

TIBCO Silver[®] Fabric Enabler for Adapter for LDAP

User's Guide

*Software Release 3.0
March 2016*



Two-Second Advantage[®]

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO Silver, TIBCO Silver Fabric, TIBCO ActiveMatrix Adapter for LDAP, TIBCO Rendezvous, TIBCO Administrator, TIBCO Enterprise Message Service, TIBCO InConcert, TIBCO Policy Manager, TIBCO Runtime Agent, and TIBCO Hawk are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

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

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

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

Copyright © 2012 -2016 TIBCO Software Inc. All rights reserved.

TIBCO Software Inc. Confidential Information

Contents

Figures	v
Preface	vii
Related Documentation	viii
TIBCO Silver Fabric Enabler for Adapter for LDAP Documentation	viii
Other TIBCO Product Documentation	viii
Typographical Conventions	ix
Connecting with TIBCO Resources	xi
How to Join TIBCOCommunity	xi
How to Access All TIBCO Documentation	xi
How to Contact TIBCO Support	xi
Chapter 1 Introduction	1
Product Overview	2
Main Functionalities	2
Components	3
Chapter 2 Creating a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack	4
Overview	5
Creating TIBCO Silver Fabric Enabler for Adapter for LDAP Components	6
Changing the Component Enabler	20
Creating a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack	21
Dependency Requirements	23
TIBCO Silver Fabric Engine Statistics	26
Running TIBCO Silver Fabric Enabler for Adapter for LDAP Stack	31
Updating TIBCO Silver Fabric Enabler for Adapter for LDAP Stack	32
Chapter 3 Ant Scripts	55
Chapter 4 Log Files	63
TIBCO® Adapter for LDAP Log Files	64
Retained Log Files	65

Index 67

Figures

Figure 1	Name and Describe the Component	7
Figure 2	Choose TIBCO Product Distribution Versions	7
Figure 3	Choose Optional Distributions	8
Figure 4	Adding a Runtime Context Variable	16
Figure 5	Editing a Runtime Context Variable	17
Figure 6	Edit the Configuration File page	18
Figure 7	Creating a Stack.	21
Figure 8	Stack Builder Page.	22
Figure 9	Creating a new Threshold Activation Rule	27
Figure 10	Running a stack	31
Figure 11	Log Files.	64

Preface

Topics

- [Related Documentation, page viii](#)
- [Typographical Conventions, page ix](#)
- [Connecting with TIBCO Resources, page xi](#)

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Silver Fabric Enabler for Adapter for LDAP Documentation

The following documents form the TIBCO Silver Fabric Enabler for Adapter for LDAP documentation set:

- *TIBCO Silver[®] Fabric Enabler for Adapter for LDAP Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO Silver[®] Fabric Enabler for Adapter for LDAP User's Guide* Read this manual for instructions on using the product.
- *TIBCO Silver[®] Fabric Enabler for Adapter for LDAP Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

TIBCO Silver Fabric Enabler for Adapter for LDAP empowers TIBCO Silver[®] Fabric private cloud infrastructure to run TIBCO[®] Adapter for LDAP[®].

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

- TIBCO Silver[®] Fabric
- TIBCO ActiveMatrix[®] Adapter for LDAP[®]
- TIBCO Designer[™]
- TIBCO Administrator[™]
- TIBCO Rendezvous[®]
- TIBCO Hawk[®]
- TIBCO Runtime Agent[™]
- TIBCO Enterprise Message Service[™]




Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
<i>TIBCO_HOME</i>	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as <i>TIBCO_HOME</i> . The default value of <i>TIBCO_HOME</i> depends on the operating system. For example, on Windows systems, the default value is C:\tibco.
<i>ENV_HOME</i>	
<i>SFLD_HOME</i>	
<i>SILVERFABRIC_HOME</i>	
	Other TIBCO products are installed into an <i>installation environment</i> . Incompatible products and multiple instances of the same product are installed into different installation environments. An environment home directory is referenced in documentation as <i>ENV_HOME</i> . The default value of <i>ENV_HOME</i> depends on the operating system. For example, on Windows systems, the default value is C:\tibco.
	TIBCO Silver Fabric Enabler for Adapter for LDAP is installed into a directory that is referenced in documentation as <i>SFLD_HOME</i> . The value of <i>SFLD_HOME</i> depends on the operating system. For example, on Windows systems, the default value is C:\tibco\sfl
	TIBCO Silver Fabric is installed into a directory that is referenced in documentation as <i>SILVERFABRIC_HOME</i> . The value of <i>SILVERFABRIC_HOME</i> depends on the operating system. For example, on Windows systems, the default value can be C:\fabric
code font	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use MyCommand to start the foo process.
bold code font	Bold code font is used in the following ways: <ul style="list-style-type: none"> • In procedures, to indicate what a user types. For example: Type admin. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [enable disable]

Table 1 General Typographical Conventions (Continued)

Convention	Use
<i>italic font</i>	<p>Italic font is used in the following ways:</p> <ul style="list-style-type: none">• To indicate a document title. For example: See TIBCO® Adapter for LDAP <i>Concepts</i>.• To introduce new terms. For example: A portal page may contain several portlets. <i>Portlets</i> are mini-applications that run in a portal.• To indicate a variable in a command or code syntax that you must replace. For example: <code>MyCommand <i>PathName</i></code>
Key combinations	<p>Key names separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C.</p> <p>Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.</p>
	<p>The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.</p>
	<p>The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.</p>
	<p>The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.</p>

Connecting with TIBCO Resources

How to Join TIBCOCommunity

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

How to Access All TIBCO Documentation

After you join TIBCOCommunity, you can access the documentation for all supported product versions here:

<http://docs.tibco.com>

How to Contact TIBCO Support

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

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

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

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

<https://support.tibco.com>

Access to this site is restricted to designated customer contacts, but you can request a user login when you have a valid maintenance or support contract.

Chapter 1 **Introduction**

This chapter briefly introduces this product.

Topics

- [Product Overview, page 2](#)

Product Overview

TIBCO Silver® Fabric Enabler for Adapter for LDAP is a complementary software component. It publishes TIBCO ActiveMatrix® Adapter for LDAP projects in cloud environments based on TIBCO Silver® Fabric and leverage Silver Fabric capabilities. This accelerates publishing of TIBCO ActiveMatrix® Adapter for LDAP projects, enforces its industry best practices, and provides elastic optimization of computing resources.

Main Functionalities

TIBCO Silver Fabric Enabler for Adapter for LDAP provides the following main functionality:

- It enables you to quickly set up an environment for TIBCO ActiveMatrix® Adapter for LDAP on multiple machines. A TIBCO administrator can then publish TIBCO ActiveMatrix® Adapter for LDAP projects onto this environment using traditional tools, such as the TIBCO Administrator User Interface or its command-line tools.
- It enables you to quickly define, set up, and publish a complete stack based on TIBCO® Adapter for LDAP projects onto a set of virtual or physical machines. These actions include installation of this software, creation of TIBCO Domain, starting up TIBCO Administrator Server, and publishing of the TIBCO® Adapter for LDAP projects on one or multiple machines.
- It ensures that all publishing follows a set of recommended and supported TIBCO practices to implement fault tolerance, load balancing, software updates, and stack updates.
- It collects statistics from TIBCO® ActiveMatrix Adapter for LDAP and reports them to the Silver Fabric Broker for automated scaling of engines based on rules and threshold action levels. It scales up and down the number of engines required to process the workload. Therefore, it provides elasticity and optimization of computing resources.
- Supports Archive Scaling: Gathers and reports statistics from Adapter for LDAP service instances published by TIBCO Silver® Fabric to support automated rule-based scaling from archive statistics.

Components

TIBCO Silver Fabric Enabler for Adapter for LDAP consists of the following components:

- TIBCO® Adapter for LDAP Enabler which is used to configure, start, and manage TIBCO® Adapter for LDAP engines and its published archives.
- A set of distributions that includes all the software pieces required to run a TIBCO® Adapter for LDAP environment: TIBCO Runtime Agent, TIBCO Administrator, TIBCO Rendezvous®, TIBCO Hawk®, and TIBCO® Adapter for LDAP.

Chapter 2

Creating a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack

This chapter explains how to configure and publish TIBCO Silver Fabric Enabler for Adapter for LDAP stack.

Topics

- [Overview, page 5](#)
- [Creating TIBCO Silver Fabric Enabler for Adapter for LDAP Components, page 6](#)
- [Creating a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack, page 21](#)
- [Dependency Requirements, page 23](#)
- [TIBCO Silver Fabric Engine Statistics, page 26](#)
- [Running TIBCO Silver Fabric Enabler for Adapter for LDAP Stack, page 31](#)
- [Updating TIBCO Silver Fabric Enabler for Adapter for LDAP Stack, page 32](#)

Overview

A TIBCO Silver Fabric Enabler for Adapter for LDAP Stack is an entity that runs inside TIBCO Silver® Fabric. It executes all the components necessary to publish the TIBCO® Adapter for LDAP platform and to publish TIBCO® Adapter for LDAP projects.

A TIBCO Silver Fabric Enabler for Adapter for LDAP Stack consists of a TIBCO Administrator component, and one or more TIBCO® Adapter for LDAP components.

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

To build and run a TIBCO Silver Fabric Enabler for Adapter for LDAP components , you must perform the following tasks:

- Create and publish one or more TIBCO® Adapter for LDAP components. Refer to [Creating TIBCO Silver Fabric Enabler for Adapter for LDAP Components on page 6](#).
- Create a TIBCO Silver Fabric Enabler for Adapter for LDAP stack. Refer to [Creating a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack on page 21](#).
- Set a dependency to the TIBCO Administrator component for each TIBCO® Adapter for LDAP Components. Refer to [Dependency Requirements on page 23](#).
- Optionally set rules for TIBCO Silver® Fabric Engines. Refer to [TIBCO Silver Fabric Engine Statistics on page 26](#).

After completing these tasks, you can run and update a TIBCO Silver Fabric Enabler for Adapter for LDAP stack. See page [31](#) for information on how to run and update a stack.

Creating TIBCO Silver Fabric Enabler for Adapter for LDAP Components

This component performs the following tasks:

- Add a virtual machine to TIBCO Domain.
- Start TIBCO Hawk[®] Agent.
- Publish one or more Adapter for LDAP projects. This task is optional. If you do not upload the Adapter for LDAP project, you still can publish the Adapter for LDAP projects using the TIBCO administrator GUI or with the REST service.

To configure the TIBCO[®] Adapter for LDAP Component, perform these tasks:

- [Task A, Specifying the Component Name, page 6](#)
- [Task B, Select the Software Version, page 7](#)
- [Task C, Set the TIBCO ActiveMatrix Adapter for LDAP Basic Configuration, page 8](#)
- [Task D, Upload a Project \(optional\), page 12](#)
- [Task E, Hawk Application Management Interface \(AMI\) Configuration \(optional\), page 14](#)
- [Task F, Configure TIBCO Hawk Agent Running Condition, page 15](#)
- [Task G, Uploading Hawk MicroAgent Plugin \(optional\), page 15](#)
- [Task H, Add or Edit Runtime Context Variables \(optional\), page 16](#)
- [Task I, Upload, Add, Customize, or Remove a Content File, page 17](#)
- [Task J, Remove or Add Adapter for LDAP Component Statistics, page 17](#)
- [Task K, Edit Configuration File Screen, page 17](#)
- [Task L, Finish Configuring the Component, page 19](#)

Task A Specifying the Component Name

1. Using the TIBCO Silver[®] Fabric Administration Tool, select **Stacks > Components**.
2. On the Components page, select **Create New TIBCO ActiveMatrix Adapter for LDAP Component** in the Global Actions list.
3. Provide a name and description for the Component.

Figure 1 Name and Describe the Component

TIBCO ActiveMatrix Adapter for LDAP: My_LDAP_Component

Configure general properties

Name My_LDAP_Component

Description My AMX Adapter for LDAP

Cancel Previous Menu Next Finish

Task B Select the Software Version

You can select the version of the distribution you want TIBCO® Adapter for LDAP to run, as shown in [Figure 2](#).

You can select any version of the distributions installed in the Silver® Fabric Broker. By default, the latest versions of the distributions are displayed.

Figure 2 Choose TIBCO Product Distribution Versions

TIBCO ActiveMatrix Adapter for LDAP: My_LDAP_Component

Choose TIBCO Product Distribution Version:

TIBCO Product Version

TIBCO_ActiveMatrix_AdLDAP_distribution 6.1.1.0.0

TIBCO_TRA_distribution 5.7.4.0.0

TIBCO_HAWK_distribution 4.9.0.0

TIBCO_RV_distribution 8.4.0.2.0

Cancel Previous Menu Next Finish

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

Figure 3 Choose Optional Distributions

- You must choose Optional Distribution(s) only if your TIBCO Administrator implementation uses TIBCO Enterprise Messaging Service™ (EMS). You must then upload the latest TIBCO EMS distribution to the TIBCO Silver Fabric Broker.
- There is no EMS client embedded in TIBCO Runtime Agent since version 5.9.x. If you use EMS as a transport, you must install the EMS distribution and then select an EMS distribution version in the optional dependency screen.

Task C Set the TIBCO ActiveMatrix Adapter for LDAP Basic Configuration

The TIBCO ActiveMatrix Adapter for LDAP Basic Configuration page allows for definition of the JDBC driver, setting a specific TIBCO Domain machine name (the TIBCO Logical

Machine name - TLM), provides for specification of Fast TLM Restart, and allows uploading of external jar files and prepending/appendixing the path in the ClassPath. Each of the settings are described as below:

1. Upload a JDBC Driver (optional)

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

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



Each Adapter for LDAP component is dependent on TIBCO Administrator. Refer to [Dependency Requirements on page 23](#) for setting that dependency.

If the Domain is not stored in a database, then don't upload a JDBC driver.

2. 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 whatever reason.

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

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

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

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



If the TIBCO Domain machine name field is left blank the component name is used for setting the domain machine name.

When the component instance number is 0, the TLM machine name should be `<Your_TLM_Name>`.

3. Status of TIBCO Services after TLM restart

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

4. Specify Drive For Fast TLM Restart Configurations

Fast TIBCO Logical Machine Restart provides for accelerated restart of the engine so that many archives can be re-published within a reduced amount of time. Enhanced restart times are achieved by using a saved state stored on a shared Network File System drive instead of synchronizing with the domain repository. To set your expectations properly, "fast" does not mean

instantaneous or even amazingly fast, but it is faster than if the archives were loaded from the domain repository.

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

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



Recommendation: For those implementations that use fewer than ten Enabler for Adapter for LDAP deployments per component, avoid using Fast TLM restart to avoid the constraints mentioned in the preceding section.



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

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

5. 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 checked, 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 will be lost. So if you check this check box and deploy applications through REST call, the applications will be deleted and all the data will be lost after the component is restarted.

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

6. 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 check 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.

7. Force Kill of Adapter for LDAP Process by Engine Daemon at Shutdown

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



Use this option only if regular stop does not work. It kills all the TIBCO adapter for LDAP processes on shutdown and stops all the processes that were started by the engine daemon.

8. Add JAR Files before Enabler for Adapter for LDAP Classpath

Add external JAR file(s) to the Adapter for LDAP Component: As an option, JAR files can be uploaded for publishing with the TIBCO Silver Fabric Enabler for Adapter for LDAP component.

The check box *"Add the JAR file(s) in front of the ClassPath (check), otherwise at the end of the ClassPath (unchecked)"* means just what it says. The JAR file name can be prepended in front of the ClassPath by checking the box and if it is left unchecked the JAR file name is appended at the end of the ClassPath. Of course, this setting does not apply if a JAR file is not uploaded for addition to the ClassPath.

9. Upload JAR file(s) in ZIP format to the Adapter for LDAP ClassPath

Upload individual JAR file(s) 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.

Task D Upload a Project (optional)

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

1. Click the **Add** button in the Upload, Remove, or Recorder Archive Files panel
2. Click the **Browse** button in the Upload A File panel to navigate to the EAR or ZIP file and click the **OK** button.

You can upload one of two types of archive files:

- EAR File

When you upload a Adapter for LDAP EAR file, the deployment uses the default value of the global variables set in TIBCO Designer.

- ZIP File

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

To create an XML file, properties file for the project or for an archive for continuous deployment, perform the following steps:

- a. In `TIBCO_HOME/tra/tra_version/bin`, run the following command:
`AppManage -export -ear EarFile.ear -out DeploymentConfig.xml`
- b. Edit the file and set the value for the deployment.
- c. Create a ZIP file with the file `EarFile.ear` and `DeploymentConfig.xml`.



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/cc**".

When the deployment path is not specified, the deployed .ear/application files can be found in the root folder level of the Application Management in

TIBCO Administrator. When the EAR application is deployed to the adapter for LDAP 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.

These files can also be uploaded separately and then deployed, undeployed, started, and stopped by the HTTP REST request. For more information, refer to [Deploying Archives Directly to Endpoints - Continuous Deployment](#).

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

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

Task E Hawk Application Management Interface (AMI) Configuration (optional)

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

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 can be the same or different. By default, different TLMs on the same engine daemon (physical machine) are using the same RV transport (default AMI Hawk Service=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, then the service is reported in the hawkagent.cfg file as 6465. Then when the component is scaled up to other engine instances the AMI Hawk Service value is incremented higher by the enabler automatically.

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

Task F Configure TIBCO Hawk Agent Running Condition

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

Enter an integer to specify the number of seconds between periodic verification checks that the TIBCO Hawk Agent is still running.

If the TIBCO Hawk Agent becomes unresponsive to this verification then the process is automatically restarted.

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

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

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



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.

Task G Uploading Hawk MicroAgent Plugin (optional)

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

The .zip file gets extracted into the HMA plugin directory. If the directory is not empty, select one of the following options:

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

Task H Add or Edit Runtime Context Variables (optional)

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

Select a variable type from the **Add Variable** drop-down list or edit a variable.

Click the **Add Variable** selector and select the variable type from the list: String, Environment, System, or Encrypted.

Figure 4 Adding a Runtime Context Variable

TIBCO ActiveMatrix Adapter for LDAP: My_LDAP_Component

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

-- Add Variable --

-- Add Variable --

String

Environment

System

Encrypted

Add from EnablerEditRemove

	Value	Type	Description	Export	Auto Increment	Overridden	Contain
Y_DIRECTORY	\$(ENGINE_VWORK_DIR)/domaindata/archives	Environment	File archives will be copied to this location.	False	None		True
TIBCO_DOMAIN_DATA_TMP_DIRECTORY	\$(ENGINE_VWORK_DIR)/domaindata/tmp	Environment	File archives will be unpacked to this location.	False	None		True

CancelPreviousMenuNextFinish

Variable values from an enabler can be added to the runtime as well. Use the **Add from Enabler** button to add container-specific context variables.

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

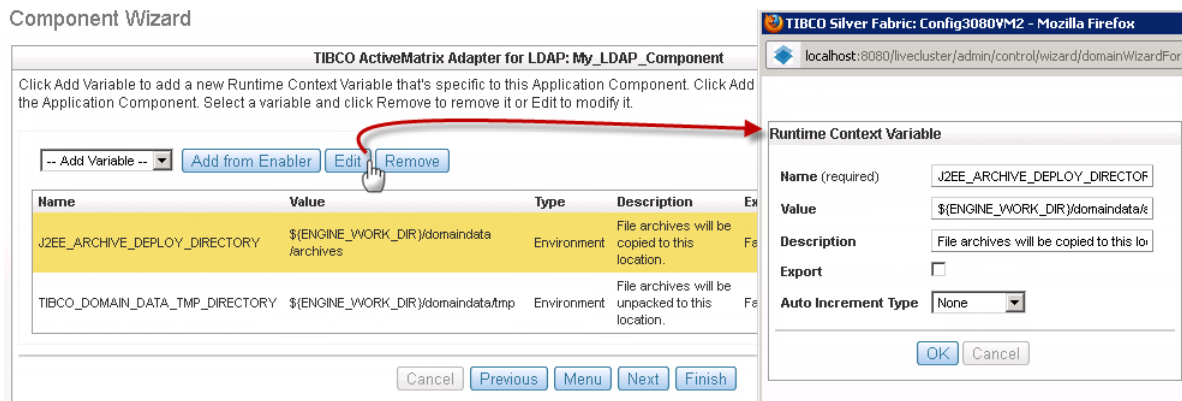
ARCHIVE_DETECTION_FREQUENCY is the periodic interval (the default value is 30 seconds) for detection of deployed archives and report to the broker. The broker uses this data to synchronize the archive with scaling rules.

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.

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

Figure 5 Editing a Runtime Context Variable



Task I Upload, Add, Customize, or Remove a Content File

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

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

Task J Remove or Add Adapter for LDAP Component Statistics

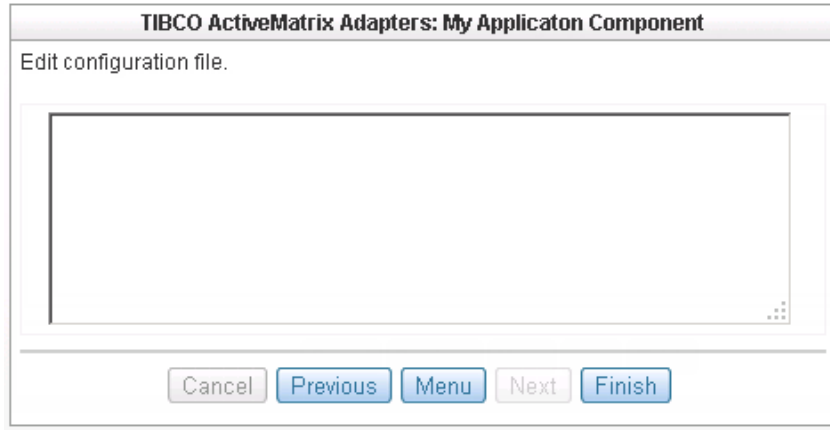
You can see the list of component runtime statistics that are tracked when the Adapter for LDAP Component is published and running. You can remove or later add back those statistics according to your implementation needs.

Task K 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 is used for more advanced customizations and normally it should be left alone.

For more information on what and how the `configure.xml` can be changed refer to "The configure.xml File" section in the *TIBCO Silver® Fabric Developer's Guide*.

Figure 6 Edit the Configuration File page



WARNING! Changes to the configuration.xml can break the installation. Before making any changes to the configuration.xml please back it up and secure it. Please consult an expert to ensure that any distribution changes are properly made.

As an example, if you wanted 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 would be `${TIBCO_HOME}/tra/5.8/template/domainutility`

The property, **include**, in <configFiles> element is used to specify which file(s) are to be replaced. It can specify whatever files you want to change. The asterisk wild card can be used to represent a string of characters like for instance: `"*.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.

Task L Finish Configuring the Component

Almost all of the other screens are generic for TIBCO Silver[®] Fabric Enablers.

These final configurations are optional for TIBCO[®] Adapter for LDAP Components. Refer to *TIBCO Silver[®] Fabric User's Guide* for more information on these configuration screens.

Click the **Finish** button 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** list located at the line of the component you just created.

Changing the Component Enabler

Upgrading a component created with TIBCO ActiveMatrix® Adapter for LDAP release 2.0 to 3.0 is easy.



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

1. Using the TIBCO Silver Fabric Administrator > Components page identify the Enabler for Adapter for LDAP component you wish to upgrade from release 2.0.x or 3.0.x. The **Enabler Version** column helps to identify out-of-date enablers.

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.

2. Click the component **Actions** menu icon again, click **Publish Changes**, and then click **OK** to publish the selected component.
3. Switch to the **Engines** page and those stacks that used the upgraded component appears in red if they require a component restart.

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

Creating a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack

A TIBCO Silver Fabric Enabler for Adapter for LDAP Stack must contain a single TIBCO Administrator component and one or more TIBCO® Adapter for LDAP Components. But a Stack can also be created with a single component and it could be run in standalone configurations. After creating and publishing all these components, you can create a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack.

Each *TIBCO_DOMAIN* has one TIBCO Silver Fabric Enabler for Adapter for LDAP Stack. After starting a stack, you can update it by adding or removing TIBCO® Adapter for LDAP Components.

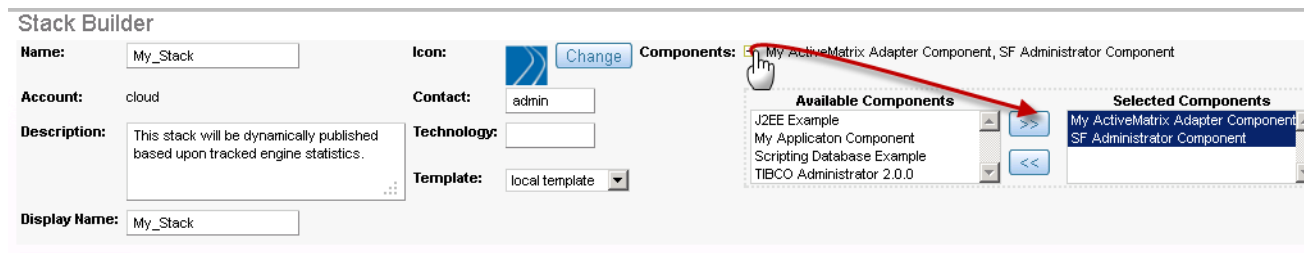
To create a TIBCO Silver Fabric Enabler for Adapter for LDAP Stack:

1. In the TIBCO Silver® Fabric Administration Tool, select **Stacks > Stacks**.
2. use the **Create New Stack** button.
3. Enter a stack name in the Stack Builder page as shown in Figure [Creating a Stack](#).

In the Components area:

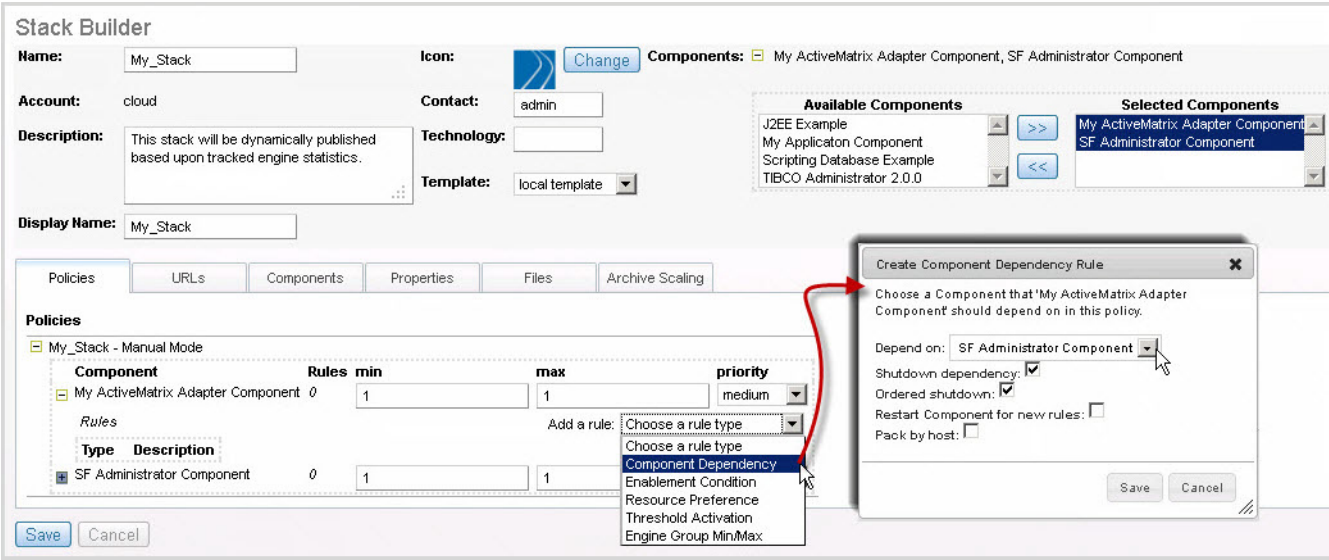
- a. Add one TIBCO Administrator component
- b. Add one or more TIBCO® Adapter for LDAP components.

Figure 7 Creating a Stack.



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

Figure 8 Stack Builder Page



Dependency Requirements

Each TIBCO® Adapter for LDAP Component, must have a Component Dependency set on one TIBCO Administrator component. You must set this Component Dependency for each of your TIBCO® Adapter for LDAP components. Those components without defined Component Dependencies do not have enough information for proper publishing.



This step is required. If you do not set the dependency, TIBCO Silver Fabric Enabler for Adapter for LDAP does not work.

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

To set the Component Dependency on the Administrator, follow these steps:

1. During creation or edit of the TIBCO® Adapter for LDAP Component, select **Add/edit default rule settings** from the Menu of the Component Wizard.
2. Use the **Add Rule** pull down to select the **Component Dependency** option.

Refer to the TIBCO Silver Fabric documentation for more details on the definition and use of Threshold Activation rules.

3. In the **Depend on** field, select the name of the TIBCO Administrator component that runs inside your Stack.



If you run TIBCO Administrator in the Fault Tolerant mode, clear the **Shutdown Dependency** check box. Otherwise, all TIBCO® Adapter for LDAP Components might stop if TIBCO Administrator stops working.

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 TIBCO EMS Server Component.

You can set the component dependency policy from the stack too.

Shutdown Dependency

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

If you use a Database domain, enabler for adapter for LDAP continues to work. However, every change in TIBCO Administrator is not taken into account by Enabler for Adapter for LDAP even after exporting the changes.

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

Ordered Shutdown

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

Restart Component for new rules

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

Pack by Host

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

For more information on all these setting, refer *TIBCO Silver Fabric User's Guide*.

TIBCO Silver Fabric Engine Statistics

If you want TIBCO® Adapter for LDAP Components to scale (add new engines) automatically, you can define rules that add or remove TIBCO Silver® Fabric engines based on engine statistics or Adapter for LDAP component statistics.

Data collected are aggregated. The aggregate is used to average raw statistic values by using a source ID. The average is calculated by individually averaging the statistic values for each source ID (for 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, then it does not add an engine. Removing engines is not affected by variance.

You can set up Enablement Condition rules. The engine starts when conditions specified are met by statistical values reported by other engines.

You also can setup rules on threshold activation, which is a statistic on the engine itself or other engines.

Task A Set Threshold Activation and Enablement Conditions for a Stack

1. In the Policies area of the TIBCO Silver Fabric Enabler for Adapter for LDAP Stack page, select the component you want to set up rules.
2. Select **Threshold Activation** or **Enablement Condition** in the Add A Rule list.

Figure 9 Creating a new Threshold Activation Rule

Stack Builder

Name: My_Stack **Icon:** (none) **Change** **Components:** SF Administrator Component, My ActiveMatrix Adapter Component, My ActiveMatrix Adapter Component

Account: cloud **Contact:** admin

Description: This Stack will be dynamically published based upon tracked engine statistics. **Technology:** **Template:** not template

Display Name:

Policies | URLs | Components | Properties | Files | Archive Scaling

My_Stack - Manual Mode

Component	Rules	min	max	priority
SF Administrator Component	0	1	1	medium
My ActiveMatrix Adapter Component	2	1	20	medium

Rules

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

Create Threshold Activation Rule

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

Condition type: Asset Manager Status, **Component:** My Application Component, **Statistic:** Free Memory, **Comparison:** Less than, **Value:** 1000.0 KB, **Sampling Window:** 120 seconds, **Action:** Add Engine

Save **Cancel**

Refer to the TIBCO Silver Fabric documentation for more details on the definition and use of the Threshold Activation rules.

3. If you select **Threshold Activation**, specify the following parameters in the Create Threshold Activation Rule panel:
 - In Condition Type list: select the **Component Statistic** item.
 - In the Action list, select **Add Engine** or **Remove Engine**.
When the threshold rule is set to add an engine, the Max engine of the LDAP component must be set to more than 1, or the rule do not work.
 - In the Component list, select the component where the statistic rules apply.
 - In the Statistics list, select the statistics property on which the activation is based.
 - In the Comparison list, select the operator such as: **Greater Than** or **Less Than**.
 - In the Value field, set the value of the measure that serves as the threshold or defining line that triggers the action when criteria is met.
 - In the 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.

Statistics consist of engine and machine measures that are independent of TIBCO® Adapter for LDAP and any other statistical information gathered from the Adapter for LDAP via TIBCO Hawk® when that is applicable.

You can see the statistics that are tracked by the component created to publish the Adapter for LDAP by selecting that component in the rule creation window.

Create Archive Scaling Rules

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

Create Archive Scaling Rules to automate creation or removal of new Adapter for LDAP Component Archives when rules based on archive statistics meet or exceed thresholds or conditions you have set.

After adding components to your stack you can create new scaling rules by opening the **Archive Scaling** tab and clicking the **Create New** button.

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.

The **Archives** tab defines what archive is added or removed according to the rules you define in the other tabs.

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.

Select the Adapter for LDAP component that was used to upload the application archive file and use the **Archive Names** field to specify what archive is subject to the rules you set using the other tabs of the Archive Scaling Rule Editor.

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.

Select the statistics, the operator, the value, and the sampling window period in seconds to define your condition for adding a new archive instance.



The Sampling Window must be a sufficiently large time period (in seconds) to collect the aggregated statistics. If the Sampling Window is not big enough there is a chance that no statistics is reported in a particular time-frame creating an inadvertent trigger condition.

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

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

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.

The **Remove Archive Conditions** tab enables you to release computing resources and remove unused or idle Component Archives or process instances to scale down your Component Archives just as you scaled them up according to conditions you define on usage statistics.

With 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 would on the Add Archive Conditions tab and further restrict where the new Archive instances can start depending on component Instances that have:

Same Component: works all the time for Component Archive scaling.

From the set of Components: works only if your Component Archive is compatible with the set of components present. For example an Adapter for LDAP archive can 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 the Component Archive runs on machines with the appropriate middle ware: TIBCO Adapter for LDAP, TIBCO TRA, TIBCO Hawk, etc.

Same Enabler and Middleware Version: this selection ensures that your Component Archive scales up successfully, but it is the least restrictive of the target component conditions. .



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

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

Running TIBCO Silver Fabric Enabler for Adapter for LDAP Stack

After you have created your stack, publish it. Then click **Run Stack In Manual Mode** in the Actions drop-down list as shown in [Figure 10](#).

Figure 10 Running a stack



If you selected a policy schedule while creating an stack, you can run the stack in the Auto mode. The stack runs given the schedule defined. For more information on creating and running stacks, refer to the TIBCO Silver Fabric documentation.

Updating TIBCO Silver Fabric Enabler for Adapter for LDAP Stack

When a stack is published and running, you can still make changes to the stack such as adding other components, changing allocation rules, changing threshold activation rules, or deploying and starting archives on the runtime Adapter for LDAP 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 may require restart of the changed resource, so consult the TIBCO Silver Fabric documentation for best practices prior to making changes to a production system.

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.

If you want to change an TIBCO® Adapter for LDAP component, stop and restart your entire stack. To deploy, start, stop, and undeploy TIBCO® Adapter for LDAP project archives, use the following ways:

Micro-scaling: Start and stop TIBCO® Adapter for LDAP archives based on your defined rules when they are already in your component. For more information, refer to: [Create Archive Scaling Rules on page 28](#)

Continuous Deployment (deploy archives directly to Adapter for LDAP endpoints) - publish (deploy), unpublish (undeploy), start, or stop Adapter for LDAP archives without having to change any stacks, components, or Adapter for LDAP Adapter for LDAP engines. Deploying Adapter for LDAP application archives via REST and cURL commands is described in the next section.

Deploying Archives Directly to Endpoints - Continuous Deployment

There are situations where deploying archives directly to a running TIBCO Adapter for LDAP 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 Adapter for LDAP instances already running on Silver Fabric Engines using the command line 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 via 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 good detail in the section on *Deploying Archives Directly to Components in the TIBCO Silver Fabric Administration Guide*.

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

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

Continuous Deployment Life Cycle

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

- **Deploy:** Send an archive to a Silver Fabric Engine running an Adapter for LDAP component that meets the criteria specified. The Deploy REST method enables specification of a properties files with criteria dictating where and how the archive should be deployed.
- **Start:** Start an archive that was deployed to an engine.
- **Stop:** Stop an archive that is running on an engine.
- **Undeploy:** Remove an archive from an engine, stopping any running instances of that archive on that engine.

Deploy

Adapter for LDAP application archives can 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/eng
ines/archives"
[-F "AppName=YourDirABC/YourAppName"]
[-F "AppSettings.element1.element2=SomeValue"]
[-F "ArchiveSettings.element1.element2=SomeValue"]
[-F "Archives=Archive_A,Archive_B,Archive_X"]
[-F "configurationFile=YourConfigurationfile.xml"]
[-F "ForceDeploy=true"]
[-F "GV=globalVariableA=123,globalVarB=SomeString"]
[-F "InstanceSettings.element1.element2=SomeValue"]
[-F "NoDeploy=true"]
[-F "NoStart=true"]
[-F "PUID=PUID_Value"]
[-F "VariableProvider=Some_PROVIDER_Name"]

```

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



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

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

Example 1 An example cURL archives deploy statement:

```

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

```

Where the user specified by -u must have Silver Fabric administrator level permissions, and the form-data fields (-F "property=value") can be specified in any order. The form-data fields (-F) are described as follows:

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

archiveFile: specifies your Adapter for LDAP archive file (.zip, .par, 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 by specifying that list with the **Archives** form-data field. You can also upload archives using the Component Wizard.

deploymentFile: specifies the properties file that defines endpoint selection criteria described in more detail as follows:

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

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

-v: specifies the target of the cURL POST execution and asks for a verbose response. The cURL `-v` expression should specify the appropriate Silver Fabric directory. For the default installation that expression 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.

AppName: specifies the directory location where the application archive(s) is deployed and what the application is named.

Where your archive application is deployed 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 the 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 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.

Do not specify the optional form-data fields, unless you want to change the default behavior. By default all archives in the archive file are deployed and started unless an archive of the same name is already deployed and started, in which case that archive can run without interruption or replacement.

AppSettings: (optional) specifies settings that your application uses when deployed.

All deployment configuration cURL expressions take the form:

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

Some examples:

```
-F "AppSettings.localRepoInstance.encoding=UTF-8"
```

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

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

- a. **description:** A string describing the application.
- b. **contact:** A string to name the person responsible for the deployment.
- c. **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.
- d. **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.

- e. **encoding:** Specifies encoding for the repository instance. If this element is not specified, then the encoding for the admin server is used. If the admin server is not available, then 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 zip that are to be deployed. If an **archives** list is omitted then all archives in the application archive package are deployed. Example:

-F "Archives=Archive_A,Archive_B,Archive_X"

ArchiveSettings: (optional) form-data parameter specifies settings for the archive.

- a. **enabled:** *true* or *false*. Only enabled services are deployed. Disabling a service, effectively undeploys just that service while letting all other services in the application run as normal. This can be useful when you wish to deploy an application that includes a service for which you don't have the required software. A deployment configuration cURL expressions takes the form:

-F "ArchiveSettings.enabled=true"

- b. **av:** Specify values for archive runtime variables with a comma-separated string with each key value pair joined by an equals (=) sign. For example:-

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



Refer to the Appendix of the *TIBCO Runtime Agent Scripting Deployment User's Guide* for a full list of Archive Settings properties, parameters, descriptions, and usage. Not all elements and properties are supported for use by the TIBCO ActiveMatrix® Adapter for LDAP. The following is a first attempt at listing them all.

- c. **hbInterval**: heartbeat interval. The heartbeat interval determines the time (in milliseconds) between heartbeat messages.
- d. **activationInterval**: activation interval. This field specifies the amount of time to expire since the last heartbeat from the master before the secondary restarts the process starters and process engines.
- e. **preparationDelay**: preparation interval. This field is used to specify a delay before the master engine restarts.
- f. **ruleBases**: rule bases for the archive

`ruleBase.uri`: location of the rulebase file. Example syntax:

```
-F "ArchiveSettings.ruleBases.ruleBase.uri=someValue"
```

`ruleBase.data`: Rulebase content. Do not change this setting.

- g. **failureEvents.failureEvent**:

`failureType`: *ANY, FIRST, SECOND, Subsequent*

If event is used then Type must be specified.

Example syntax:

```
-F
```

```
"ArchiveSettings.failureEvents.failureEvent.failureType=ANY"
```

`restart`: *true or false*. If true, the service instance is restarted upon failure.

`description`: information that describes this operation.

`alertAction.performPolicy`: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

`alertAction.enabled`: *true or false*. If true, the action occurs when conditions for the action are true. If false, the action is not called."

`alertAction.level`: *High, Medium, Low*

`alertAction.message`: The message that displays when this alert is triggered.

- h. **failureEvents.emailAction**:

`performPolicy`: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

`enabled`: *true or false*. If true, the action occurs when conditions for the action are true. If false, the action is not called.

`message`: the message to send.

`to`: a comma-separated list of email addresses to which the message is

sent.

cc: a comma-separated list of email addresses to which copies of the message are sent.

subject: the subject of the email message.

SMTPServer: The mail server (SMTP server) to use to send the message. Specify the host name or the host IP address.

i. **failureEvent.customAction:**

performPolicy: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

enabled: *true* or *false*. If true, the action occurs when conditions for the action are true. If false, the action is not called.

command: specify the script to execute. Script files are highly recommended.

arguments: the list of arguments for the command.

j. **suspendProcessEvents.suspendProcessEvent:**

restart: true or false. If true, the service instance is restarted upon failure.

description: information that describes this operation.

alertAction.performPolicy: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

alertAction.enabled: true or false. If true, the action occurs when conditions for the action are true. If false, the action is not called.

alertAction.level: High, Medium, Low

alertAction.message: The message that displays when this alert is triggered.

emailAction.performPolicy: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

emailAction.enabled: true or false. If true, the action occurs when conditions for the action are true. If false, the action is not called.

emailAction.message: the message to send.

emailAction.to: a comma-separated list of email addresses to which the message is sent.

emailAction.cc: a comma-separated list of email addresses to which copies of the message are sent.

emailAction.subject: the subject of the email message.

`emailAction.smtpServer`: The mail server (SMTP server) to use to send the message. Specify the host name or the host IP address.

`customAction.performPolicy`: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

`customAction.enabled`: true or false. If true, the action occurs when conditions for the action are true. If false, the action is not called.

`customAction.command`: specify the script to execute. Script files are highly recommended.

`customAction.arguments`: the list of arguments for the command.

k. **logEvents.logEvent**:

`restart`: *true* or *false*. If true, the service instance is restarted upon failure.

Example syntax:

```
-F "ArchiveSettings.logEvents.logEvent.restart=true"
```

`match`: The string in the log file to match.

`description`: information that describes this operation.

`alertAction.performPolicy`: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

`alertAction.enabled`: *true* or *false*. If true, the action occurs when conditions for the action are true. If false, the action is not called.

`alertAction.level`: *High*, *Medium*, *Low*

`alertAction.message`: The message that displays when this alert is triggered.

`emailAction.performPolicy`: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

`emailAction.enabled`: *true* or *false*. If true, the action occurs when conditions for the action are true. If false, the action is not called.

`emailAction.message`: the message to send.

`emailAction.to`: a comma-separated list of email addresses to which the message is sent.

`emailAction.cc`: a comma-separated list of email addresses to which copies of the message are sent.

`emailAction.subject`: the subject of the email message.

`emailAction.smtpServer`: The mail server (SMTP server) to use to send the message. Specify the host name or the host IP address.

`customAction.performPolicy`: the policy to perform. *Once* - Generates an alert only for the first occurrence. *Always* - Generates an alert for each occurrence.

`customAction.enabled`: *true* or *false*. If true, the action occurs when conditions for the action are true. If false, the action is not called.

`customAction.command`: specify the script to execute. Script files are highly recommended.

`customAction.arguments`: the list of arguments for the command

- l. **failureCount**: The value in this field defines how many restarts should be attempted before resetting the error counter to 0.
- m. **failureInterval**: The value in this field defines how much time should expire before resetting the error counter to 0.

InstanceSettings: (optional) Some syntax examples:

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

- a. **description:** Specify any pertinent information about the binding.
- b. **contact:** Name the person responsible for this application instance.
- c. **startOnBoot:** When *true* the service instance starts when the computer is restarted. Default value is *false*.
- d. **enableVerbose:** When *true*, sets the enabler for verbose tracking for service instances. Default value is *false*.
- e. **maxLogFileSize:** Sets the maximum size (in kilobytes) that a log file can reach before the engine switches to the next log file.
- f. **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.
- g. **threadCount:** Specifies the number of threads to use to execute process instances. The number of threads determines how many process instances can execute concurrently.
- h. **prepandClassPath:** The values supplied here are prepended to your CLASSPATH environment variable.
- i. **appendClassPath:** The items you supply here are appended to your CLASSPATH environment variable.
- j. **initHeapSize:** Specifies the initial size (in MB) for the JVM used for the process engine. Default is 32 MB.
- k. **maxHeapSize:** Specifies the maximum size (in MB) for the JVM used for the process engine. Default is 256 MB.
- l. **threadStackSize:** Specifies the size of the thread stack. Default is 256 KB.
- m. **runAsNT:** When set to *true* the service is run as a Microsoft Windows Service. You can then manage the engine as you would any other service, and you can specify that it starts automatically when the machine reboots.
- n. **startUpType:** Specifies the instance service startup type as either: *Automatic*, *Manual*, or *Disabled*.
- o. **login:** Specifies the login account for the service, if any. The domain name must be specified as well.
- p. **password:** Sets the password for that service, if any.
- q. **checkpoint:** When set to *true*, the process engine waits for all jobs to finish (up to the maximum timeout) before shutting down the engine, rather than removing jobs at their next checkpoint.
- r. **timeout:** The maximum timeout in seconds the process engine waits for jobs to finish before shutting down the engine. A zero (0) value means 0

seconds, which effectively turns the graceful shutdown into an immediate shutdown.

- s. **iv:** This element uses a comma-separated string with name-value pairs with each key value pair joined by an equals (=) 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 must 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 TIBCO Silver Fabric Enabler for Adapter forLDAP *Developer's Guide*.

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

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

GV: (optional) sets global variables for use on the targeted application endpoint by the archive. The GV form-data field lets you define a comma delimited list of declarative name equals value statements.

```
-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 then those values that you have set in the deployment `configurationFile.xml` are applied.

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

VariableProvider: (optional) Specifies a variable provider to set global variables for applications deployed with REST. The variable provider is a Java Class extension compiled into a JAR and loaded into a Silver Fabric directory with an appropriate XML so that it can be called by REST during application deployment.

-F "VariableProvider=*Some_PROVIDER_Name*"



Creating a variable provider for use by a REST call

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

NoDeploy: (optional) 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"

NoStart: (optional) 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"

Deployment File

When deploying an archive by REST you must include a deployment file, which specifies at least one selection criteria for determining which engine and component receives the deployed archive. The deployment file is a simple properties text file specified by a form-data field like the following for REST upload with the archive:

-F "deploymentFile=@YourDeploymentFileName.properties"

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

criteria comparator value

Some criteria require an argument to be specified in parentheses:

criteria(argument) comparator value

For example, the following is a simple statement to deploy an archive to an engine running a component with a Component Type that contains Adapter as part of the name and which has a Domain name of YourDomain:

ComponentType contains Adapter

ImportedVariable(TIBCO_DOMAIN_NAME) = YourDomain

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

Table 2 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.

Table 2 Supported Criteria

Name	Definition
EngineProperty(<i>name</i>)	<p>A named engine property. The following engine properties may be used:</p> <ul style="list-style-type: none">• id: The numeric ID unique to each engine, generated upon installation• guid: The globally unique identifier for the engine (network card address)• instance: The instance of the engine, starting with zero. Multi-CPU hosts may have multiple instances of Engines running concurrently• IP: The IP address of the engine• username: The username of the engine, which is the hostname of the engine unless installed with tracking properties• hostname: The hostname of the engine• cpuNo: The number of CPUs on the engine's host computer• cpuCoreCount: The total number of CPU cores on the engine's host computer, if available. A value of -1 indicates that a value could not be determined• cpuSocketCount: The total number of physical CPUs on the engine's host computer, if available. A value of -1 indicates that a value could not be determined• cpuThreadCount: The total number of hardware threads on the engine's host computer, if available. A value of -1 indicates that a value could not be determined• cpuIdString: The CPUID string of the first physical CPU on the Engine's host computer, if available• cpuTotal: The amount of processing power on the Engine's host computer, measured in millions of floating-point arithmetic operations per second. Calculated on the engine daemon using a Linpack performance calculation and reported when the engine daemon logs into the Manager and updated on a frequency that is set in the engine configuration

Table 2 Supported Criteria

Name	Definition
<code>EngineProperty(name)</code>	<ul style="list-style-type: none"> • <code>totalMemInKB</code>: The amount of total physical memory on the Engine's host computer, in kilobytes • <code>freeMemInKB</code>: The amount of free physical memory on the Engine's host computer, in kilobytes, updated when any Engine instance is launched • <code>freeDiskInMB</code>: The amount of available disk space on the Engine's host computer, in megabytes, updated when any Engine instance is launched • <code>os</code>: The operating system platform. This corresponds with the Engine installation. Available platforms can be viewed on the Engine Install page • <code>osVersion</code>: The version of the operating system • <code>homeDir</code>: Home directory, the installation directory on the Engine's host computer • <code>dataDir</code>: Data directory, the directory which is home to the DDT (direct data transfer) data • <code>dataUrl</code>: Data URL, the URL corresponding to the DDT directory • <code>workUrl</code>: Work URL, the URL of work directory on the Engine's fileserver that stores log and temporary files • <code>sharedDirID</code>: A number which identifies a group of Engines running from the same home directory • <code>osUsername</code>: The owner of the process running the Engine • <code>configurationName</code>: The current Engine Configuration name. The value is <code>[os]:[name]</code> like on the Engine Configuration page • <code>pid</code>: The process ID of the Engine process
<code>ActivationInfoProperty(name)</code>	A named property, such as <code>ClusterName</code> , or <code>HTTP_STATIC_ROUTE_PREFIX</code> within the Component's <code>ActivationInfo</code> object. Archives that can be scaled elastically keep a list of <code>ActivationInfo</code> properties and their respective values for failover Broker.
<code>ImportedVariable(name)</code>	A variable imported into a Component.

Table 2 Supported Criteria

Name	Definition
ExportedVariable(<i>name</i>)	A variable exported from a Component.
Statistic(<i>name</i>)	A named enabler for Adapter for LDAP statistic like: Total Memory, Free Memory, Free Disk, CPU Utilization, DS CPU Utilization, Expected Engine Count, Actual Engine Count, or Allocating Engine Count
ArchiveStatistic(<i>statName</i> , <i>e</i> , <i>archiveName</i>)	A named statistic for a specified archive.
DependencyComponent	A named dependency on a component.
DependencyEngine(<i>componentName</i>)	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 ActiveMatrix Adapter for LDAP:2.0.0"
Statistic(CPU Utilization) < 80
ActivationInfoProperty(ClusterName) matches dev_cluster.*
```

Successful Deployment

REST returns a Status of 200 (OK) when the archive is successfully deployed. A successful REST execution returns the Engine-Instance and Engine-Id where the archive deployed. Example response (application/JSON):

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

```
"status": 200
}
```

The Engine-Id, Engine-Instance, and the full ArchiveName invokes of START, STOP, and UNDEPLOY methods to enable full control of the Archive life cycle.

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

Continuous Deployment Timeout

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

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

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

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



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

Start

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

Example 2 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/{full_archive-name}/"
```

`{archive-id}/start"`

Expressions in the generic form cURL expression are 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.

Obtain the values of `{engine-id}` and `{engine-instance}` 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.

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

The `{full_archive-name}` can be copied from the TIBCO Silver Fabric Administrator > Dashboard > **Scaled Archives** page.



The `{full_archive-name}` 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 in appropriately.

Example 3 A cURL example that starts an Archive.

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

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

...to 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%2FSFFL%2
022E5%2FDomainMachineName%2FprocessOrder%2FOrders%2FMyProcOrder/Pr
ocess_Archive.par/start"
```

Stop

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


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

Refer to the [Start](#) method for a description on how to obtain the values of {*engine-id*}, {*engine-instance*}, and {*full_archive-name*}.

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/{full_archive-name}/{archive-id}/undeploy"
      -F "DeleteApp=true"
```

Refer to the [Start](#) method for a description on how to obtain the values of {*engine-id*}, {*engine-instance*}, and {*full_archive-name*}.

DeleteApp: (optional - for use with undeploy only) The default value is false and it can be omitted. When the DeleteApp parameter is false then 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 TIBCO Silver Fabric Enabler for Adapter for LDAP Administration Tool. For information on using REST services, see "Using REST Services" in the TIBCO Silver Fabric Enabler for Adapter for LDAP *Developer's Guide*.

The Archive Management Support Feature

When components are activated, by default all archives are deployed and 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 feature has an option that can be cleared to prevent archives from starting when the component is activated.

Chapter 3 **Ant Scripts**

In addition to the 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 which can be easier to use in some configuration scenarios, and also provides an easy way to configure Silver Fabric from within an IDE with a built-in Ant support.

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

Samples

The Silver Fabric Command Line Interface installation contains a samples directory, which contains example build files using Ant tasks for several Enablers. See the enclosed Readme for more information on the examples.

You can also create a sample build.xml and build.properties based on any of your published stacks. See "Packaging stacks For Ant Task Deployment" in the *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="sfld-build" default="release" basedir="." xmlns:sf="antlib:com.datasynapse.fabric.ant">

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

  <sf:connection-props brokerurl="http://lin64cdc201.qa.datasynapse.com:8000" username="admin" password="admin"
    clientssltrustfile="{DSSSLTrustFile}" />

  <target name="release" depends="release-component, release-stack"/>
  <target name="clean" depends="clean-stack, clean-component"/>
```

```

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

  <sf:option name="${component.name}" action="replace">
    <sf:property name="Department" value="${department}" />
    <sf:property name="Location" value="${location}" />
    <sf:property name="Partition" value="${partition}" />
    <sf:property name="Engine Blacklisting" value="false" />
    <sf:property name="Failures Per Day Before Blacklist" value="0" />
    <sf:property name="Archive Scale Up Timeout" value="${archive.scale.up.timeout}" />
    <sf:property name="Archive Scale Down Timeout" value="${archive.scale.down.timeout}" />
    <sf:property name="Maximum Deactivation Time" value="${deactivation.timeout}" />
    <sf:property name="Maximum Activation Time" value="${activation.timeout}" />
    <sf:property name="Maximum Capture Time" value="${maximum.capture.time}" />
    <sf:property name="Maximum Instances Per Host" value="${max.instances.per.host}" />
    <sf:property name="Separator Tags" value="${separator.tags}" />
    <sf:property name="Statistics Collection Frequency" value="${stats.collection.frequency}" />
    <sf:property name="Activation Delay" value="0" />
    <sf:property name="Engine Reservation Expiration" value="${engine.reservation.expiration}" />
  </sf:option>

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

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

  <sf:feature name="${component.name}" action="add" feature="HTTP Support" />
  <sf:feature name="${component.name}" action="update" feature="HTTP Support">
    <sf:property name="HTTP Enabled" value="${http.enabled}" />
    <sf:property name="HTTPS Enabled" value="${https.enabled}" />
    <sf:property name="Routing Prefix" value="${routing.prefix}" />
    <sf:property name="Route Directly To Endpoints" value="${routing.direct}" />
  </sf:feature>

  <sf:feature name="${component.name}" action="add" feature="Archive Management Support" />

```

```

<sf:feature name="${component.name}" action="update" feature="Archive Management Support">
  <sf:property name="Start Archives On Activation" value="true" />
</sf:feature>

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

<target name="clean-component">
  <echo message="Cleaning ${component.name}" />
  <sf:unpublish type="component" name="${component.name}" onerror="ignore" />
  <sf:remove type="component" name="${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="${component.name}" />
  <sf:stack-policy action="update" name="${stack.name}">
    <sf:policy manualpolicy="true">
      <sf:compalloc component="${component.name}" min="${min}" max="${max}" priority="${priority}">
        <sf:allocationrule type="Resource Preference">
          <sf:property name="propertyName" value="os" />
          <sf:property name="propertyValue" value="linux" />
          <sf:property name="operator" value="contains" />
          <sf:property name="affinityPos" value="2" />
        </sf:allocationrule>
      </sf:compalloc>
    </sf:policy>
  </sf:stack-policy>

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

<target name="clean-stack">
  <echo message="Cleaning ${stack.name}" />
  <sf:unpublish type="stack" name="${stack.name}" onerror="ignore" />
  <sf:remove type="stack" name="${stack.name}" onerror="ignore" />
</target>
</project>

```

build.properties

Following is a sample of build.properties file.

```

# stack properties
stack.name=Example Stack for ${component.name}
stack.description=Example Stack for ${component.name} built by Ant

# component properties
component.name=SFLD Example Component
component.description=SFLD Example Component built by Ant
component.type=TIBCO ActiveMatrix Enabler for Adapter for LDAP:3.1.0
enabler.name=TIBCO ActiveMatrix Enabler for Adapter for LDAP container
enabler.version=3.0.0.0
utility=false

# load balancing
routing.direct=false
routing.prefix=${cluster.name}

# logging
archive.logs=true
log.file.pattern=\
././domaindata/logs/domainutility.log,\
././domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/Hawk.log,\
././domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/tsm.log,\
././domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/msghma.log,\
././domaindata/tra/${TIBCO_DOMAIN_NAME}/logs/ApplicationManagement.log,\
././domaindata/tra/${TIBCO_DOMAIN_NAME}/application/logs/*.log
checkpoint.frequency=300

# instances
max.instances.per.host=1
min=1
max=4
priority=3

# http
http.enabled=true
https.enabled=true

# build directories
content.dir=content
configure.dir=configure
script.path=

# timeouts
archive.scale.up.timeout=120
archive.scale.down.timeout=120

```

```
activation.timeout=300  
deactivation.timeout=120  
maximum.capture.time=300  
stats.collection.frequency=120  
engine.reservation.expiration=300
```

```
# options  
department=  
location=  
partition=  
separator.tags=
```


Chapter 4 **Log Files**

This chapter introduces log files.

Topics

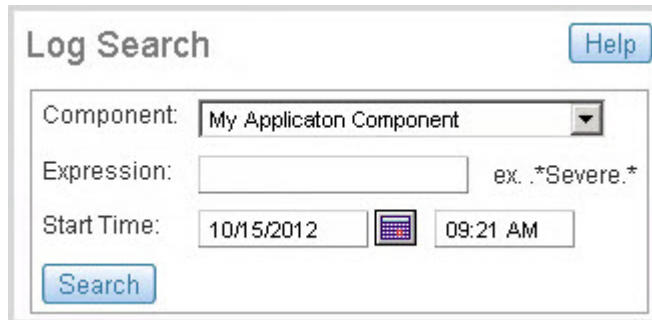
- [TIBCO® Adapter for LDAP Log Files, page 64](#)
- [Retained Log Files, page 65](#)

TIBCO® Adapter for LDAP Log Files

You can retrieve TIBCO Administrator and TIBCO® Adapter for LDAP log files from the TIBCO Silver® Fabric Administration Tool. To do so, follow these steps:

1. In TIBCO Silver® Fabric Administration Tool, select **Engines > Log Search**.
2. Select the component from which you want to see the log files, as shown in [Figure 11](#).
3. Optionally you can search for a regular expression using the **Expression** field.
4. Select the **Start Time** to see the logs since that time.

Figure 11 Log Files



The screenshot shows a 'Log Search' dialog box with a 'Help' button in the top right corner. Inside the dialog, there are three main input fields: 'Component', 'Expression', and 'Start Time'. The 'Component' field is a dropdown menu currently showing 'My Application Component'. The 'Expression' field is a text box with a sample value 'ex. .*Severe.*' to its right. The 'Start Time' field consists of a date box showing '10/15/2012', a calendar icon, and a time box showing '09:21 AM'. A 'Search' button is located at the bottom left of the dialog box.

Retained Log Files

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

- [TIBCO Hawk Agent Log Files, page 65](#)
- [TIBCO® Adapter for LDAP Log Files, page 65](#)
- [TIBCO® Adapter for LDAP Log Files, page 64](#)

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 appear in the TIBCO Silver® Fabric Administrator GUI.

For both TIBCO Administrator components and TIBCO® Adapter for LDAP Components, TIBCO Silver Fabric Enabler for Adapter for LDAP uses TIBCO Hawk® and TIBCO Hawk® Agent. [Table 3](#) lists the retained TIBCO Hawk® log files.

Table 3 Retained TIBCO Hawk® Agent Log Files

Name	Location	Purpose of the Log
Hawk.log	<code>\$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOM AIN/logs</code>	Log file of the Hawk call in the Hawk® Agent.
tsm.log	<code>\$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOM AIN/logs</code>	Log file of the Hawk® Agent.
msghma.log	<code>\$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOM AIN/logs</code>	Log file for tibhawkhma.

TIBCO® Adapter for LDAP Log Files

[Table 4](#) lists the retained TIBCO® Adapter for LDAP Log Files.

Table 4 Retained TIBCO® Adapter for LDAP Log Files

Name	Location	Purpose of the Log
ApplicationManagem ent.log	<code>\$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOMAIN/logs</code>	Generated by Appmanage, which publishes TIBCO® Adapter for LDAP Applications.

Table 4 Retained TIBCO® Adapter for LDAP Log Files

Name	Location	Purpose of the Log
domainutility.log	<i>\$ENGINE_WORK_DIR/tibco/tra/tra_version_2digits/logs</i>	Log file of the domainUtility command used to create a domain or add a machine. This file is common for all engines.
TIBCO® Adapter for LDAP Application Name, for example, OrderConsolidation-Order_Consolidation.log	<i>\$ENGINE_WORK_DIR/domaindata/tra/TIBCO_DOMAIN/application/logs</i>	These are the most important log files, as they are the log files from TIBCO® Adapter for LDAP and trace all activities that have happened.

Index

Symbols

32

A

Add Archive Conditions [29](#)
 Administrator Component
 Publish [19](#)

C

Component upgrade [20](#)
 Component, containers [3](#)
 Components [3](#)
 Components, creating [6](#)
 configure.xml [17](#)
 Configuring the Component, task list [6](#)
 Creating an application [21](#)
 customer support [xi](#)

D

Dependency Requirements [23](#)
 dependency, setting [23](#)
 distributions [3, 7](#)
 Documentation [viii, viii, xi](#)
 Domain Logical Machine Name [9](#)

E

EAR File [12](#)
 Enablement condition [27](#)
 entity [5](#)
 environment variable [16](#)

F

Fault Tolerant [23, 24](#)
 Functionalities [2](#)

L

Log Files [65](#)
 Log files [64](#)
 log files
 ApplicationManagement.log [65](#)
 domainutility.log [66](#)
 Hawk.log [65](#)
 msghma.log [65](#)
 tsm.log [65](#)
 Log Files, Hawk Agent [65](#)
 Log Files, retained [65](#)

P

Product_HOME [ix](#)

R

runtime specific context variables [16](#)

Z

ZIP File [12](#)

S

SILVERFABRIC_HOME [ix](#)

source ID [26](#)

Statistics [26](#)

support, contacting [xi](#)

system variable [16](#)

T

tasks, creating an application [5](#)

technical support [xi](#)

Threshold Activation [27](#)

TIBCO Adapter for LDAP Log files [65](#)

TIBCO Administrator Container [3](#)

TIBCO Domain Logical Machine Name [9](#)

TIBCO Silver Fabric Enabler, updating [32](#)

TIBCO Silver Fabric Engine [26](#)

TIBCO_HOME [ix](#)

TIBCommunity [xi](#)

TLM [9](#)

V

variable, environment [16](#)

variable, string [16](#)

variables, encrypted [16](#)

variables, System [16](#)

X

XML file [12](#), [12](#)