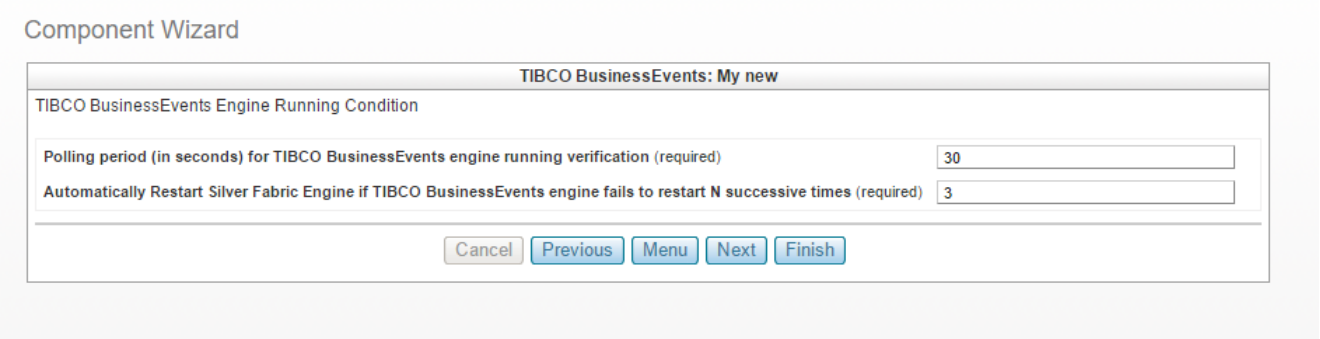
If you choose to deploy TIBCO BusinessEvents as a standalone component on Silver Fabric engines, ensure that you have set ports be sure to avoid conflicts. In standalone mode, global variables such as those for ports (that is port1=*1234* and port2=*5678*) can be set in a file ending in *.cfg suffix in the uploaded compressed archive file.

For more information, refer to the *TIBCO BusinessEvents Administration Guide* - "Building and Deploying EAR Files at the Command Line" : Starting a TIBCO BusinessEvents Engine at the Command Line.

TIBCO BusinessEvents Engine Running Condition Configuration

In the TIBCO BusinessEvents Engine Running condition window, enter values in each field.

TIBCO BusinessEvents Engine Running Condition



Polling Period (in seconds) for TIBCO BusinessEvents Engine Running Condition (required)

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

The running condition check might be disabled with a value of 0.

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

Enter an integer to specify the number of restart retries for the TIBCO BusinessEvents Engine, before you restart the TIBCO Silver Fabric engine enabler. A successful restart resets the count.

The automatic restart might be disabled with a value of 0.

Adding or Editing Runtime Context Variables (optional)

String, **Environment**, **System**, or **Encrypted** variables might be added to the component to define and set runtime specific context variables.

You can add a string variable to change the preexisting context variables such as: `ARCHIVE_DETECTION_FREQUENCY`.

`ARCHIVE_DETECTION_FREQUENCY` is the periodic interval (the 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.

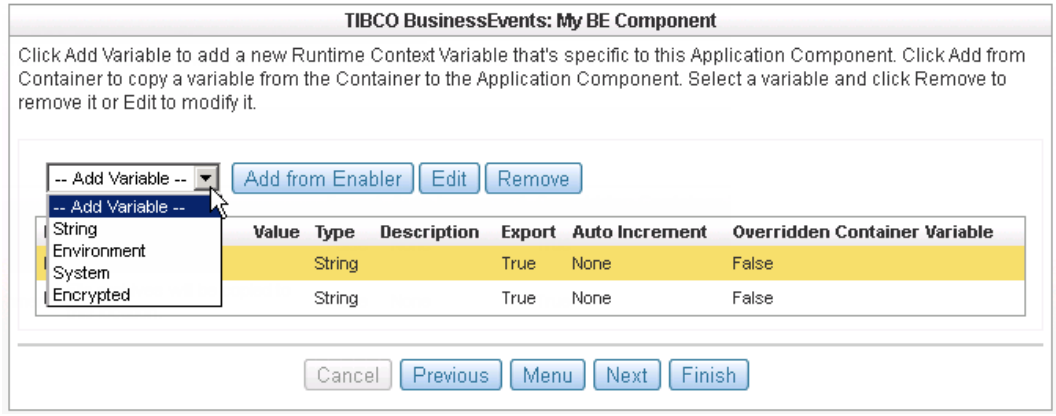
These variables are not exposed in the interface, but you can change their values. In this case, ensure that you set the variable values higher than the time needed to deploy and start an archive.

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

Procedure

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

Adding a Runtime Context Variable



Variable values from an enabler might be added to the run time as well. Use the **Add from Enabler** button to add enabler-specific context variables.

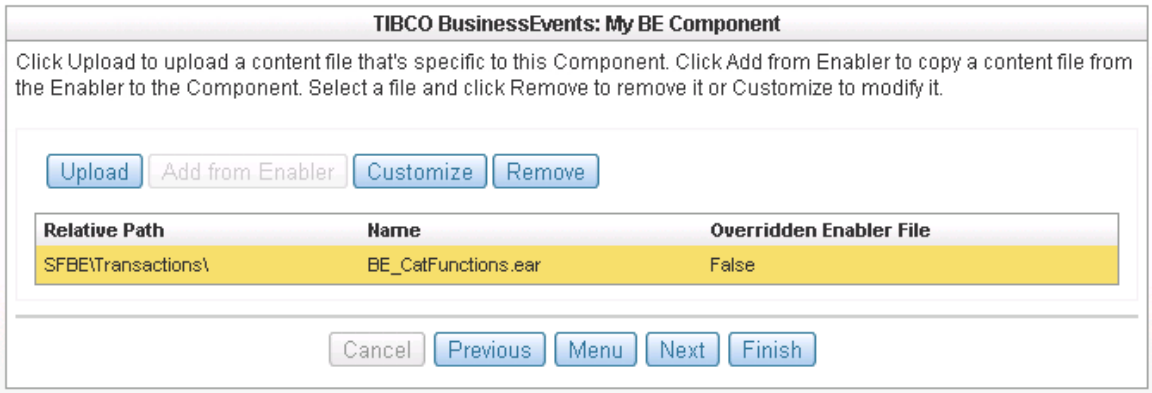
2. After you have added a runtime context variable you can select the variable (selected row is highlighted) and click **Edit** to change its attributes. Selected rows can also be removed.

Uploading a Content File

Content files can be uploaded, added from an enabler, edited with a simple text editor, or removed using the Add/override/customize Enabler and component-specific content files page.

Add files associated with the TIBCO BusinessEvents® component that might be required for the component to be run according to design. An enterprise archive (EAR) file might be uploaded to a specific relative path for runtime component use.

Uploading Content Files



Add Allocation Rule Settings

Add rules to specify and set the behavior of the component.

Add rules to do the following:

- Specify Resource Preferences
- Set Thresholds for Activation
- Set Enablement Conditions
- Specify Component Dependency or
- Set Engine Group Minimums or Maximums

Each rule selection brings up a slightly different dialog window. You can select a property of a tracked engine or evaluate the component archive statistic according to a logical operator and a value you specify to define an action using the dialog window. Refer Allocation Rule Settings for more details.

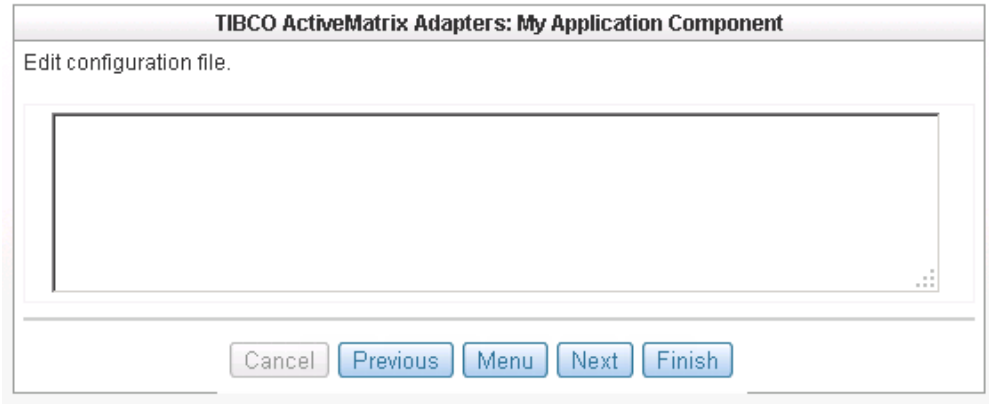
Edit Configuration File Screen

Use the Edit the Configuration File page with extreme caution. Do not use the page unless the `configuration.xml` is backed up and specific knowledge about the TIBCO Silver Fabric system is being applied. This interface can be used for more advanced customizations and normally it should be left alone.

For more information, refer "The configure.xml File" section in *TIBCO Silver® Fabric Developer's Guide*.

⚠ WARNING! Changes to the configuration.xml can break the installation. Before making any changes to the configuration.xml back it up and secure it. Consult an expert to ensure that any distribution changes are properly made. More information on the use of the `configuration.xml` can be found in the TIBCO Silver Fabric documentation.

Edit the Configuration File Page



The remaining screens of the component wizard are generic for TIBCO Silver® Fabric Enablers. These final configurations are optional for TIBCO BusinessEvents components.

Refer to TIBCO Silver® Fabric User's Guide for more information on the configuration screens.

Click **Finish** to save your changes and then you can publish the component as part of a stack. To do this, select **Publish Component** in the **Actions** drop down list located at the line of the component you just created.

Creating a Stack

Components are deployed within a stack. A stack for TIBCO Silver Fabric Enabler for TIBCO BusinessEvents normally contains a single TIBCO Administrator component or TIBCO Enterprise Administrator component, and one or more TIBCO BusinessEvents components. A stack can be created with a single component and it could be run in stand alone configurations. After creating and publishing the TIBCO Administrator or TIBCO Enterprise Administrator, and BusinessEvents components, you can create a complete stack for TIBCO Silver Fabric Enabler for TIBCO BusinessEvents.

After initially defining a stack, you can still update it by adding or removing TIBCO BusinessEvents components.

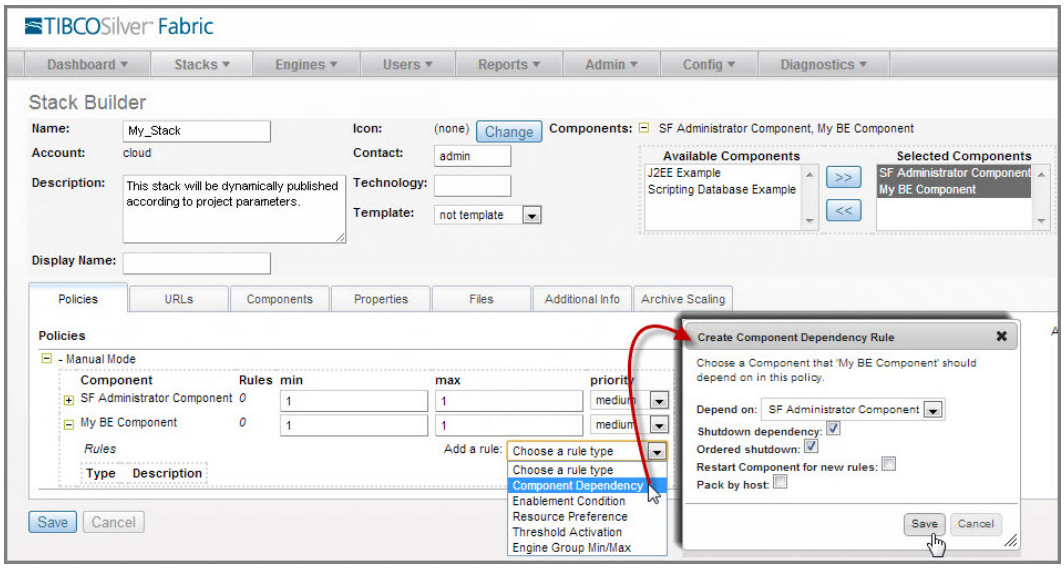
Procedure

1. In the TIBCO Silver® Fabric Administration Tool, select **Stacks** > **Stacks** .
2. Click **Create New Stack**.

Creating a Stack

3. Enter a name for the stack in the Stack Builder page.
4. In the Components area, add one TIBCO Administrator component and one or more TIBCO BusinessEvents components when you use BusinessEvents Deployment **with TIBCO Administrator** mode, or add one TIBCO Enterprise Administrator component and one or more BusinessEvents component, when you use BusinessEvents deployment **with TIBCO Enterprise Administrator** mode. Other components (EMS, and so on) must be added to the stack according to your implementation.
5. In the Policies area, expand the component you just added to view the details of the component.

Stack Builder Page



Changing the Component Enabler

Upgrading a component created with TIBCO Silver Fabric Enabler for TIBCO BusinessEvents release to this latest version is easy.

⚠ Back up your engines and consider when you want to perform a component restart, so brief operational downtime has minimal impact.

Procedure

1. Using the TIBCO Silver Fabric **Administrator Components** page, identify the enabler for the BusinessEvents component that you wish to upgrade from release 2.5.x or 3.1.x. The **Enabler Version** column helps to identify out-of-date enablers.

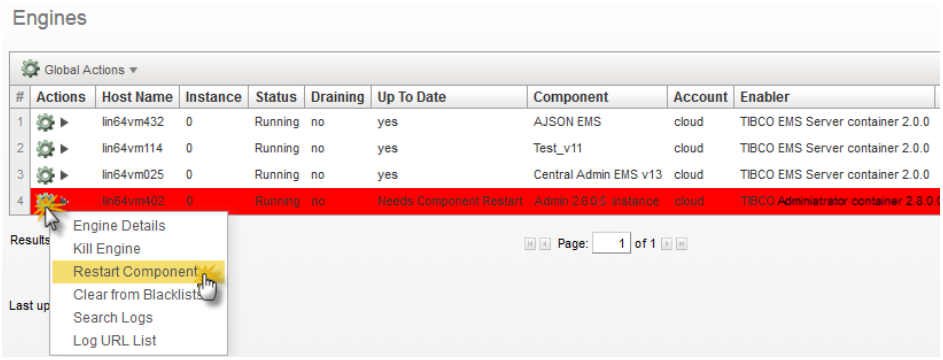
2. Click the **Component** Actions menu icon on the row for the component you want to update and choose **Change Enabler**. Select the enabler version upgrade target and click **OK**.

   You cannot downgrade a component.

3. Click the component **Actions** menu icon again, click **Publish Changes**, and then click **OK** to publish the selected component.
4. Switch to the **Engines** page and those stacks that used the upgraded component is displayed in red if they require a component restart.
5. Click the **Actions** menu icon that corresponds to "**Needs a Component Restart**", in the **Up to Date** column and then click **Restart Component**.

   Restart the Component



Setting the Dependency Requirements

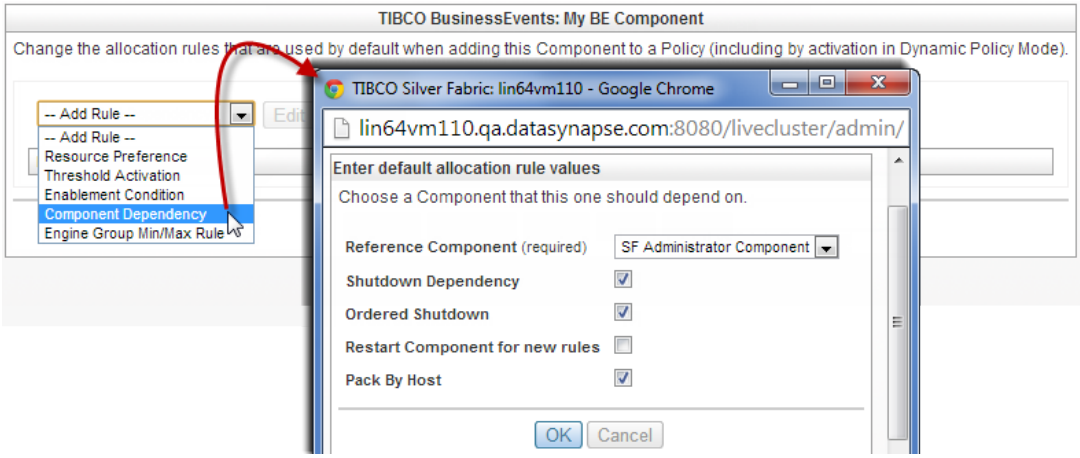
Each TIBCO BusinessEvents component, unless its used in standalone mode, has component dependency set on one TIBCO Administrator component or TIBCO Enterprise Administrator component. You can set this component dependency for each of your TIBCO BusinessEvents components. Those components without defined component dependencies do not have enough information for proper publishing.

The TIBCO BusinessEvents component can have the TIBCO Administrator or TIBCO Enterprise Administrator component configuration information so that it can publish, unpublish, and communicate with other components (if required) successfully. A connection is supported between the TIBCO BusinessEvents component and one instance of the TIBCO Administrator component or TIBCO Enterprise Administrator component. Connecting with more than one TIBCO Administrator component or TIBCO Enterprise Administrator component is not supported. After setting the dependency, TIBCO BusinessEvents starts after TIBCO Administrator or TIBCO Enterprise Administrator starts running.

Procedure

1. During creation or during edit of the TIBCO BusinessEvents component, select "**Add/edit default rule settings**" from the menu of the component wizard.
2. Use the **Add Rule** pull-down list to select the **Component Dependency** option.

Setting Component Dependency



3. From the **Reference Component** drop-down list, select the name of the TIBCO Administrato or TIBCO Enterprise Administrator component that runs inside your stack.

Shutdown Dependency

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

Ordered Shutdown

It 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, the order of the 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 want to manually restart components later to propagate changes, leave this box cleared.

If the administrator component was configured to Use dependent EMS server then that dependency must be set here as well.

When using a dependent TIBCO Enterprise Message Service™ server, the dependency should be set in the TIBCO Administrator component, which must also have a dependency on the TIBCO EMS Server component.

Pack by Host

Select this to specify that dependent components must run on the same host.

For more information on these settings refer, *TIBCO Silver Fabric User's Guide*

Using Statistics

If you want TIBCO BusinessEvents components to scale automatically (adding or removing engines), you can define rules that add or remove TIBCO Silver® Fabric engines based on engine statistics or BusinessEvents component statistics.

Scaling using statistics is not supported when TIBCO Silver Fabric publishes the TIBCO BusinessEvents components to a single TIBCO Domain Machine.

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 each engine), and then averaging the results across all engines.

For example, if a specific aggregated value triggers the rule, but the normalized geometric variance across the engines is less than 0.85, it does not add an engine. Removing engines is not affected by variance.

When an engine is added, it automatically publishes the BusinessEvents archive that was configured with the enabler component on a new engine. For HTTP activities, for example, for web services using SOAP over HTTP transport and HTTP receiver activities, the engine adds the URL to the load balancer automatically.

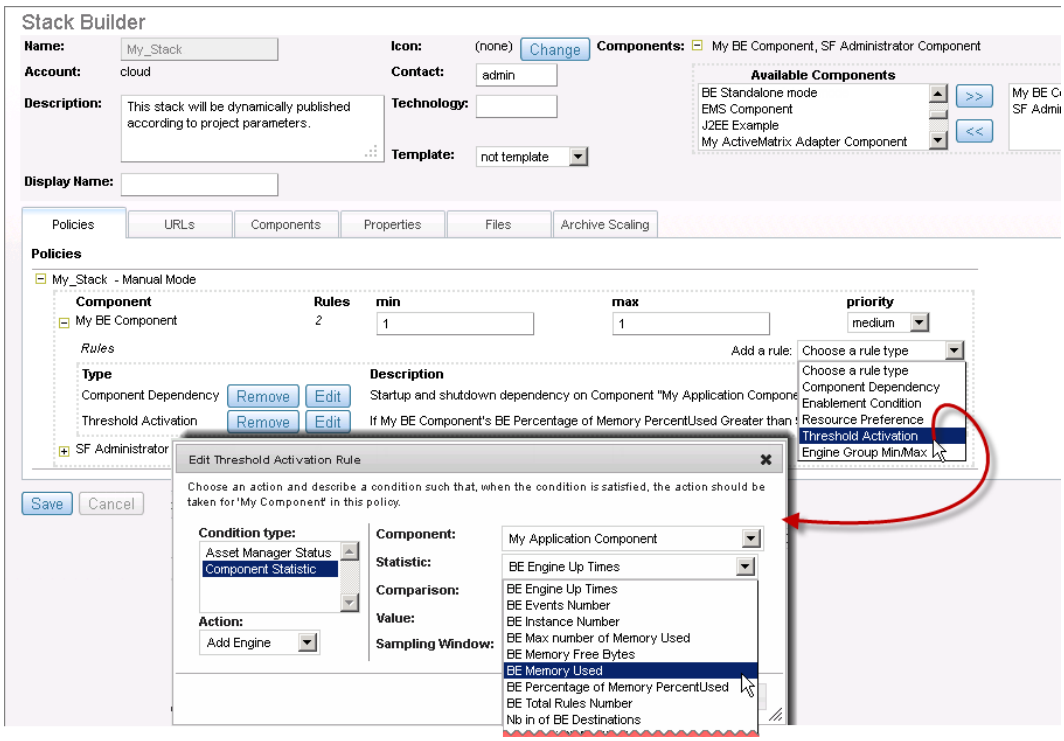
You can set up rules on enablement condition. The engine starts upon statistics rules on other engines. You also can set up rules on threshold activation, which is the statistic on the engine itself or other engines.

Setting Rules for an Engine

Procedure

1. In the **Policies** area of the TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Stack Builder page, select the component for which you want to set up rules.
2. Select **Threshold Activation** or **Enablement Condition** present in the A**dd A Rule** list.
3. If you select **Threshold Activation**, specify the following parameters in the Create Threshold Activation Rule panel:
   - From the **Condition Type** list, select the **Component Statistic** item.
   - From the **Action list**, select **Add Engine** or **Remove Engine**.
   - From the **Component** list, select the component where the statistic rules apply.
   - From the **Statistics** list, select the base property for the activation.
   - From the **Comparison** list, select an operator such as: **Greater Than** or **Less Than**.
   - **Value** field, set the value of the measure that serves as the threshold or defining line that triggers the action when criteria are met.
   - **Sampling Window** field, set the time interval (in seconds). It specifies how often the statistics are evaluated against the criteria defined to trigger the action selected.

Creating Rules



TIBCO Silver Fabric Enabler for TIBCO BusinessEvents Statistics

Engine statistics consist of statistic information about the engine, machine (independent of BusinessEvents components), and statistic information gathered from the BusinessEvents engine through TIBCO Hawk®.

The following table lists the available BusinessEvents statistics.

TIBCO Silver Fabric Enabler for TIBCO BusinessEvents (BE) Statistics

Name	Description
Nb in of BE Archive Destinations	The Number in of destinations of BE
Rate in of BE Archive Destinations	The Rate in of destinations of BE
Nb out of BE Archive Destinations	The Number out of destinations of BE
Rate out of BE Archive Destinations	The Rate out of destinations of BE
BE Archive Engine Up Times	The up time of the engine running this BE Archive instance
BE Archive Memory Free Bytes	The total number of bytes that are not currently in use
BE Archive Percentage of Memory Percent Used	Percentage of memory used (Used/Allocated)
BE Archive Max number of Memory Used	The max number of bytes that are currently in use

Name	Description
BE Archive Memory Used	The total number of bytes that are currently in use
BE Archive Events Number	The total number of events
BE Archive Instance Number	The total number of instances
BE Archive Total Rules Number	The total number of rules fired
Nb in of BE Destinations	The number in of destinations of BE
Rate in of BE Destinations	The rate in of destinations of BE
Nb out of BE Destinations	The number out of destinations of BE
Rate out of BE Destinations	The rate out of destinations of BE
BE Engine Up Times	The up time of the engine running this BE instance
BE Memory Free Bytes	Total number of bytes that are not currently in use
BE Percentage of Memory Percent Used	Percentage of memory used (Used/Allocated)
BE Max number of Memory Used	The max number of bytes that are currently in use
BE Memory Used	The total number of bytes that are currently in use
BE Events Number	The total number of events
BE Instance Number	The total number of instances
BE Total Rules Number	The total number of rules fired

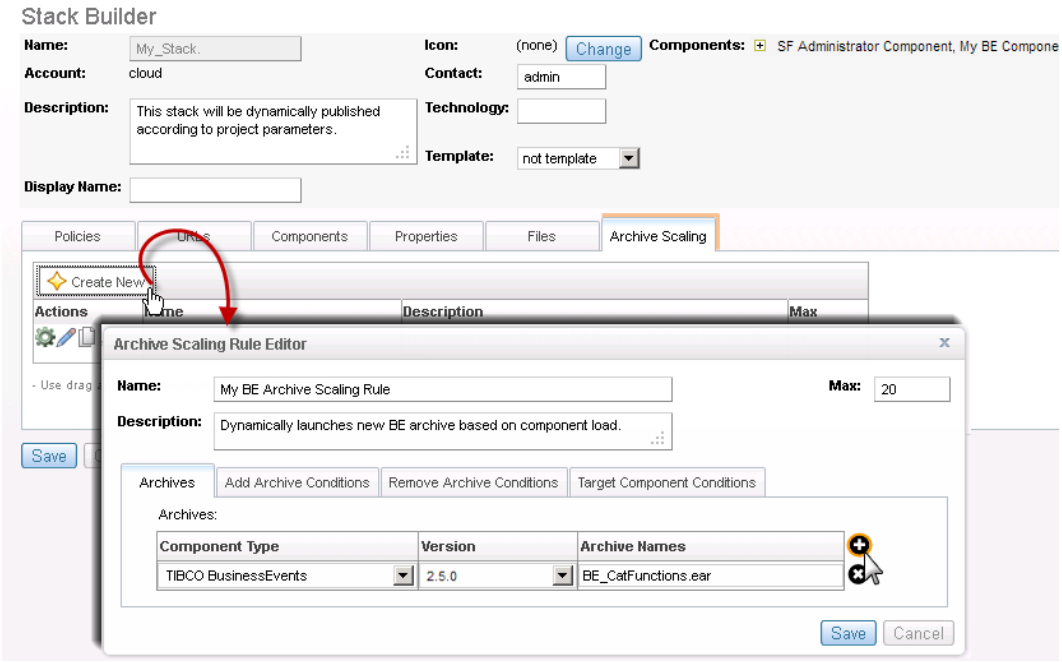
Creating Archive Scaling Rules

You can define stacks that add or remove application archives based on archive statistics that are monitored for triggering conditions. Create Archive Scaling Rules to automate creation or removal of new BusinessEvents application 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 the **Create New** button.
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. The **Archives** tab defines what archive is added or removed according to the rules you define in the other tabs.
4. Use the **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



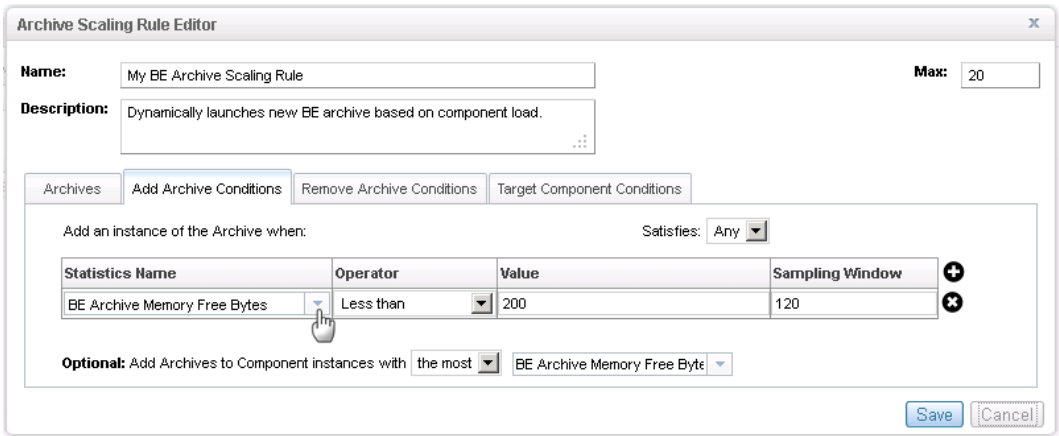
5. Select the BusinessEvents component that was used to upload the application archive file. Use the **Archive Names** field to specify which archive is subject to the rules you set using the other tabs of the Archive Scaling Rule Editor.
6. Use the **Add Archive Conditions** tab and click the **Add** icon to the right of the column heading row to create and define a new Add Archive rule.
7. 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) so that the aggregated statistics is collected. If the sampling window is not big enough statistics might not be reported in a particular time-frame creating an inadvertent trigger condition.

By default allocation statistics such as "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.

8. 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 the addition of a new archive instance.

Define Archive Scaling Rule conditions



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.

9. Using the **Remove Archive Conditions** tab you can 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.
10. Using the **Target Component Conditions** tab you can 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 set on the Add archive conditions tab. Further restrict where the new archive instances might start depending on component instances that have:

    Same Component

    This setting works all the time for 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 BusinessEvents archive does not scale on an 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 a specific enabler should use this option.

    Same Middleware Version

    This selection ensures that a component archive that is created with a dependency on a specific compatible component runs on machines with that appropriate version of the middle ware: TIBCO TRA, TIBCO Hawk, etc.

    Same Enabler and Middleware Version

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

    Only the selections: **From the Same Set of Components** or **Same Enabler and Same Middleware Version works every 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 BusinessEvents component archive, a selection of the "Same Component Type" ensures that the Silver Fabric Broker tries and finds an engine with same BusinessEvents component type on which to run the new BusinessEvents component archive.

    If the application that is to be scaled includes an HTTP Port Variable, you must set the HTTP Port Variable at the service level using TIBCO BusinessEvents® Studio. To make the HTTP Port Variable "service settable" select the **Service Settable** check box in the Global Variable Editor.

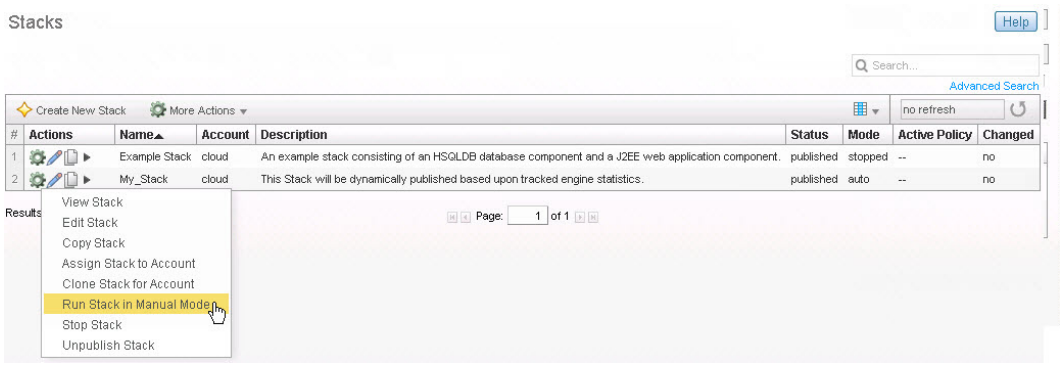
    Refer to the section "Working with Global Variables" in the *TIBCO BusinessEvents Developers Guide* for more information.

Running a Stack

Procedure

- After you have created your stack, you must publish it. Then click **Run Stack In Manual Mode** in the Actions drop-down list as shown in Running a Stack figure.

  Running a Stack



  If you selected a policy schedule while creating a stack, you can run the stack in the auto mode. The stack runs given the schedule defined.

Updating the Stack

When a stack is published and is 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 BusinessEvents Application instantiated on the engine.

Making changes to the stack is as easy as editing, saving, and publishing those changes to any instances that might be 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 and then from the **Actions** list in the main stack page, select **Publish Changes**. The specified engines are affected by the changes immediately.
2. If you want to change a BusinessEvents component, do not stop and restart your entire stack. If you want to either deploy, start, stop, or undeploy Business Events project archives here are a couple of ways to accomplish that:
   - **Micro-scaling:** Start and stop BusinessEvents 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 BusinessEvents endpoints) - publish (deploy), unpublish (undeploy), start, or stop BusinessEvents archives without having to change any stacks, components, or BusinessEvents engines. Deploying BusinessEvents application archives through REST and cURL commands is described in the next section.

Continuous Deployment - Deploying Archives Directly to Endpoints

In some situations, deploying archives directly to a running TIBCO BusinessEvents 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 BusinessEvents instances already running on Silver Fabric engines using the commandline interface (CLI), Silver Fabric API, 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 through REST are described in further detail here.

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 the section "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 Strict Validation setting at **Config** > **Broker** > **General** to "false".

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

Continuous Deployment Life Cycle

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

Deploy

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

Start

Starts an archive that was deployed to an engine.

Stop

Stops an archive that is running on an engine.

Undeploy

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

Deploy

BusinessEvents application archives might be deployed directly to an appropriate Silver Fabric engine (TIBCO Logical Machine) by calling the REST method. In this document cURL syntax is used to show REST inputs in a generic form:

```
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 "CDD=ApplicationCDDName.cdd"]
    [-F "configurationFile=YourConfigurationfile.xml"]
    [-F "ForceDeploy=true"]
    [-F "GV=globalVariableA=123,globalVarB=SomeString"]
    [-F "InstanceSettings.element1.element2=SomeValue"]
    [-F "NoDeploy=true"]
```

```
    [-F "NoStart=true"]
    [-F "NAME=value"]
    [-F "JMX_PORT=value"]
    [-F "GLOBAL_FILE=/opt/qa/config.cfg"]
    [-F "NoStop=true"]
    [-F "PUID=PUID_Value"]
    [-F "VariableProvider=Provider_A, Provider_B, Provider_X"]
    [-F "StreamBase.ServerURI=StreamBase_Server_URI"]
    [-F "StreamBase.UserName=User_Name"]
    [-F "StreamBase.Password=Password"]
```

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 such as 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 can be sent and received properly.

Expressions in the generic form cURL expression, shown in the previous syntax were separated by line breaks to help with readability, but normally a string is submitted in the execution as shown in the following example:

A 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 "CDD=ApplicationCDDName.cdd" -F "PUID=PUID_Value" -
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") are specified in any order.

Mandatory Form-Data

Mandatory form-data contain the files necessary for the deployment.

archiveFile

Specifies your BusinessEvents archive file (.zip or .ear file) to upload to the Silver Fabric Broker, which then publishes the archive to the appropriate Silver Fabric engine. Multiple application archives can 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 in a by specifying the list with the `Archives` form-data field. You can upload the archives using the component wizard also.

If archiveFile is an EAR file, it is mandatory to set PUID and CDD.

CDD

Specify the Cluster Deployment Descriptor (CDD) file used to configure your project for deployment. Refer to the TIBCO ActiveMatrix BusinessEvents documentation on the use of CDD files for more information on which application project parameters can be set and how to generate the CDD file. Example syntax:

```
  -F "CDD=YourCDD_FileName.cdd"
```

PUID

Specify the processing unit ID (PUID) for the engine. Specify the value of PUID as in the *.cdd file or as "*default*"

-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 looks 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.

deploymentFile

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

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

When deploying an archive by REST, you must include the deployment file, which specifies at least one selection criterion 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 criterion, 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, the following is a simple statement to deploy an archive to an engine running a component with a component type that contains BusinessEvents as part of the name and which has a domain name, *YourDomain*:

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

By default, all deployment file criteria statements must be satisfied for the deployment to occur. 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

Name	Definition

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`

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 BusinessEvents Enabler statistic such as:
	`BE Process Count`
	`BE Memory Free Bytes`
	`BE Percentage of Memory Used`
	`BE Average Elapse Execution Time`
	`BE Total Number Error`
	`BE 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.

Example Deployment File

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

Optional Form-Data

Optional form-data fields must be specified only when 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 runs without interruption or replacement.

AppSettings

(Optional) Specifies the settings that your application uses when deployed.

All deployment configuration cURL expressions takes the following 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 within the compressed files that are to be deployed. If an `archives` list is omitted then all archives in the application archive package is deployed. 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 want 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"
```

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 of the person responsible for this application instance.
- `startOnBoot`: when the value is true the service instance starts when the computer is restarted. The default value is false.
- `enableVerbose`: when the value is set as true, the enabler for verbose tracking for service instances is activated. 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.
- `prepandClassPath`: values supplied here are prepended to your CLASSPATH environment variable.
- `appendClassPath`: 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.
- `iv` : This element uses a comma-separated string with name-value pairs with each key value pair joined by an equal (=) sign. For example:

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 the *Silver Fabric Enabler for TIBCO BusinessEvents Developer's Guide.*

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"

AppName

(Optional) Specifies the directory location where the application archive(s) is 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 like *MyAppArchive*, varying the AppName specification gives following behavior:

- If the AppName form-data field is not specified, then *MyAppArchive* is deployed at the top level of t he Administrator directory.
- If -F "AppName=A" is submitted in the curl request, then *MyAppArchive* is renamed to *A* and deployed at the top level.
- If -F "AppName=A/" is sent, then the directory folder *A* is used or it is created and *MyAppArchive* is deployed within that sub-directory.
- If -F "AppName=A/B" is sent, then the sub-directory *A* is used or created, and *MyAppArchive* is deployed there and renamed to *B*.
- If -F "AppName=A/B/" is sent, then the folder *A* with a sub-folder *B* is used or created and *MyAppArchive* is published within sub-folder *B*.

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

ForceDeploy

(Optional) Redeploy, forces a stop and overwrite of a preexisting 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 preexisting deployment of the same name. If there is a change of the archive file, 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, only those archives are stopped, undeployed, and redeployed.

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

GV

(Optional) Sets global variables for use on the targeted application endpoint by the archive. You can define a comma delimited list of declarative name equals value statements using the GV form-data field.

```
-F "GV=globalVariableA=123,globalVarB=SomeString"
```

For example, to change the JMS SSL and Rv Service ports:

```
-F "GV=JmsSslProviderUrl=ssl://localhost:7555,RvService=7222"
```

If global variables are not defined with explicit values in the cURL statement, those values you might have set in the deployment `configurationFile.xml` apply.

If global variables with the same name are set by both REST statement and a specified variable provider, the value set by REST statement overwrites and takes precedence over the value set in the variable provider.

NAME

(Optional) Sets the `-n` parameter in the component wizard when uploading the EAR file. When deploying with REST, use the property `-F "NAME=<name>"`

When running multiple instances of a component, the `-n` parameter value is changed to `<name>_ComponentInstanceID` to maintain the `-n` parameter as a unique value.

JMX_PORT

(Optional) Sets `-propvar jmx_port` in the component wizard when uploading the EAR file. When deploying with REST, use the property `-F "JMX_PORT=<port_number>"`

GLOBAL_FILE

(Optional) Sets -p parameter in the component wizard when uploading the EAR file. When deploying with REST, use the property `-F "GLOBAL_FILE=<FileName>"`

For file configuration uploaded in the component : specify the GLOBAL_FILE value to `${ENGINE_WORK_DIR}/fabric/<path>/<filename>.cfg`

Upload the file `<filename>.cfg` in the component wizard "upload a content file"with the appropriate `<path>`.

For deployment using REST call or file not uploaded in the component specify the full path name of the configuration file in the GLOBAL_FILE value.

The configuration file must accessible from the machine where the component run.

STREAMBASE_SERVERURI

(Required if you are using Streambase Channel) A Uniform Resource Identifier containing information necessary to connect to a StreamBase server. Example: sb://10.128.88.96:10000

When deploying with REST, use the property `-F "StreamBase.ServerURI=<StreamBase_Server_URI>"`.

STREAMBASE_USERNAME

(Required if you are using Streambase Channel) Name of the user if created, to connect to the Streambase server. But it can be left blank if no user is created. When deploying with REST, use the property `-F "StreamBase.UserName=<User_Name>"`.

STREAMBASE_PASSWORD

(Required if you are using Streambase Channel) Password to connect to the Streambase server user if created. It can be left blank if no user is created. When deploying with REST, use the property `-F "StreamBase.Password=<Password>"`.

VariableProvider

(Optional) Specifies one or more variable providers to set global variables for applications deployed with REST. The variable provider is a Java Class extension compiled into a JAR. It is loaded into a Silver Fabric directory with an appropriate XML so that it might be called by REST during application deployment.

Multiple comma-separated variable providers can be specified. If the same variable is in multiple providers, the value from the last provider in the list is used. For example, when specifying *-F "VariableProvider=VP_A,VP_B,VP_C"*, if the same variable is set in VP_A and VP_B, the value in VP_B is used.

In the case there is an error in the name of the variable provider or if it does not exist, a log server error is logged and the deployment does not occur.

```
-F "VariableProvider=Provider_A,Provider_B,Provider_X"
```

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 as follows 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 the REST invocation on the TIBCO Silver Fabric Administrator > Admin > Variables page.

NoDeploy

(Optional) The default value is `false` which means that the archive(s) are uploaded to the Silver Fabric Broker *and* they are deployed to the application endpoints. When `NoDeploy` is set to true, the archives are uploaded with associated service enabler bindings created in TIBCO Administrator, but the archives are not deployed to a Silver Fabric engine and the application endpoint.

```
-F "NoDeploy=true"
```

NoDeploy and NoStart are not supported when TIBCO BusinessEvents has been deployed in stand-alone mode. Archive in the EAR is always deployed and started when it is invoked by REST.

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.

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

STREAMBASE_SERVERURI

(Required if you are using Streambase Channel) A Uniform Resource Identifier containing information necessary to connect to a StreamBase server. Example: sb://10.128.88.96:10000

STREAMBASE_USERNAME

(Required if you are using Streambase Channel) Name of the user if created, to connect to the Streambase server. But it can be left blank if no user is created.

STREAMBASE_PASSWORD

(Required if you are using Streambase Channel) Password to connect to the Streambase server user if created. It can be left blank if no user is created.

The above Streambase properties are applicable only when your application channel is using Streambase channel.

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
}
```

You can invoke the START, STOP, and UNDEPLOY methods to enable full control of the Archive life cycle using the Engine-Id, Engine-Instance, and the full ArchiveName.

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 were deployed, but not started. You must know the Engine-Id, Engine-Instance, and the full application ArchiveName.

Here 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/engines/{engine-
id}/{engine-instance}/archives/{appname}/start"
```

Expressions in the generic form cURL expression shown in the previous syntax were separated by line breaks to help with readability, but normally an unbroken string is submitted in the execution as in the example shown in the tip on {appname}.

The values of `{engine-id}` and `{engine-instance}` might 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.

Silver Fabric REST methods to GET the engine-id and engine-instance exist 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**.

If the component is configured to use Full Archive Name, and the {appname} does not have a .par, or .bar extension then the {appname} can be copied from the TIBCO Silver Fabric Administrator > Dashboard > **Scaled Archives** page.

The {appname} must be 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 that starts an Archive.

The full archive name shown as follows had to be URL encoded:

```
/Silver Fabric/SFBE 22E5/DomainMachineName/processOrder/Orders/MyProcOrder/
Process_Archive.bar
```

It can now be passed in the cURL statement as follows:

```
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%2FSFBE%2022E5%2FDomainMachineName
%2FprocessOrder%2FOrders%2FMyProcOrder/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/engines/{engine-
id}/{engine-instance}/archives/{appname} /{archive-id}/stop"
```

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

Edit Deployment

Using REST you can edit an application deployment. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that edits an Archive.

```
curl -u <USER>:<PASSWORD> -X POST -H
"Accept: application/json" -H "Content-Type: multipart/form-data"
-v
"<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/
<DAEMON_ID>/<ENGINE_INSTANCE>/archives"
-F "archiveFile=@<ARCHIVE_FILE>" -F
"COMMAND=editdeployment" -F
"CDD=<CDD_FILE>"
```

Import Deployment

Using REST you can import an application using the specified CDD, EAR, and site. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that imports an Archive

```
curl -u <USER>:<PASSWORD> -X POST -H
"Accept: application/json" -H "Content-Type: multipart/form-data"
-v
"<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/
<DAEMON_ID>/<ENGINE_INSTANCE>/archives"
-F "archiveFile=@<ARCHIVE_FILE>" -F
"COMMAND=importdeployment" -F "STFILE=<ST_FILE>" -F"
"CDD=<CDD_FILE>
```

Here,

ST_FILE: The name of the site topology file.

Delete Deployment

Using REST you can delete an application deployment. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that deletes an Archive.

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json" -H "Content-Type:
multipart/form-data" -v "<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/
<DAEMON_ID>/<ENGINE_INSTANCE>/archives" -F "archiveFile=@<ARCHIVE_FILE>" -F
"COMMAND=editdeployment" -F "CDD=<CDD_FILE>"
```

Update JVM Property

Using REST you can update the JVM Property. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that updates the JVM Property.

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json" -H "Content-Type:
multipart/form-data" -v
"<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/<DAEMON_ID>/
<ENGINE_INSTANCE>/archives" -F
"archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=savejvmproperty" -F
"INSTANCES=<INSTANCE_1,INSTANCE_2,...,INSTANCE_n>" -F
"PROPNAME=<PROPNAME_IN_JVMPROPERTIES_FILE>" -F "PROPVALUE=<NEW_PROPVALUE>"
```

Here,

INSTANCES : Identifies multiple instance names.

PROPNAME : The JVM property name.

PROPVALUE: The JVM property value

Update Property

Using REST you can update BE property for one or multiple instances. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that updates BE property for one or multiple instances.

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json" -H "Content-Type:
multipart/form-data" -v
"<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/ <DAEMON_ID>/
<ENGINE_INSTANCE>/archives" -F
"archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=savebeproperty" -F
"INSTANCES=<INSTANCE_1,INSTANCE_2,...,INSTANCE_n>"
-F "PROPNAME=<PROPNAME_IN_BEPROPERTIES_FILE>" -F "PROPVALUE=<NEW_PROPVALUE>"
```

Here,

INSTANCES : Identifies multiple instance names

PROPNAME : The BE property name.

PROPVALUE : The BE property value

Update a Global Variable

Using REST you can update global variables for one or multiple instances. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that updates global variable for one or multiple instances.

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json" -H "Content-Type:
multipart/form-data"
-v "<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/<DAEMON_ID>/
 <ENGINE_INSTANCE>/archives"
-F "archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=saveglobalvariable" -F "INSTANCES=
  <INSTANCE_1,INSTANCE_2,...,INSTANCE_n>"
-F "VARNAME=<VARNAME_IN_GLOBALVARIABLES_FILE>" -F "VARVALUE=<NEW_VARVALUE>"
```

Here,

INSTANCES : Identifies multiple instance names

VARNAME : The global variable name.

VARVALUE : The global variable value.

Copy Instance

Using REST you can copy an existing instance of an application. The JMX username and password is governed by policy in the CDD file. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that copy an existing instance of an application:

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json"
-H "Content-Type: multipart/form-data"
-v "<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/<DAEMON_ID>/
<ENGINE_INSTANCE>/archives"
-F "archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=copyinstance" -F
"INSTANCENAME=<INSTANCE_NAME>"
-F "NEWINSTANCENAME=<NEW_INSTANCE_NAME>" -F "PUID=<PU>" -F "JMX_PORT=<JMX_PORT>"
-F "JMXUSER=<JMX_USER>" -F "JMXPASSWORD=<JMX_PASSWORD>"
-F "BEHOME=<BE_HOME>
```

Here,

COMMAND: copyinstance

archiveFile: The application ear file.

INSTANCENAME : The application instance name.

NEWINSTANCENAME : The new application instance name.

PUID : The processing unit name.

JMX_PORT : The JMX port number.

JMXUSER (optional) : The JMX user name

JMXPASSWORD (optional) The JMX password

BEHOME: (optional): BusinessEvents Home

These commands need to be run in the same directory where the ear file is, similar to the regular archive deployment.

Edit Instance

Using REST you can edit an existing instance of an application. The JMX username and password is governed by policy in the CDD file. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that copy an existing instance of an application:

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json"
-H "Content-Type: multipart/form-data"
-v "<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/<DAEMON_ID>/
<ENGINE_INSTANCE>/archives"
-F "archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=editInstance" -F
"INSTANCENAME=<INSTANCE_NAME>"
-F "PUID=<PU>" -F "JMX_PORT=<JMX_PORT>" -F "JMXUSER=<JMX_USER>"
-F "JMXPASSWORD=<JMX_PASSWORD>" -F "BEHOME=<BE_HOME>
```

Here,

COMMAND: editInstance

archiveFile: The application ear file.

INSTANCENAME : The application instance name.

PUID : The processing unit name.

JMX_PORT : The JMX port number.

JMXUSER (optional) : The JMX user name

JMXPASSWORD (optional) The JMX password

BEHOME: (optional): BusinessEvents Home

These commands need to be run in the same directory where the ear file is, similar to the regular archive deployment.

Update System Property

Using REST you can update System property for one or multiple instances. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that updates system property for one or multiple instances.

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json" -H "Content-Type:
multipart/form-data" -v
"<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/<DAEMON_ID>/
<ENGINE_INSTANCE>/archives"
 -F "archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=savesystemproperty" -F
"INSTANCES=<INSTANCE_1,INSTANCE_2,...,INSTANCE_n>"
 -F "PROPNAME=<PROPNAME_IN_SYSTEMPROPERTIES_FILE>" -F "PROPVALUE=<NEW_PROPVALUE>"
```

Here,

INSTANCES : Identifies multiple instance names

PROPNAME : The system property name.

PROPVALUE : The system property value

Hot Deploy

Using REST you can hotdeploy an application provided by the EAR file. It is supported only when you are using deployment with TIBCO Enterprise Administrator mode.

Here is a generic cURL example that hotdeploys an application provided by the EAR file:

```
curl -u <USER>:<PASSWORD> -X POST -H "Accept: application/json"
-H "Content-Type: multipart/form-data"
-v "<BROKER_URL>:<BROKER_PORT>/livecluster/rest/v1/sf/engines/<DAEMON_ID>/
<ENGINE_INSTANCE>/archives"
-F "archiveFile=@<ARCHIVE_FILE>" -F "COMMAND=hotdeploy"
```

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"
     -F "DeleteApp=true"
```

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

DeleteApp

(Optional - for 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.

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

The Archive Management Support Feature

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

Ant Scripts

In addition to Administration Tool, commandline 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. It is 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 to "Silver Fabric Ant Scripts" in *TIBCO Silver Fabric Developer's Guide.*

Samples

The Silver Fabric Commandline Interface installation contains a samples directory, which contains example build files using Ant tasks for several enablers. See the enclosed readme files 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

The following is a sample build.xml file.

```
<project name="Stack_BE-build" default="release" basedir="."
xmlns:sf="antlib:com.datasynapse.fabric.ant">

<property file="build.properties" />
<property file="fabric-cli.properties" />

<sf:connection-props brokerurl="${DSPrimaryDirector}" username="${DSUserName}"
password="${DSPassword}"
clientssltrustfile="${DSSSLTrustFile}" />    <target name="release"
depends="create-all-components,
release-SFBEV15-component, release-stack" />
<target name="clean" depends="clean-stack, clean-SFBEV15-component" />

 <target name="create-all-components">
<sf:component action="create" name="${SFBEV15.component.name}"
description="${SFBEV15.component.description}" type="${SFBEV15.component.type}"
enablername="${SFBEV15.enabler.name}" enablerversion=
  "${SFBEV15.enabler.version}"
 utility="${SFBEV15.utility}" />    </target>
 <target name="release-SFBEV15-component">
 <echo message="Building ${SFBEV15.component.name}" />

<sf:option name="${SFBEV15.component.name}" action="replace">
<sf:property name="Department" value="${SFBEV15.department}" />
<sf:property name="Location" value="${SFBEV15.location}" />
<sf:property name="Partition" value="${SFBEV15.partition}" />
<sf:property name="Engine Blacklisting" value="${SFBEV15.blacklisting}" /
>
<sf:property name="Failures Per Day Before Blacklist"
   value="${SFBEV15.blacklist.failures}" />
<sf:property name="Archive Scale Up Timeout"
    value="${SFBEV15.archive.scale.up.timeout}" />
<sf:property name="Archive Scale Down Timeout"
value="${SFBEV15.archive.scale.down.timeout}" />
<sf:property name="Maximum Deactivation Time"
```

```
value="${SFBEV15.deactivation.timeout}" />
<sf:property name="Maximum Activation Time"
value="${SFBEV15.activation.timeout}" />
<sf:property name="Maximum Capture Time"
value="${SFBEV15.maximum.capture.time}" />
<sf:property name="Maximum Instances Per Host"
value="${SFBEV15.max.instances.per.host}" />
<sf:property name="Statistics Collection Frequency"
value="${SFBEV15.stats.collection.frequency}" />
<sf:property name="Activation Delay" value="${SFBEV15.activation.delay}" /
>
<sf:property name="Engine Reservation Expiration"
value="${SFBEV15.engine.reservation.expiration}" />
</sf:option>

<sf:default-settings name="${SFBEV15.component.name}" action="update">
  <sf:property name="Default Min Engines" value="${SFBEV15.min}" />
  <sf:property name="Default Max Engines" value="${SFBEV15.max}" />
  <sf:property name="Default Priority" value="${SFBEV15.priority}" />
</sf:default-settings>


<sf:feature name="${SFBEV15.component.name}" action="add"
  feature="HTTP Support" />
<sf:feature name="${SFBEV15.component.name}"
 action="update" feature="HTTP Support">
  <sf:property name="Relative Url" value="${SFBEV15.relative.url}" />
  <sf:property name="Routing Prefix" value="${SFBEV15.routing.prefix}" /
>
  <sf:property name="HTTPS Enabled" value="${SFBEV15.https.enabled}" />
  <sf:property name="HTTP Enabled" value="${SFBEV15.http.enabled}" />
  <sf:property name="Route Directly To Endpoints"
value="${SFBEV15.route.directly.to.endpoints}" />
</sf:feature>

<sf:feature name="${SFBEV15.component.name}" action="add"
feature="Archive Management Support" />
<sf:feature name="${SFBEV15.component.name}" action="update"
feature="Archive Management Support">
  <sf:property name="Archive" value="${SFBEV15.archive}" />
  <sf:property name="Start Archives On Activation"
value="${SFBEV15.start.archives.on.activation}" />
</sf:feature>

<sf:feature name="${SFBEV15.component.name}" action="add"
feature="Application Logging Support" />
<sf:feature name="${SFBEV15.component.name}" action="update"
feature="Application Logging Support">
  <sf:property name="Checkpoint Frequency In Seconds"
value="${SFBEV15.checkpoint.frequency.in.seconds}" />
  <sf:property name="Archive Application Logs"
value="${SFBEV15.archive.application.logs}" />
  <sf:property name="Log File Pattern" value="${SFBEV15.log.file.pattern}" /
>
</sf:feature>

<sf:content-file type="component" name="${SFBEV15.component.name}"
action="add">
  <sf:contentset dir="${SFBEV15.content.dir}">
   <include name="**/*.*" />
  </sf:contentset>
</sf:content-file>

<sf:archive-file name="${SFBEV15.component.name}" action="add" >
 <path>
   <pathelement path="${SFBEV15.archive.0.path}" />
 </path>
</sf:archive-file>



<sf:context-variable type="component" name="${SFBEV15.component.name}"
```

```
action="update">
  <sf:contextvar name="NB_HAWK_RESTART_BEFORE_RESTART_ENGINE" type="string"
   value="${SFBEV15.contextvar.nb.hawk.restart.before.restart.engine}"
   export="false" autoincrement="none" description="null" />
   <sf:contextvar name="HAWK_POLLPERIOD" type="string"
   value="${SFBEV15.contextvar.hawk.pollperiod}"
   export="false" autoincrement="none" description="null" />
   <sf:contextvar name="NB_BE_RESTART_BEFORE_RESTART_ENGINE" type="string"
   value="${SFBEV15.contextvar.nb.be.restart.before.restart.engine}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="BE_POLLPERIOD" type="string"
value="${SFBEV15.contextvar.be.pollperiod}" export="false"
autoincrement="none" description="null" />
<sf:contextvar name="HTTP_PORT_BASE" type="environment"
value="${SFBEV15.contextvar.http.port.base}"
export="false" autoincrement="none" description="Consult the user
documentation on automated HTTP port setting" />
<sf:contextvar name="HTTP_PORT_INCREASE" type="environment"
value="${SFBEV15.contextvar.http.port.increase}"
export="false" autoincrement="none"
description="Consult the user documentation on automated HTTP port setting" /
>
<sf:contextvar name="IS_PREPAND" type="environment"
value="${SFBEV15.contextvar.is.prepand}" export="false"
autoincrement="none" description="null" />
<sf:contextvar name="EXTERNAL_JAR_FILE" type="environment"
value="${SFBEV15.contextvar.external.jar.file}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="PLUGIN_CONFIG_FILE" type="environment"
value="${SFBEV15.contextvar.plugin.config.file}"
 export="false" autoincrement="none" description="null" />
<sf:contextvar name="USE_OPT_EMS_DISTRO" type="environment"
value="${SFBEV15.contextvar.use.opt.ems.distro}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="USE_OPT_SB_DISTRO" type="environment"
value="${SFBEV15.contextvar.use.opt.sb.distro}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="USE_DEPEND_TEASERVER" type="environment"
value="${SFBEV15.contextvar.use.depend.teaserver}"
export="false" autoincrement="none"
description="Requires a running TEA component" />
<sf:contextvar name="BE_TEAGENT_PORT_BASE"
type="environment" value="${SFBEV15.contextvar.be.teagent.port.base}"
export="false" autoincrement="numeric" description="null" />
<sf:contextvar name="TEA_SERVER_URL" type="environment"
value="${SFBEV15.contextvar.tea.server.url}"
export="false" autoincrement="none"
description="URL of existing TEA Server to connect to" />
<sf:contextvar name="TEA_SERVER_USERNAME" type="environment"
value="${SFBEV15.contextvar.tea.server.username}"
export="false" autoincrement="none"
description="Username of existing TEA Server to connect to" />
<sf:contextvar name="BE_TEA_USE_CUSTOM_SSH_KEYS" type="environment"
value="${SFBEV15.contextvar.be.tea.use.custom.ssh.keys}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="BE_TEA_SSH_PRIVATE_KEY" type="environment"
value="${SFBEV15.contextvar.be.tea.ssh.private.key}"
export="false" autoincrement="none"
description="Private key file path for password-less SSH authentication" /
>
<sf:contextvar name="BE_TEA_SSH_PUBLIC_KEY" type="environment"
value="${SFBEV15.contextvar.be.tea.ssh.public.key}"
export="false" autoincrement="none" description="null" />

<sf:contextvar name="TEA_SSL_ENABLED" type="environment" value=
  "${BE_Component.contextvar.tea.ssl.enabled}" export="false"
   autoincrement="none" description="null" />
<sf:contextvar name="BE_TEA_SSL_SERVER_CERT" type="environment"
 value="${BE_Component.contextvar.be.tea.ssl.server.cert}" export="false"
  autoincrement="none" description="Server SSL certificate path" />
 <sf:contextvar name="BE_TEA_SSL_CLIENT_CERT" type="environment"
  value="${BE_Component.contextvar.be.tea.ssl.client.cert}"
```

```
   export="false" autoincrement="none" description="Client SSL
   certificate path" />
  <sf:contextvar name="BE_JRE_VERSION" type="environment" value=
   "${BE_Component.contextvar.be.jre.version}" export="false"
   autoincrement="none" description="The JRE version required for
   BusinessEvents" />
   <sf:contextvar name="BE_TEA_AGENT_REQUEST_TIMEOUT" type="environment"
    value="${BE_Component.contextvar.be.tea.agent.request.timeout}"
   export="false" autoincrement="none" description="The TIBCO BusinessEvents
    Enterprise Administrator Agent request timeout" />

<sf:contextvar name="BE_USE_SB_CHANNEL" type="environment"
value="${SFBEV15.contextvar.be.use.sb.channel}"
export="false" autoincrement="none"
description="Requires BE 5.4.1.0.0 distribution or later" />
<sf:contextvar name="DELETE_BE_TEA_MACHINE_ON_SHUTDOWN" type="environment"
value="${SFBEV15.contextvar.delete.be.tea.machine.on.shutdown}"
export="false" autoincrement="none"
description="Condition to delete the machine registered in TEA Server.
Works in fast TLM restart" />
<sf:contextvar name="JDBC_DRIVER_FILE" type="environment"
value="${SFBEV15.contextvar.jdbc.driver.file}"
export="false" autoincrement="none"
description="Only if the Domain is stored in a database" />
<sf:contextvar name="DOMAIN_ENABLE_EMS_SSL" type="environment"
value="${SFBEV15.contextvar.domain.enable.ems.ssl}"
export="false" autoincrement="none"
description="If EMS SSL is enable for the domain" />
<sf:contextvar name="EMS_SSL_TRUSTED" type="environment"
value="${SFBEV15.contextvar.ems.ssl.trusted}" export="false"
autoincrement="none" description="If EMS SSL is enable for the domain" /
>
<sf:contextvar name="EMS_SSL_IDENTITY" type="environment"
value="${SFBEV15.contextvar.ems.ssl.identity}" export="false"
autoincrement="none" description="If EMS SSL is enable for the domain" /
>
<sf:contextvar name="EMS_SSL_PRIVATE_KEY" type="environment"
value="${SFBEV15.contextvar.ems.ssl.private.key}"
export="false" autoincrement="none"
description="If EMS SSL is enable for the domain" />
<sf:contextvar name="NOTIFIER_EMS_SSL_VENDOR" type="environment"
value="${SFBEV15.contextvar.notifier.ems.ssl.vendor}"
export="false" autoincrement="none" description="The EMS SSL vendor" />
<sf:contextvar name="DELETE_APPLICATION_CONF_AT_SHUTDOWN"
type="environment"
value="${SFBEV15.contextvar.delete.application.conf.at.shutdown}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="DO_NOT_REDEPLOY_EAR_FILE_AT_STARTUP" type="environment"
value="${SFBEV15.contextvar.do.not.redeploy.ear.file.at.startup}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="MANAGED_PROCESS_HAWK_AGENT_ENABLED" type="environment"
value="${SFBEV15.contextvar.managed.process.hawk.agent.enabled}"
export="false" autoincrement="none"
description="If the regular stop is not succesfull" />
<sf:contextvar name="MANAGED_PROCESS_SERVICE_ENABLED" type="environment"
value="${SFBEV15.contextvar.managed.process.service.enabled}"
export="false" autoincrement="none"
description="If the regular stop is not succesfull" />
<sf:contextvar name="LOGICAL_MACHINE_NAME" type="environment"
value="${SFBEV15.contextvar.logical.machine.name}" export="false"
autoincrement="none"
description="Unique Application Component machine name instantiated only once" /
>
<sf:contextvar name="DOMAINDATA_DIR" type="environment"
value="${SFBEV15.contextvar.domaindata.dir}"
export="false" autoincrement="none" description="Share drive where deployment
  configuration will be created" />
<sf:contextvar name="TIBCO_SERVICES_STATE_AFTER_TLM_MOVED"
type="environment"
value="${SFBEV15.contextvar.tibco.services.state.after.tlm.moved}"
export="false" autoincrement="none"
description="On logical machine restart, all services are either
```

```
 started or stopped" />
<sf:contextvar name="USE_FULL_ARCHIVE_NAME" type="environment"
value="${SFBEV15.contextvar.use.full.archive.name}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="WITHOUT_TRA" type="environment"
value="${SFBEV15.contextvar.without.tra}" export="false"
autoincrement="none" description="null" />
<sf:contextvar name="BE_EXECUTION_MODE"
type="environment" value="${SFBEV15.contextvar.be.execution.mode}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="AMI_HAWK_SERVICE" type="environment"
value="${SFBEV15.contextvar.ami.hawk.service}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="AMI_HAWK_DAEMON" type="environment"
value="${SFBEV15.contextvar.ami.hawk.daemon}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="AMI_HAWK_NETWORK" type="environment"
value="${SFBEV15.contextvar.ami.hawk.network}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="HAWK_PLUGIN_CONFIG_FILE" type="environment"
value="${SFBEV15.contextvar.hawk.plugin.config.file}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="ACTION_TO_PERFORM_FOR_HAWK_FILE" type="environment"
value="${SFBEV15.contextvar.action.to.perform.for.hawk.file}"
export="false" autoincrement="none" description="null" />
<sf:contextvar name="HTTPChannel_STREAMBASE_PASSWORD"
type="string" value="${SFBEV15.contextvar.httpchannel.streambase.password}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_JMX_PORT" type="string"
value="${SFBEV15.contextvar.httpchannel.jmx.port}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_STREAMBASE_SERVERURI" type="string"
value="${SFBEV15.contextvar.httpchannel.streambase.serveruri}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_PUID" type="string"
value="${SFBEV15.contextvar.httpchannel.puid}" export="true"
autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_CDD" type="string"
value="${SFBEV15.contextvar.httpchannel.cdd}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_NAME" type="string"
value="${SFBEV15.contextvar.httpchannel.name}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_STREAMBASE_USERNAME" type="string"
value="${SFBEV15.contextvar.httpchannel.streambase.username}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="HTTPChannel_GLOBAL_FILE" type="string"
value="${SFBEV15.contextvar.httpchannel.global.file}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="fdsb_STREAMBASE_SERVERURI"
type="string" value="${SFBEV15.contextvar.fdsb.streambase.serveruri}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="fdsb_STREAMBASE_PASSWORD" type="string"
value="${SFBEV15.contextvar.fdsb.streambase.password}" export="true"
autoincrement="none" description="" />
<sf:contextvar name="fdsb_NAME" type="string"
value="${SFBEV15.contextvar.fdsb.name}" export="true"
autoincrement="none" description="" />
<sf:contextvar name="fdsb_PUID" type="string"
value="${SFBEV15.contextvar.fdsb.puid}" export="true" autoincrement="none"
description="" />
<sf:contextvar name="fdsb_GLOBAL_FILE" type="string"
value="${SFBEV15.contextvar.fdsb.global.file}" export="true"
autoincrement="none" description="" />
<sf:contextvar name="fdsb_JMX_PORT" type="string"
value="${SFBEV15.contextvar.fdsb.jmx.port}" export="true" autoincrement="none"
description="" />
<sf:contextvar name="fdsb_STREAMBASE_USERNAME" type="string"
value="${SFBEV15.contextvar.fdsb.streambase.username}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="fdsb_CDD" type="string"
value="${SFBEV15.contextvar.fdsb.cdd}" export="true" autoincrement="none"
```

```
description="" />            <sf:contextvar name="DEMOA"
type="string" value="${SFBEV15.contextvar.demoa}" export="false"
autoincrement="none" description="" />
<sf:contextvar name="fd_STREAMBASE_PASSWORD" type="string"
value="${SFBEV15.contextvar.fd.streambase.password}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="fd_STREAMBASE_USERNAME"
type="string" value="${SFBEV15.contextvar.fd.streambase.username}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="fd_STREAMBASE_SERVERURI" type="string"
value="${SFBEV15.contextvar.fd.streambase.serveruri}"
export="true" autoincrement="none" description="" />
<sf:contextvar name="fd_CDD" type="string"
value="${SFBEV15.contextvar.fd.cdd}" export="true"
autoincrement="none" description="" />
<sf:contextvar name="fd_PUID" type="string"
value="${SFBEV15.contextvar.fd.puid}" export="true" autoincrement="none"
description="" />
<sf:contextvar name="fd_JMX_PORT" type="string"
value="${SFBEV15.contextvar.fd.jmx.port}" export="true"
autoincrement="none" description="" />
<sf:contextvar name="fd_GLOBAL_FILE" type="string"
value="${SFBEV15.contextvar.fd.global.file}" export="true"
 autoincrement="none"
description="" />
<sf:contextvar name="fd_NAME" type="string"
 value="${SFBEV15.contextvar.fd.name}"
 export="true" autoincrement="none"
description="" />
<sf:contextvar name="TEA_SERVER_PASSWORD" type="encrypted"
value="${SFBEV15.contextvar.tea.server.password}"
export="false" autoincrement="none"
description="Password of existing TEA Server to connect to" />
<sf:contextvar name="BE_TEA_SSH_KEY_PASSPHRASE" type="encrypted"
value="${SFBEV15.contextvar.be.tea.ssh.key.passphrase}"
export="false" autoincrement="none"
description="Passphrase to private key file for password-less
 SSH authentication" />
</sf:context-variable>

<sf:publish type="component" name="${SFBEV15.component.name}" />
</target>


<target name="clean-SFBEV15-component">
<echo message="Cleaning ${SFBEV15.component.name}" />
<sf:unpublish type="component" name="${SFBEV15.component.name}"
 onerror="ignore" />
<sf:remove type="component"
name="${SFBEV15.component.name}" onerror="ignore" />    </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="${SFBEV15.component.name}" /><!--  uncomment
this section to create schedules-->
<sf:stack-policy action="update" name="${stack.name}" >
<sf:policy manualpolicy="true">
<sf:compalloc component="${SFBEV15.component.name}"
min="${Stack_BE.policy.0SFBEV15.min}"
max="${Stack_BE.policy.0SFBEV15.max}"
 priority="${Stack_BE.policy.0SFBEV15.priority}"/>
</sf:policy>        </sf:stack-policy>
<sf:publish type="stack" name="${stack.name}" />
<sf:stack-mode name="${stack.name}" action="run" mode="${stack.mode}" />
</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" />
<!--  uncomment this section to remove schedules on clean-->
</target>

</project>
```

build.properties

Following is a sample of build.properties file.

```
#properties for the build.xml#Fri May 26 00:55:33 EDT 2017stack.name=Stack_BE
stack.description=BE_Component.component.name=BE_Component
BE_Component.component.description=BE_Component.component.type=
 TIBCO BusinessEvents\:3.3.0.0
BE_Component.enabler.name=TIBCO BusinessEvents container
BE_Component.enabler.version=3.3.0.0
BE_Component.utility=false
BE_Component.middleware.versions=TIBCO_BusinessEvents_distribution\:
 5.4.1.0.0,TIBCO_ActiveSpaces_distribution\:2.2.1
BE_Component.optional.middlware.versions=TIBCO_HAWK_distribution\:
 5.1.1.0.0,TIBCO_RV_distribution\:8.4.2.0.0
BE_Component.blacklisting=false
BE_Component.blacklist.failures=0
BE_Component.archive.scale.up.timeout=120
BE_Component.archive.scale.down.timeout=120
BE_Component.deactivation.timeout=3600
BE_Component.activation.timeout=3600
BE_Component.maximum.capture.time=180
BE_Component.max.instances.per.host=0
BE_Component.stats.collection.frequency=10
BE_Component.activation.delay=0
BE_Component.engine.reservation.expiration=300
BE_Component.min=1
BE_Component.max=1
BE_Component.priority=3
BE_Component.relative.url=BE_Component.routing.prefix=
 BE_Component.https.enabled=false
BE_Component.http.enabled=true
BE_Component.route.directly.to.endpoints=false
BE_Component.archive=fdsb.ear
BE_Component.start.archives.on.activation=true
BE_Component.checkpoint.frequency.in.seconds=300
BE_Component.archive.application.
 logs=true
BE_Component.log.file.pattern=/../domaindata/logs/.*\\.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
BE_Component.archive.0.path=BE_Component/archives/fdsb.ear
BE_Component.contextvar.nb.hawk.restart.before.restart.engine=3
BE_Component.contextvar.hawk.pollperiod=30
BE_Component.contextvar.nb.be.restart.before.restart.engine=3
BE_Component.contextvar.be.pollperiod=30
BE_Component.contextvar.http.port.base=8200
BE_Component.contextvar.http.port.increase=50
BE_Component.contextvar.is.prepand=BE_Component.contextvar.external.jar.file=
 BE_Component.contextvar.plugin.config.file=
BE_Component.contextvar.use.opt.ems.distro=true
BE_Component.contextvar.use.opt.
 sb.distro=false
BE_Component.contextvar.use.depend.teaserver=true
BE_Component.contextvar.be.teagent.port.base=9777
BE_Component.contextvar.be.teagent.jmx.port.base=5566
BE_Component.contextvar.tea.server.url=http\://localhost\:8777/tea
BE_Component.contextvar.tea.server.username=admin
BE_Component.contextvar.be.tea.use.custom.ssh.keys=false
BE_Component.contextvar.be.tea.ssh.private.key=
BE_Component.contextvar.be.tea.ssh.public.key=
BE_Component.contextvar.tea.ssl.enabled=false
BE_Component.contextvar.be.tea.ssl.server.cert=
```

```
BE_Component.contextvar.be.tea.ssl.client.cert=
BE_Component.contextvar.be.jre.version=1.8.0
BE_Component.contextvar.be.tea.agent.request.timeout=10
BE_Component.contextvar.delete.be.tea.machine.on.shutdown=false
BE_Component.contextvar.jdbc.driver.file=
BE_Component.contextvar.domain.enable.ems.ssl=false
BE_Component.contextvar.ems.ssl.trusted=BE_Component.contextvar.ems.ssl.identity=
BE_Component.contextvar.ems.ssl.private.key=BE_Component.contextvar.notifier.ems.
 ssl.vendor=j2se
BE_Component.contextvar.delete.application.conf.at.shutdown=false
BE_Component.contextvar.do.not.redeploy.ear.file.at.startup=false
BE_Component.contextvar.managed.process.hawk.agent.enabled=true
BE_Component.contextvar.managed.process.service.enabled=true
BE_Component.contextvar.logical.machine.name=
BE_Component.contextvar.domaindata.dir=BE_Component.contextvar.tibco.services.
 state.after.tlm.moved=Started
BE_Component.contextvar.tea.tibco.services.state.after.tlm.moved=Started
BE_Component.contextvar.use.full.archive.name=true
BE_Component.contextvar.without.tra=false
BE_Component.contextvar.be.execution.mode=BE with TEA Integration
BE_Component.contextvar.ami.hawk.service=7475
BE_Component.contextvar.ami.hawk.daemon=tcp\:7474
BE_Component.contextvar.ami.
 hawk.network=BE_Component.contextvar.hawk.plugin.config.file=
BE_Component.contextvar.action.to.perform.for.hawk.file=
BE_Component.contextvar.fdsb.streambase.serveruri=sb\://lin64vm451.rofa.tibco.com\:
 10000/BE_Component.contextvar.fdsb.name=
BE_Component.contextvar.fdsb.puid=BE_Component.contextvar.fdsb.global.file=
 BE_Component.contextvar.fdsb.jmx.port=
BE_Component.contextvar.fdsb.streambase.username=admin
BE_Component.contextvar.fdsb.cdd=fdsb.cdd
BE_Component.contextvar.tea.server.
 password=admin
BE_Component.contextvar.be.tea.ssh.key.passphrase=BE_Component.contextvar.fdsb.
 streambase.password=adminStack_BE.policy.0
BE_Component.min=1Stack_BE.policy.0BE_Component.max=
 1Stack_BE.policy.0BE_Component.priority=3Stack_BE.policy.0.BE_Component.rule.0.
component.name=TEA_ComponentStack_BE.policy.0.BE_Component.rule.0.shutdown=
 truestack.mode=stopped
```

HTTP Load Balancer for a BusinessEvents Stack

In a private cloud, when you run TIBCO BusinessEvents, you do not know in advance the address of the machine or where it can run unless you set resource preferences in the rules.

The URL to invoke the BusinessEvents endpoint when used with an Administrator domain (a web service using SOAP over HTTP transport or an HTTP Receiver activity) is in the following form:

http://*BrokerMachineName:BrokerPort/TIBCO_ADMIN_DOMAIN/BE-APPLICATION-PATH.NAME/ARCHIVE_ID/BE_HttpPortName*

Where all the italicized parts are concatenated to form this HTTP URL.

- *BrokerMachineName*: machine name or IP Address where you installed TIBCO Silver® Fabric.
- *BrokerPort*: port of the Silver® Fabric Administrator GUI. The default value is 8080.
- *TIBCO_ADMIN_DOMAIN*: value of the Domain name you entered when you configured TIBCO Administrator component.
- *BE-APPLICATION-PATH.NAME*: deployment directory and name of the BusinessEvents application. The folder and application name is delimited by a period. For example if the BusinessEvents application is named *AppName* and was deployed in the TIBCO Admin directory say: `aaa/bbb/ccc/` then the *{BE-APPLICATION-PATH.NAME}* would be: `aaa.bbb.ccc.AppName`. The *{BE-APPLICATION-PATH.NAME}* can be copied from the TIBCO Silver Fabric Administrator and then **Dashboard**, **Scaled Archives page**.
- *ARCHIVE_ID:* name of the archive as specified by the project archive name and the runtime directory in which it is published.
- *BE_HttpPortName*: global variable, endpoint URI, is defined at archive design-time as the HTTP port created to accept requests. Use TIBCO Studio to create the variable (the name is arbitrary) in the global variable group. It should be set as modifiable at the service level so that the value can be changed at run time.

  The base port value for HTTP request activity and SOAP/HTTP Web service activity can be changed at the component level as shown in preceding section.

  - If deployed from TIBCO Administrator UI or AppManage, the default value is set in designer.
  - If deployed from BE Enabler Component (REST, .ear file uploaded), the HTTP port is calculated to avoid port conflict.

For example:

http://10.107.172.95:8080/MyDomain/HTTPChannel/HTTPChannel.bar/httpport1/Channels/HTTP/PostEmpl

If your BusinessEvents stack invokes other BusinessEvents components, you can set the HTTP load balancer value as a global variable.

When BusinessEvents is used in standalone mode, the URL is:

http://*BrokerMachineName*:*BrokerPort*/be/*BE-APPLICATION-NAME/ARCHIVE_ID/BE_HttpPortName*

Also when BusinessEvents component is deployed with TIBCO Enterprise Administrator, the URL is:

http://BrokerMachineName:BrokerPort/BE-COMPONENT-NAME/BE-APPLICATION-PATH.NAME/ARCHIVE_ID/BE_HttpPortName

Retrieving Log Files

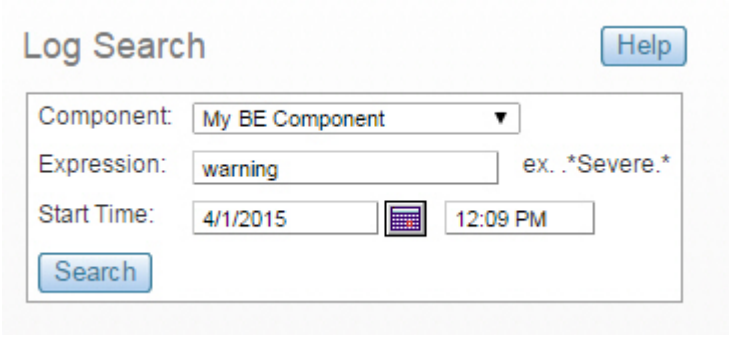
You can retrieve TIBCO Administrator and TIBCO BusinessEvents log files from the TIBCO Silver® Fabric Administration Tool.

In TIBCO Silver® Fabric Administration Tool, select **Engines** > **Log Search** .

Procedure

1. Select the component from which you want to see the log files, as shown in Log Files figure.
2. Optionally, you can search for a regular expression using the **Expression** field.
3. Select the start time to see the logs since that time.

Log Files



Retained Log Files

In addition to the engine log file, the TIBCO Hawk Agent log file and TIBCO BusinessEvents Log File are retained when BusinessEvents component is working with BusinessEvents Deployment with TIBCO Administrator mode:

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. The Hawk® log files do not open in the TIBCO Silver® Fabric Administrator GUI.

For both TIBCO Administrator components and TIBCO BusinessEvents components, TIBCO Silver Fabric Enabler for TIBCO BusinessEvents uses TIBCO Hawk® and TIBCO Hawk® Agent. The Retained TIBCO Hawk Agent Log Files table lists the retained TIBCO Hawk® log files.

Retained TIBCO Hawk® Agent Log Files

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

TIBCO BusinessEvents Deployment Log Files

The Retained TIBCO BusinessEvents Log Files table lists the retained TIBCO BusinessEvents Log Files.

Retained TIBCO BusinessEvents Log Files

Name	Location	Purpose of the Log
`ApplicationManagement.log`	`$ENGINE_WORK_DIR/domaindata/tra/`*`TIBCO_DOMAIN`*`/logs`	Generated by Appmanage, which deploys TIBCO BusinessEvents Applications.
`domainutility.log`	`$ENGINE_WORK_DIR/tibco/tra/`*`tra_version_2digits`*`/logs`	The Log file of the **`domainUtilitty`** command used to create a domain or add a machine. This file is common for all engines.
TIBCO BusinessEvents 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 TIBCO BusinessEvents and trace all activities that have happened.

For BusinessEvents component running in deployment with TIBCO Enterprise Administrator mode, only the TIBCO BusinessEvents Application Instance log files are retained.

Retained TIBCO BusinessEvents Log Files for BusinessEvents in deployment with TIBCO Enterprise Administrator mode

Name	Location	Purpose of the Log
TIBCO BusinessEvents Application Instance Name, for example, instance0.log	`$ENGINE_WORK_DIR/domaindata/logs`	These are the most important log files, as they are the log files from TIBCO BusinessEvents and trace all activities that have happened.