

# **TIBCO Silver<sup>®</sup> Fabric Enabler for Adapter for Database**

## **User's Guide**

*Software Release 3.0.0  
December 2015*

## 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 Database, TIBCO ActiveMatrix Enabler for Adapter for database, 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.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation 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 © 2011-2015 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

# Contents

<b>Figures</b> .....	<b>v</b>
<b>Tables</b> .....	<b>vii</b>
<b>Preface</b> .....	<b>ix</b>
Related Documentation .....	x
TIBCO Silver Fabric Enabler for Adapter for Database Documentation .....	x
Other TIBCO Product Documentation .....	x
Typographical Conventions .....	xi
Connecting with TIBCO Resources .....	xiii
How to Join TIBCOCommunity .....	xiii
How to Access All TIBCO Documentation .....	xiii
How to Contact TIBCO Support .....	xiii
<b>Chapter 1 Introduction</b> .....	<b>1</b>
Product Overview .....	2
Main Functionalities .....	2
Components .....	3
<b>Chapter 2 Creating a TIBCO Silver Fabric Enabler for Adapter for Database Stack</b> .....	<b>5</b>
Overview .....	6
Creating TIBCO Silver Fabric Enabler for Adapter for Database Components .....	7
Changing the Component Enabler .....	27
Creating a TIBCO Silver Fabric Enabler for Adapter for Database Stack .....	29
Dependency Requirements .....	31
Adapter for Database Statistics .....	34
Create Archive Scaling Rules .....	36
Running TIBCO Silver Fabric Enabler for Adapter for Database Stack .....	40
Updating TIBCO Silver Fabric Enabler for Adapter for Database Stack .....	41
Deploying Archives Directly to Endpoints - Continuous Deployment .....	41
Continuous Deployment Life Cycle .....	42

**Chapter 3 Ant Scripts..... 63**

**Chapter 4 Log Files ..... 68**

Adapter for Database Log Files ..... 69

Retained Log Files ..... 70

**Index ..... 73**

# Figures

Figure 1	TIBCO Silver Fabric Enabler for Adapter for Database Components	3
Figure 2	Name and Description of the Component	8
Figure 3	Choose TIBCO Product Distribution Versions	9
Figure 4	Optional Distribution	9
Figure 5	Upload a JDBC Driver for TIBCO Domain Storage	10
Figure 6	Configure the TIBCO Domain Machine Name (Optional)	11
Figure 7	Upload an ODBC.ini File	14
Figure 8	Upload a JDBC Driver for Adapter for Database	14
Figure 9	Uploading Archives	15
Figure 10	TIBCO Hawk AMI Configuration	17
Figure 11	TIBCO Administrator and Hawk Agent Running Conditions	18
Figure 12	Upload Hawk MicroAgent Plugin	19
Figure 13	Editing a Variable	20
Figure 14	Add or Edit a String Variable	21
Figure 15	Upload, Add, Customize, or Remove Content Files	22
Figure 16	Customize a Selected Text File Using the Simple Text Editor	22
Figure 17	Add or Remove Adapter for Database Component Runtime Statistics	23
Figure 18	Edit the Configuration File Page	24
Figure 19	Upgrading a Component - Change Enabler	27
Figure 20	Restart the Component	28
Figure 21	Creating a Stack	29
Figure 22	Stack Builder Page	30
Figure 23	Setting Administrator Component Dependency	31
Figure 24	Creating Rules	35
Figure 25	Statistics Available with the Silver Fabric Enabler for Adapter for Database Component	36
Figure 26	Creating a New Archive Scaling Rule	37
Figure 27	Define Archive Scaling Rule Conditions	38
Figure 28	Running a Stack	40

Figure 29 Log Files ..... 69

# Tables

Table 1	General Typographical Conventions . . . . .	xi
Table 2	Supported Criteria . . . . .	53
Table 3	Retained TIBCO Hawk® Agent Log Files . . . . .	70
Table 4	Retained TIBCO Administrator Log Files . . . . .	71
Table 5	Retained TIBCO Adapter for Database Log Files . . . . .	71





# Preface

## Topics

---

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

## Related Documentation

---

This section lists documentation resources you may find useful.

### TIBCO Silver Fabric Enabler for Adapter for Database Documentation

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

- *TIBCO Silver<sup>®</sup> Fabric Enabler for Adapter for Database Installation*  
Read this manual for instructions on site preparation and installation.
- *TIBCO Silver<sup>®</sup> Fabric Enabler for Adapter for Database User's Guide*  
Read this manual for instructions on using the product.
- *TIBCO Silver<sup>®</sup> Fabric Enabler for Adapter for Database 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 Database empowers TIBCO Silver<sup>®</sup> Fabric private cloud infrastructure to run TIBCO Adapter for Database.

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

- TIBCO Silver<sup>®</sup> Fabric
- TIBCO ActiveMatrix<sup>®</sup> Adapter for Database
- TIBCO Designer<sup>™</sup>
- TIBCO Administrator<sup>™</sup>
- TIBCO Rendezvous<sup>®</sup>
- TIBCO Hawk<sup>®</sup>
- TIBCO Runtime Agent<sup>™</sup>
- TIBCO Enterprise Message Service<sup>™</sup>
- TIBCO Adapter<sup>™</sup> SDK




## Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
<i>TIBCO_HOME</i>	<p>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.</p> <p>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.</p> <p>TIBCO Silver Fabric Enabler for Adapter for Database is installed into a directory that is referenced in documentation as <i>SFDB_HOME</i>. The value of <i>SFDB_HOME</i> depends on the operating system. For example, on Windows systems, the default value is C:\tibco\sfdb</p> <p>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.</p>
<i>ENV_HOME</i>	
<i>SFDB_HOME</i>	
<i>SILVERFABRIC_HOME</i>	
code font	<p>Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example:</p> <p>Use MyCommand to start the foo process.</p>
<b>bold code font</b>	<p>Bold code font is used in the following ways:</p> <ul style="list-style-type: none"> <li>• In procedures, to indicate what a user types. For example: Type <b>admin</b>.</li> <li>• In large code samples, to indicate the parts of the sample that are of particular interest.</li> <li>• In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [<b>enable</b>   disable]</li> </ul>

Table 1 General Typographical Conventions (Continued)

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

## 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

All product documentation is available from <https://docs.tibco.com>

Silver Fabric Enabler for Adapter for Database documentation is here:

<https://docs.tibco.com/products/tibco-silver-fabric-enabler-for-adapter-for-database-3-0-0>

### 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](#)
- [Main Functionalities, page 2](#)
- [Components, page 3](#)

## Product Overview

---

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

### Main Functionalities

TIBCO Silver Fabric Enabler for Adapter for Database provides the following main functionalities:

- You can quickly set up an environment on multiple machines. A TIBCO administrator can then publish Adapter for Database projects onto this environment using traditional tools, such as the TIBCO Administrator User Interface or its command-line tools.
- You can quickly define, set up, and publish a complete stack based on Adapter for Database projects onto a set of virtual or physical machines. These actions include the installation of this software, creation of TIBCO domain, starting up the TIBCO Administrator server, and publishing of the Adapter for Database projects on one or multiple machines.
- You can ensure that publishing follows a set of recommended and supported TIBCO practices to implement load balancing, software updates, and application updates.
- It collects Adapter for Database metrics from Adapter for Database engines to be used in TIBCO Silver® Fabric rules. It scales up and down the number of Adapter for Database engines required to process the workload. Therefore, it provides elasticity and optimization of computing resources.
- You can create a pool of TIBCO Domain Machines (TLM) for scaling up and scaling down TIBCO Silver Fabric Enabler for Adapter for Database Engines, Component Archives, and Adapters.
- Gathers and reports statistics from each TIBCO ActiveMatrix Adapter for Database service instance published by TIBCO Silver® Fabric to support automated rule-based scaling from archive statistics.
- Supports fast TLM restart for engines from shared drives. Planned TLM restarts are streamlined so that large numbers of applications with multiple archives each can be restarted more quickly.

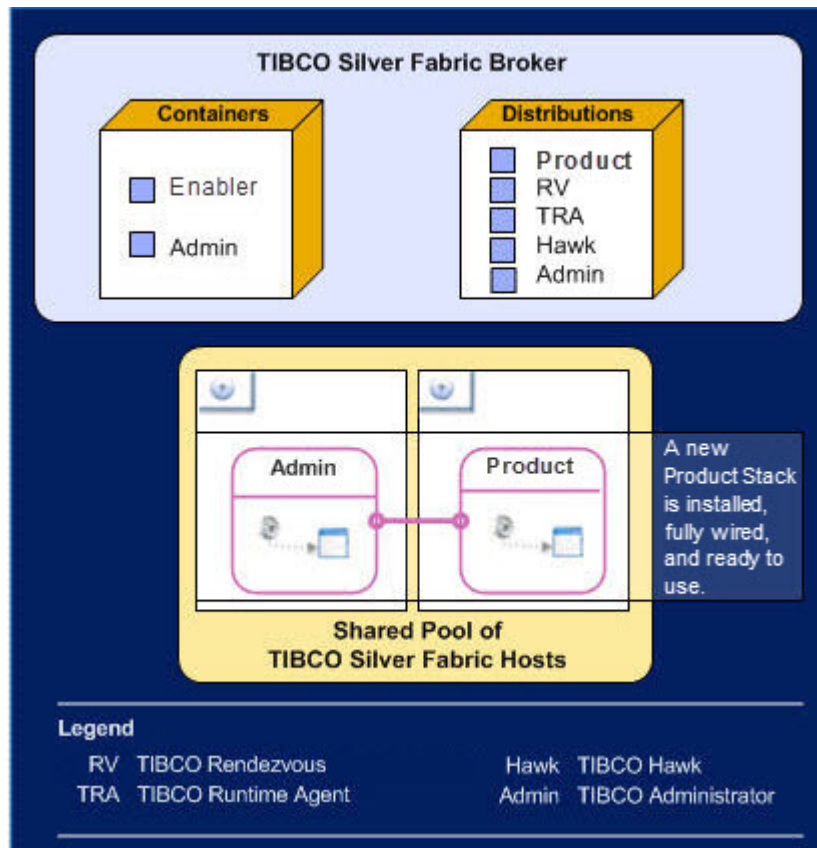


## Components

TIBCO Silver Fabric Enabler for Adapter for Database consists of the following components, as shown in [Figure 1, TIBCO Silver Fabric Enabler for Adapter for Database Components](#):

- TIBCO Adapter for Database Container  
It is used to configure, start, and manage Adapter for Database engines.
- A set of distributions that includes all the software pieces required to run an Adapter for Database environment: TIBCO Runtime Agent, TIBCO Administrator, TIBCO Rendezvous®, TIBCO Hawk®, and TIBCO Adapter for Database.

*Figure 1 TIBCO Silver Fabric Enabler for Adapter for Database Components*





## Chapter 2

# Creating a TIBCO Silver Fabric Enabler for Adapter for Database Stack

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

## Topics

---

- [Overview, page 6](#)
- [Creating TIBCO Silver Fabric Enabler for Adapter for Database Components, page 7](#)
- [Changing the Component Enabler, page 27](#)
- [Creating a TIBCO Silver Fabric Enabler for Adapter for Database Stack, page 29](#)
- [Dependency Requirements, page 31](#)
- [Adapter for Database Statistics, page 34](#)
- [Running TIBCO Silver Fabric Enabler for Adapter for Database Stack, page 40](#)
- [Updating TIBCO Silver Fabric Enabler for Adapter for Database Stack, page 41](#)
- [Adapter for Database Log Files, page 69](#)
- [Retained Log Files, page 70](#)

## Overview

---

TIBCO Silver Fabric Enabler for Adapter for Database is an entity that runs inside TIBCO Silver<sup>®</sup> Fabric. It facilitates component configuration and executes all the components necessary for publishing the Adapter for Database platform and to publish Adapter for Database projects.

A TIBCO Silver Fabric Enabler for Adapter for Database stack consists of a TIBCO Administrator component, and one or more Adapter for Database components.

To build and run a TIBCO Silver Fabric Enabler for Adapter for Database stack, perform the following tasks:

- Create and publish one or more Adapter for Database components. Refer to [Creating TIBCO Silver Fabric Enabler for Adapter for Database Components on page 7](#).
- Create a TIBCO Silver Fabric Enabler for Adapter for Database stack. Refer to [Creating a TIBCO Silver Fabric Enabler for Adapter for Database Stack on page 29](#).
- Set a dependency to the TIBCO Administrator Component for each Adapter for Database components. Refer to [Dependency Requirements on page 31](#).
- Optionally set rules for TIBCO Silver<sup>®</sup> Fabric Engines. Refer to [Adapter for Database Statistics on page 34](#).

After completing these tasks, you can run and update a TIBCO Silver Fabric Enabler for Adapter for Database stack. See page [40](#) for information on how to run and update a stack.

# Creating TIBCO Silver Fabric Enabler for Adapter for Database Components

---

This component performs the following tasks:

- Add a virtual machine to the TIBCO domain.
- Start TIBCO Hawk<sup>®</sup> Agent.
- Publish one or more Adapter for Database projects. This task is optional.

If you do not upload the Adapter for Database projects, you still can publish the Adapter for Database projects on the machine where, Adapter for Database registers to the TIBCO domain. This is done using the TIBCO Administrator GUI or deploying it with the REST service.

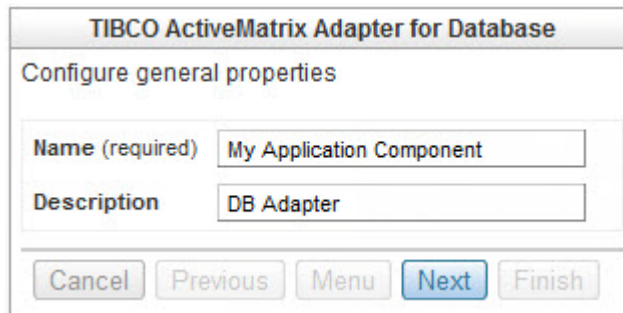
To configure the Adapter for Database Component, perform the following tasks:

- [Task A, Specify Component Name, page 8](#)
- [Task B, Select the Software Version, page 9](#)
- [Task C, Set Adapter for Database Basic Configuration, page 10](#)
- [Task D, Upload a Project, page 15](#)
- [Task E, Hawk Application Management Interface \(AMI\) Configuration \(optional\), page 17](#)
- [Task F, Configure TIBCO Hawk Agent Running Condition, page 18](#)
- [Task G, Uploading Hawk MicroAgent Plugin \(optional\), page 18](#)
- [Task H, Add or Edit Runtime Context Variables \(optional\), page 20](#)
- [Task I, Adding a String, Environment, System, or Encrypted Variable, page 20](#)
- [Task J, Add, Override, Customize Container and Component-Specific Content Files, page 21](#)
- [Task K, Add or Remove Adapter for Database Statistics, page 23](#)
- [Task L, Edit Configuration File Screen, page 24](#)
- [Task M, Finish Configuring the Component, page 26](#)

### Task A Specify Component Name

1. Using the TIBCO Silver<sup>®</sup> Fabric Administration Tool, select **Stacks > Components**.
2. On the components page, select **Create New TIBCO ActiveMatrix Adapter for Database Component** in the **Global Actions** list.
3. If you have multiple versions of the Adapter for Database Container, a dialog is displayed, where you can select the container version to create a new component. Select the container version and click **OK**.
4. Provide a name and description for the component.

Figure 2 Name and Description of the Component



The screenshot shows a dialog box titled "TIBCO ActiveMatrix Adapter for Database". Below the title bar, it says "Configure general properties". There are two text input fields: "Name (required)" with the value "My Application Component" and "Description" with the value "DB Adapter". At the bottom, there are five buttons: "Cancel", "Previous", "Menu", "Next" (which is highlighted in blue), and "Finish".

## Task B Select the Software Version

You can select the version of the distribution you want TIBCO Adapter for Database to run, as shown in [Figure 3](#).

You can select any version of the distributions installed in the Silver<sup>®</sup> Fabric Broker. By default, the latest versions of the distributions are displayed. All versions of the distributions are compatible.

*Figure 3 Choose TIBCO Product Distribution Versions*

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

*Figure 4 Optional Distribution*



- 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.

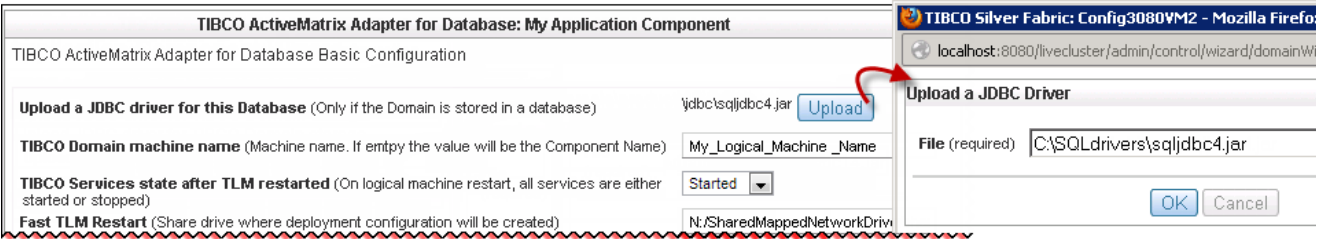
**Task C Set Adapter for Database Basic Configuration**

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

**1. Upload a JDBC Driver for this Database**

Specify a JDBC driver to publish with the TIBCO® Adapter for Database 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

*Figure 5 Upload a JDBC Driver for TIBCO Domain Storage*



Each Adapter for Database Component has dependency on TIBCO Administrator. Refer to [Dependency Requirements on page 31](#) to learn how to set the dependency.



## 2. Set the Domain Machine Name (optional)

The TIBCO Domain Machine (also known as the TIBCO Logical Machine - TLM) provides for publishing of quickly identifiable components. If the TIBCO domain name is not set, the component name is used.

The TIBCO Domain Machine virtualizes the machine so that component publishing can maintain state when the targeted engine is changed for whatever reason. When the component is instantiated more than one time, a component instance number is appended to the TIBCO domain name to identify the individual instances.

For example, if the target machine restarts because of an OS update or a hardware change, TLM restarts the virtualized machine on different hardware.



When the TIBCO Domain Machine name is set, the component .ear (.JAR) files are republished and started on the new virtual machine hardware.

When Adapter for Database Applications have been previously published through TIBCO Administrator (or through the AppManage commandline interface) those previous deployments are also published again and restarted on the new hardware.

*Figure 6 Configure the TIBCO Domain Machine Name (Optional)*

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

When the component instantiates multiple times, the TLM machine is named as follows: `<Your_TLM_Name>_<ComponentInstanceNumber>`, where the `<ComponentInstanceNumber>` is 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. Specify Drive for Fast TLM Restart Configurations.

Fast TIBCO Logical Machine restart provides for accelerated restart of the Engine and redeploying of many archives within a reduced amount of time. Enhanced restart times are achieved by using a saved state stored on a shared Network File System (NFS) drive instead of synchronizing with the domain repository. It updates the deployment configuration file with the new binary location. 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. The degree to which you can guarantee reliability and availability of that shared drive will help ensure runtime continuity.
- 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 database 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.

### 4. Status of TIBCO Services after TLM Restart.

Sets the desired services state when the TIBCO logical machine is restarted. When the TLM is restarted enabler for adapter for database service instances will be republished and either **started** or **stopped**. The stopped services state setting may be convenient for developers who are testing machines with many services that don't need to be started for every logical machine change.

#### 5. Delete Application Configuration at Shutdown

Select this option to undeploy and remove all the applications deployed on the TIBCO Logical Machine.

#### 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.

#### 7. Force Kill of Enabler for Adapter for database Process by Engine Daemon at Shutdown

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



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

#### 8. Add JAR Files before Enabler for Adapter for database Classpath

**Add external JAR file(s) to the Enabler for Adapter for database Component** - As an option, JAR files may be uploaded for publishing with the TIBCO Enabler for Adapter for database component.

The JAR file name may 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.

#### Upload JAR file(s) in ZIP Format to Enabler for Adapter for database ClassPath -

Upload individual JAR file(s) to be either appended or prepended to the ClassPath. All uploaded JARs will be added in the same way according to how the checkbox is set.

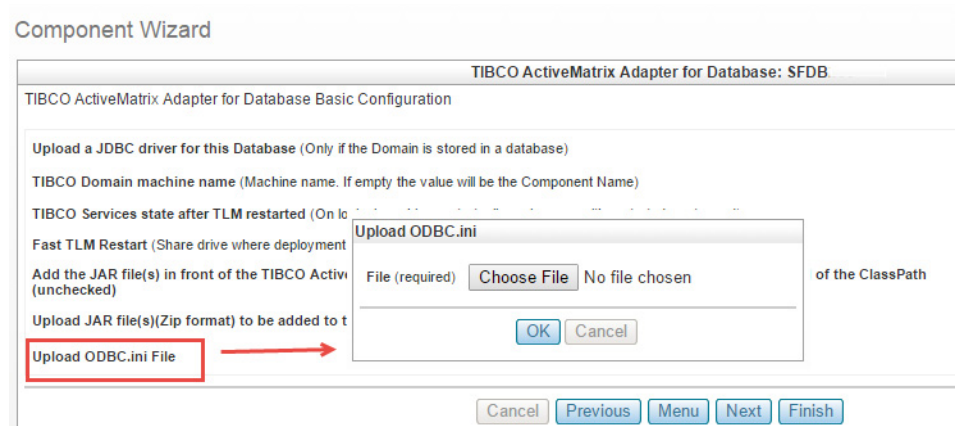


To remove unwanted JAR files use the Menu button to display the list of Wizard configuration steps and select **Add/Override/Customize Enabler and Component-specific content files**. Relative paths to external jar files added may be removed with that window.

9. Upload ODBC.ini

Upload **ODBC.ini** file in the component wizard, if you are creating components using 6.x distributions. You need the `odbc.ini` file for the ear application.

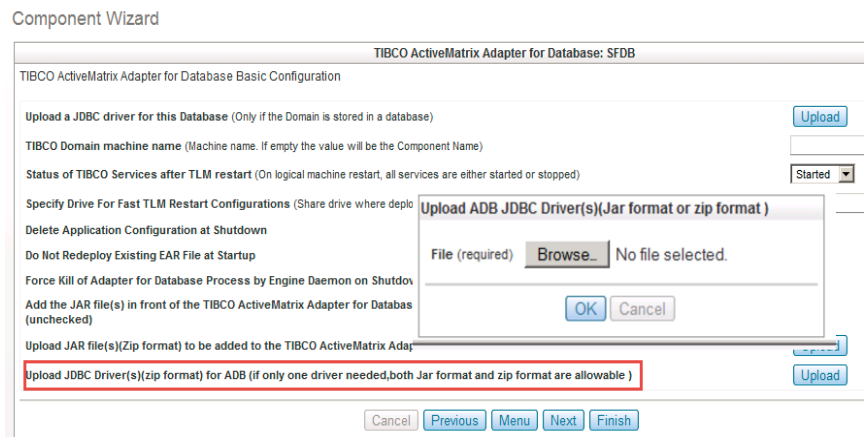
Figure 7 Upload an ODBC.ini File



10. Upload JDBC File

When you select 7.x distribution, only then the option of JDBC upload is available on the component wizard.

Figure 8 Upload a JDBC Driver for Adapter for Database



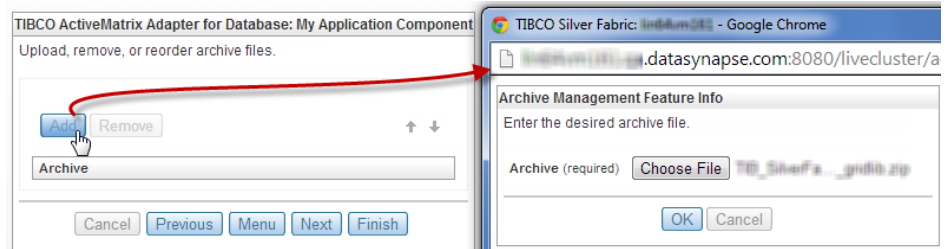
**Upload a JDBC driver for the Database** option is used for communication with the domain.

## Task D Upload a Project

If you want TIBCO Silver Fabric Enabler for Adapter for Database Applications to publish and run one or more Adapter for Database projects, upload one or more archive files (\*.ear or \*.zip files) as follows:

1. Click the **Add** button in the Upload, Remove, or Reorder Archive Files panel, as shown in [Figure 9](#).

Figure 9 Uploading Archives



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

You can upload one of following types of archive files:

- EAR file

When you upload an Adapter for Database .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 contains all the deployment configurations, deployment path name, global variables, and so on.

To create an XML 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 compressed archive can specify the deployment path. For example, create the `CustomFolder.properties` file with the content "**ApplicationFullPath=aaa/bbb/cc**".

You can also add a .cfg file that contains a list of value pair `<property>=<value>`, where the `<property>` is the global variable name

and the <value> is the value of global variable. These values are used to substitute the global variables at deployment.

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 database adapter 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 *TIBCO Runtime Agent documentation and Scripting Deployment User's Guide*. Alternatively, you can also deploy applications from TIBCO Administrator or use 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.

Figure 10 TIBCO Hawk AMI Configuration

**AMI Hawk Service:** specifies the TIBCO Hawk port number, for example, TIBCO Rendezvous connects with TIBCO Hawk on the default port 7475. AMI Hawk Service and AMI Hawk Daemon must be set with valid values. 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 may be the same or different. By default, different TLMs on the same engine daemon (physical machine) use the same RV transport (default AMI Hawk Service=7475, and default AMI Hawk daemon=tcp:7474). This setting enables visibility of all of the deployed applications on each TLM even if some applications are not deployed on this particular TLM.



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

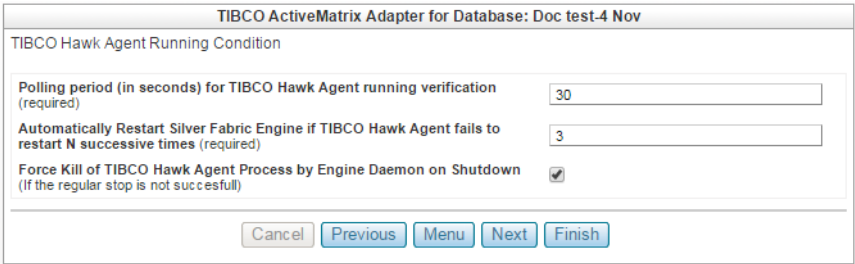
Generally, AMI Hawk Network is an empty string, and all three fields can be left empty. If all three fields are left empty, the enabler uses the default value "7475" and "tcp:7474". But if a value is set for either AMI Hawk service or AMI Hawk daemon, they must both be set with valid values. Setting only one of them results in an error.

**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, the process is automatically restarted.

*Figure 11   TIBCO Administrator and Hawk Agent Running Conditions*

The screenshot shows a configuration window titled "TIBCO ActiveMatrix Adapter for Database: Doc test-4 Nov". Inside, the "TIBCO Hawk Agent Running Condition" section contains three settings: "Polling period (in seconds) for TIBCO Hawk Agent running verification (required)" with a value of 30, "Automatically Restart Silver Fabric Engine if TIBCO Hawk Agent fails to restart N successive times (required)" with a value of 3, and "Force Kill of TIBCO Hawk Agent Process by Engine Daemon on Shutdown (If the regular stop is not successful)" which is checked. At the bottom are buttons for "Cancel", "Previous", "Menu", "Next", and "Finish".

- **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**

*Figure 12 Upload Hawk MicroAgent Plugin*

TIBCO ActiveMatrix Adapter for Database: Doc test-4 Nov

Upload Hawk MicroAgent plugin (optional)

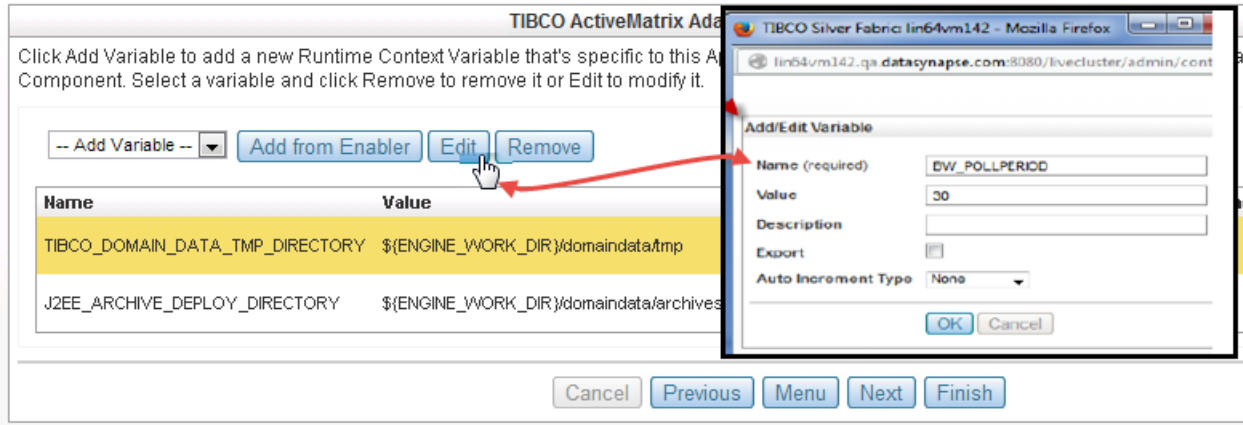
Upload HMA plugin file(s) to be executed by HawkAgent (zip format will be automatically unzipped)

Action to perform if the HMA plugin directory is not empty

## Task H Add or Edit Runtime Context Variables (optional)

**String, Environment, System, or Encrypted** variables may be added to the component to define and set runtime-specific context variables. Select a variable type from the **Add Variable** drop-down to add or edit a variable.

Figure 13 Editing a Variable



Variable values from the container may be changed as well. Select the **Add from Container** button to change values of 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.

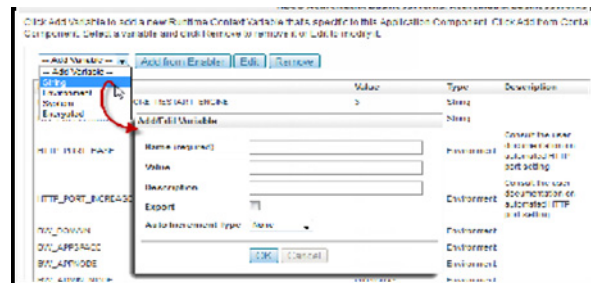
## Task I Adding a String, Environment, System, or Encrypted Variable

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

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

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

Figure 14 Add or Edit a String Variable



2. Click **OK**.

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

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

## Task J Add, Override, Customize Container and Component-Specific Content Files

Content files may be uploaded, added from a container, edited with a simple text editor, or removed using the "Add/Override/Customize Container and Component-specific content files" page.

Figure 15 Upload, Add, Customize, or Remove Content Files.

TIBCO ActiveMatrix Adapter for Database: My Application Component

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

UploadAdd from EnablerCustomizeRemove

Relative Path	Name	Overridden Enabler File
odbc\	odbc.ini	False

CancelPreviousMenuNextFinish



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, may be removed with that window.

Figure 16 Customize a Selected Text File Using the Simple Text Editor

TIBCO ActiveMatrix Adapter for Database: My Application C

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

UploadAdd from EnablerCustomizeRemove

Relative Path	Name	Overridden Enabler File
odbc\	odbc.ini	False

CancelPreviousMenuNextFinish

TIBCO Silver Fabric: SFBR-WIN - Windows Internet Explorer

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

Edit content file

ReportCodePageConversionErrors=0  
ReportRecycleBin=0  
ServerName=  
ServerType=0  
ServiceName=  
SID=ORCL  
TimestampEscapeMapping=0  
TNSNamesFile=tnsnames.ora  
TrustStore=  
TrustStorePassword=  
UseCurrentSchema=1  
ValidateServerCertificate=1  
WireProtocolMode=1  
[ODBC]  
IANAAppCodePage=4  
InstallDir=/tsi/tests/usr/sbabu/odbc  
Trace=0  
TraceFile=odbctrace.out  
TraceDll=/tsi/tests/usr/sbabu/odbc/lib/titrc26.so

OKCancel

## Task K Add or Remove Adapter for Database Statistics

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

Figure 17 Add or Remove Adapter for Database Component Runtime Statistics

**TIBCO ActiveMatrix Adapter for Database: NewAdb**

Add or remove statistics that you would like to track on this Component.

Name	Description
ADADB New Errors	Number of errors since the last call to this service
ADADB Component Uptime	Component up and running time since started
ADADB Messages Received Rate	Rate of TIBCO Rendezvous messages received.
ADADB Messages Sent Rate	Rate of TIBCO Rendezvous messages published.
ADADB Messages Received	Number of TIBCO Rendezvous messages received
ADADB Messages Sent	Number of TIBCO Rendezvous messages published
ADADB Total Errors	Total number of errors since startup
ADADB QueueCount	Current number of elements in the queue
ADADB MaxQueueSize	Maximum number of elements in the queue
ADADB Archive New Errors	Number of errors since the last call to this service
ADADB Archive Uptime	Archive up and running time since started
ADADB Archive Messages Received Rate	Rate of TIBCO Rendezvous messages received
ADADB Archive Messages Sent Rate	Rate of TIBCO Rendezvous messages published
ADADB Archive Messages Received	Number of TIBCO Rendezvous messages received
ADADB Archive Messages Sent	Number of TIBCO Rendezvous messages published
ADADB Archive Total Errors	Total number of errors since startup
ADADB Archive QueueCount	Current number of elements in the queue
ADADB Archive MaxQueueSize	Maximum number of elements in the queue

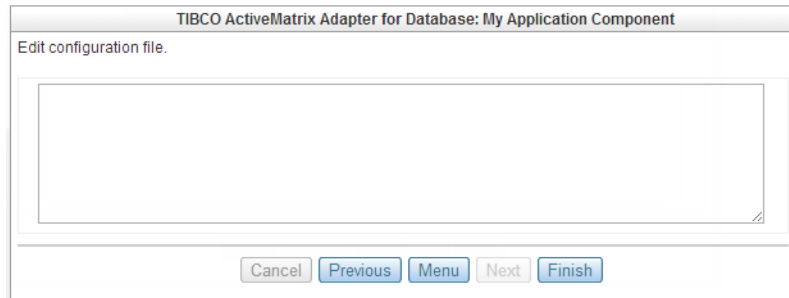
For more information on using these statistics for scaling, refer to [Adapter for Database Statistics on page 34](#)

## Task L Edit Configuration File Screen

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

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

Figure 18 Edit the Configuration File Page



As an example, if a user wants to replace the property `tibco.env.CUSTOM_PATH`, `tibco.env.CUSTOM_LIB_PATH` and `tibco.env.ODBCINI` in `${TIBCO_HOME}/adapter/adadb/6.3/bin`, the following XML example code described, accomplishes this when used in the Edit configuration file text field.

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<containerConfig>
  <configFiles baseDir="${TIBCO_HOME}/adapter/adadb/6.3/bin"
    include="adbagent.tra">
    <regex pattern="tibco\.env\.CUSTOM_PATH"
      replacement="tibco.env.CUSTOM_PATH xxxx" />
    </configFiles>
  <configFiles baseDir="${TIBCO_HOME}/adapter/adadb/6.3/bin"
    include="adbagent.tra">
    <regex pattern="tibco\.env\.CUSTOM_LIB_PATH.*"
      replacement="tibco.env.CUSTOM_LIB_PATH xxxx" />
    </configFiles>
  <configFiles baseDir="${TIBCO_HOME}/adapter/adadb/6.3/bin"
    include="adbagent.tra">
    <regex pattern="tibco\.env\.ODBCINI.*"
      replacement="tibco.env.ODBCINI xxxx" />
    </configFiles>
</containerConfig>
```

The property, **baseDir**, in the <configFiles> element specifies the path that includes the file to be updated. You can modify it if required. For example, if the TIBCO ActiveMatrix Adapter for Database (ADADB) version is 6.3 instead of 6.2, the baseDir value is:

```
${TIBCO_HOME}/adapter/adadb/6.3/bin
```

The property, **include**, in <configFiles> element specifies which files 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, for instance: `"*.tra"` to change all of the .tra files in %baseDir%.

The property, **pattern**, in the <regex> element specifies 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 specifies the new contents of the node specified by the pattern property value.

Where xxxx is the path in your implementation environment.

## Task M Finish Configuring the Component

Almost all of the other screens are generic for TIBCO Silver<sup>®</sup> Fabric Containers.

These final configurations are optional for Adapter for Database components. Refer to *TIBCO Silver<sup>®</sup> 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.

To do this, select **Publish Components** in the **Actions** list located at the line of the component you just created.



## Changing the Component Enabler

Upgrading a component created with TIBCO Silver Fabric Enabler for Adapter for Database release 2.5 to 3.0 is easy.

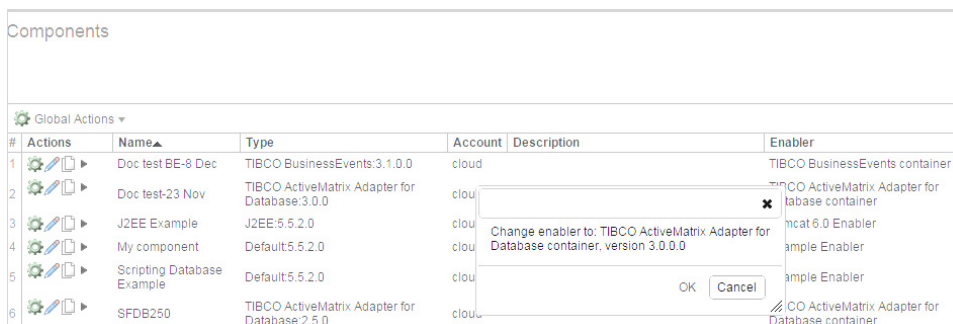


Back up your engines and consider when you wish to perform a component restart, so brief operational downtime has minimal impact.

1. Using the TIBCO Silver Fabric Administrator > Components page identify the Enabler for Adapter for database component you wish to upgrade from release 2.5.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**.

Figure 19 Upgrading a Component - Change Enabler



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**.

Figure 20 Restart the Component

Engines

Global Actions ▼

#	Actions	Host Name	Instance	Status	Draining	Up To Date	Component	Account	Enabler
1		lin64vm432	0	Running	no	yes	AJSON EMS	cloud	TIBCO EMS Server container 2.0.0
2		lin64vm114	0	Running	no	yes	Test_v11	cloud	TIBCO EMS Server container 2.0.0
3		lin64vm025	0	Running	no	yes	Central Admin EMS v13	cloud	TIBCO EMS Server container 2.0.0
4		lin64vm402	0	Running	no	Needs Component Restart	Admin 2.6.0.5 Instance	cloud	TIBCO Administrator container 2.8.0.4

Engine Details

Kill Engine

Restart Component

Clear from Blacklists

Search Logs

Log URL List

Page: 1 of 1

## Creating a TIBCO Silver Fabric Enabler for Adapter for Database Stack

A TIBCO Silver Fabric Enabler for Adapter for Database stack normally contains a single TIBCO Administrator Component and one or more TIBCO Adapter for Database components. But a stack can also be created with a single component and it could be run in standalone configurations. After creating and publishing the above components, you can create TIBCO Silver Fabric Enabler for Adapter for Database stack.

Each *TIBCO\_DOMAIN* has one TIBCO Silver Fabric Enabler for Adapter for Database stack. After starting a stack, you can update it by adding or removing Adapter for Database components.

To create a TIBCO Silver Fabric Enabler for Adapter for Database stack:

1. From the TIBCO Silver<sup>®</sup> Fabric Administration Tool, select **Stacks > Stacks**.
2. From the **Global Actions** list, select **Create New Stack** as shown in [Figure 21](#).

Figure 21 Creating a Stack

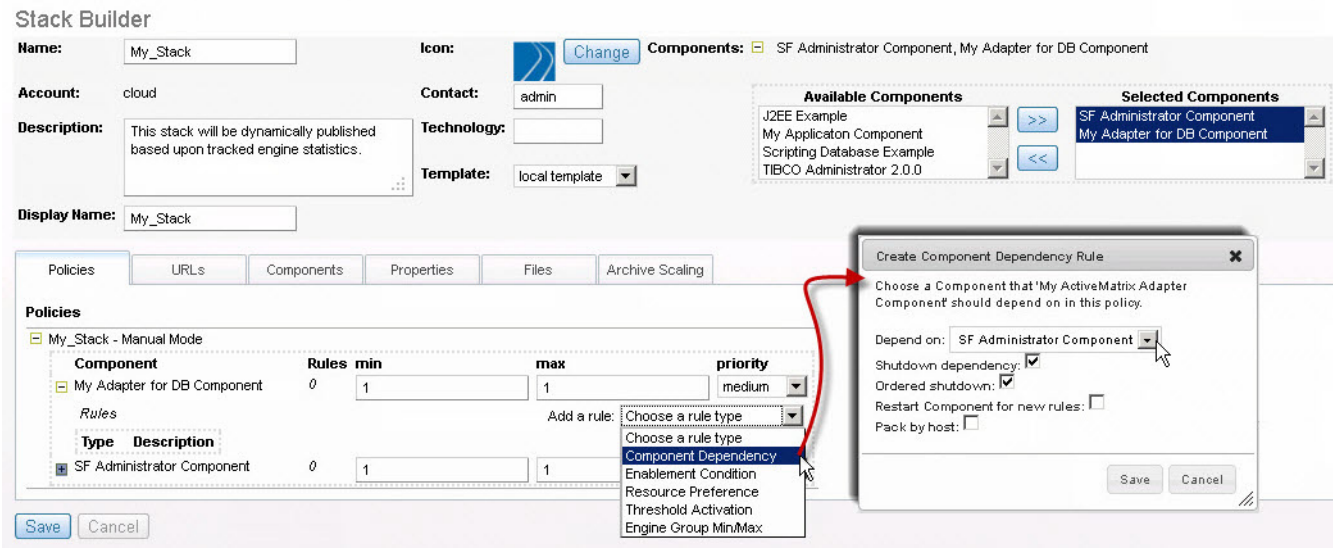
The screenshot shows the TIBCO Silver Fabric Administration Tool interface. At the top, there is a navigation bar with tabs: Dashboard, Stacks, Engines, Users, Reports, Admin, Config, and Diagnostics. The 'Stacks' tab is selected. Below the navigation bar, there is a 'Stacks' section with a 'Help' button. A search bar is present with the text 'Search...' and a link to 'Advanced Search'. Below the search bar, there is a table of stacks. The table has columns: #, Actions, Name, Account, Description, Status, Mode, Active Policy, and Changed. The 'Create New Stack' button is highlighted in the 'Actions' column of the first row.

#	Actions	Name	Account	Description	Status	Mode	Active Policy	Changed
1		AMX SG Stack	cloud		not published	--	--	--
2		Example Stack	cloud	An example stack consisting of an HSQLDB database component and a J2EE web application component.	published	stopped	--	no
3		My_Stack	cloud	This Stack will be dynamically published based upon tracked engine statistics.	published	auto	--	yes

3. Enter a stack name in the stack Builder page as shown in [Figure 22](#).
4. In the **Components** area, add one TIBCO Administrator Component and one or more Adapter for Database components.

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

Figure 22 Stack Builder Page



## Dependency Requirements

Each Adapter for Database component, must have a component dependency set on one TIBCO Administrator component. You must set this component dependency for each of your Adapter for Database components. Those components without defined component dependencies do not have enough information for proper publishing.



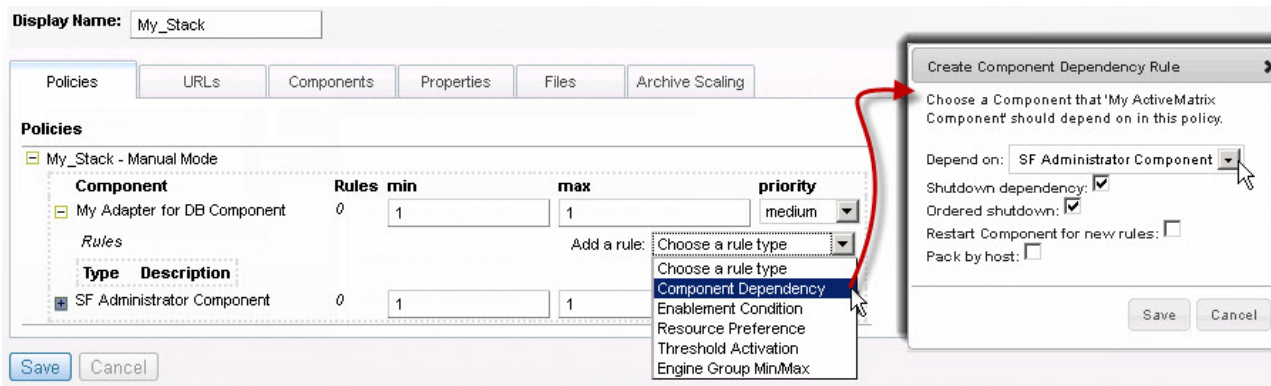
This step is required. TIBCO Silver Fabric Enabler for Adapter for Database does not work, unless you set the dependency.

The Adapter for Database component must have the TIBCO Administrator configuration information so that it may publish, unpublish, and communicate with other components (if required) successfully. A connection is supported between the Adapter for Database 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 Database starts, after TIBCO Administrator running.

To set the dependency, follow these steps:

1. During creation or edit of the Adapter for Database 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.

Figure 23 Setting Administrator Component Dependency



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



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

If you configure Administrator component to **Use dependent EMS server**, set that dependency here as well.

If you use a dependent TIBCO Enterprise Message Service™ server, set the dependency in the TIBCO Administrator component, which must also have a dependency on TIBCO EMS Server component.

### 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 database continues to work. However, every change in TIBCO Administrator is not taken into account by Enabler for Adapter for database 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*.

4. Click **Save** or create another component dependency on the Enabler for Adapter for database client component that connects with the Enabler for Adapter server.
5. Set a component dependency on Enabler for Adapters client components using the same **Add a rule** drop-down menu to set dependence on the single instance of a Enabler for Adapter server component in your stack.
6. **Save** and **Run** your stack.

## Adapter for Database Statistics

---

If you want Adapter for Database components to scale (add new engines) automatically, you can define rules that add or remove TIBCO Silver<sup>®</sup> Fabric engines based on engine statistics.

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

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

When an engine is added, it automatically publishes the Adapter for Database project on a new engine.

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

To set up rules for an engine, follow these steps:

1. In the **Policies** area of the **Stack Builder** application builder page, select the component for which you want to set up rules.
2. Select **Threshold Activation** or **Enablement Condition** in the Add A Rule list.
3. If you select **Threshold Activation**, specify the following parameters in the Create Threshold Activation Rule panel:
  - From the **Condition Type** list, select the **Component Statistic** item.
  - From the **Action** list, select **Add Engine** or **Remove Engine**.
  - From the **Component** list, select the component where the statistic rules apply.
  - From the **Statistics** list, select the statistics property on which the activation is based.
  - From the **Comparison** list, select the operator such as **Greater Than** or **Less Than**.
  - From 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.
  - From 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.



Figure 24 Creating Rules

**Create Threshold Activation Rule** [X]

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

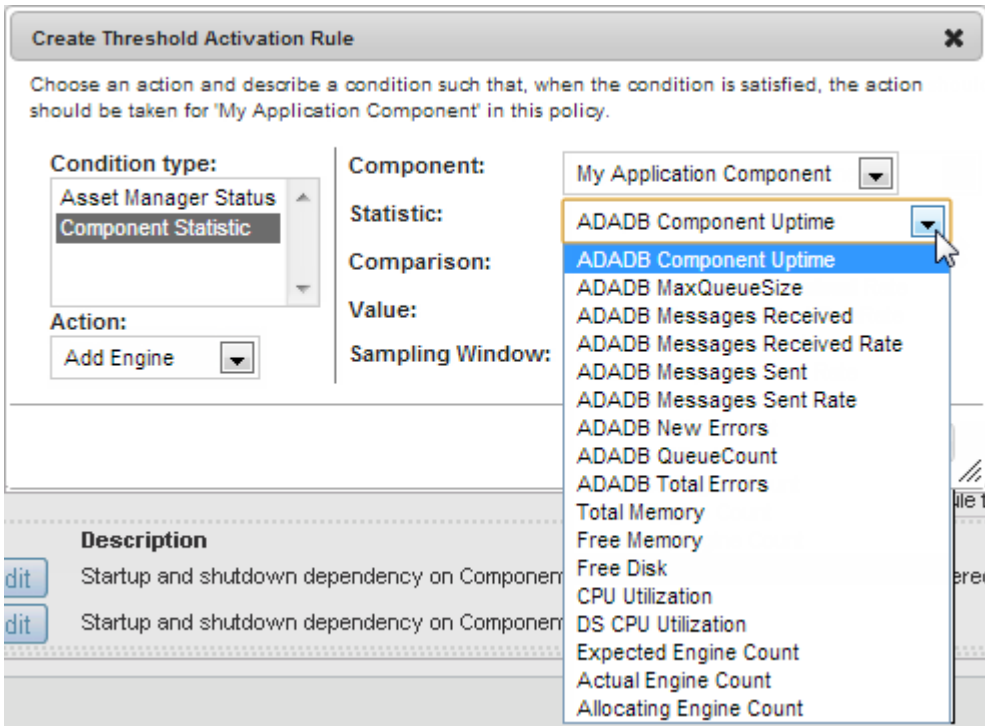
<b>Condition type:</b> Asset Manager Status <b>Component Statistic</b>	<b>Component:</b> My Application Component
<b>Action:</b> Add Engine	<b>Statistic:</b> Free Memory
	<b>Comparison:</b> Less than
	<b>Value:</b> 1000.0 KB
	<b>Sampling Window:</b> 120 seconds

Save Cancel

Statistics consist of engine and machine measures that are independent of Adapter for Database. Any other statistical information gathered from the Adapter for Database through TIBCO Hawk<sup>®</sup> when that is applicable.

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

Figure 25 Statistics Available with the Silver Fabric Enabler for Adapter for Database Component



### Create Archive Scaling Rules

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

Create archive scaling rules to automate creation or remove new Adapter for Database component archive instances 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 instances are 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.

Figure 26 Creating a New Archive Scaling Rule

Archive Scaling Rule Editor

Name: My Adapter for DB Scaling Rule Max: 10

Description: Dynamically launch new Adapter for Database archives based on component load as measure Archive Queue Count.

Archives: Add Archive Conditions Remove Archive Conditions Target Component Conditions Advanced Options

Component Type	Version	Archive Names
TIBCO ActiveMatrix Adapter for Data	2.5.0	MyArchive

any  
TIBCO ActiveMatrix Adapter for Database  
TIBCO Administrator

Save Cancel

Select the Adapter for Database component that was used to upload the application archive file. 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 statistics might not be reported in a particular time-frame, creating an inadvertent trigger condition.

By default, allocation statistics like Expected Engine Count, Client Count, Allocating Engine Count, and Actual Engine Count are collected every 60 seconds. 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 may be satisfied to trigger addition of a new archive instance.

Figure 27 Define Archive Scaling Rule Conditions

Archive Scaling Rule Editor

Name:My Adapter for DB Scaling RuleMax:10

Description:Dynamically launch new Adapter for Database archives based on component load as measured by Archive Queue Count.

Archives

Add Archive Conditions

Remove Archive Conditions

Target Component Conditions

Advanced Options

Add an instance of the Archive when:

Satisfies: Any

Statistic Name	Operator	Value	Sampling Window
ADADB Archive QueueCount	Greater than	50	120
ADADB Archive MaxQueueSize			
ADADB Archive Messages Received			
ADADB Archive Messages Received Rate			
ADADB Archive Messages Sent			
ADADB Archive Messages Sent Rate			
ADADB Archive New Errors			
ADADB Archive QueueCount			
ADADB Archive Total Errors			
ADADB Archive Uptime			

Save

Cancel

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

The **Remove Archive Conditions** tab releases computing resources and removes unused or idle component archives or process instances. This is done to scale down your component archives in the same way you scale them up according to conditions you define on usage statistics.

The **Target Component Conditions** tab restricts the start of new archive instances to those machines that have the same set of resources that you choose. Set a rule or several rules with statistics, operators, and values as you set on the **Add Archive Conditions** tab. Then restrict where the new archive instances may 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 Database archive can scale on an adapter component.

**Same component Type:** the process archive might scale 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 Database, TIBCO TRA, TIBCO Hawk, and so on.

**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.

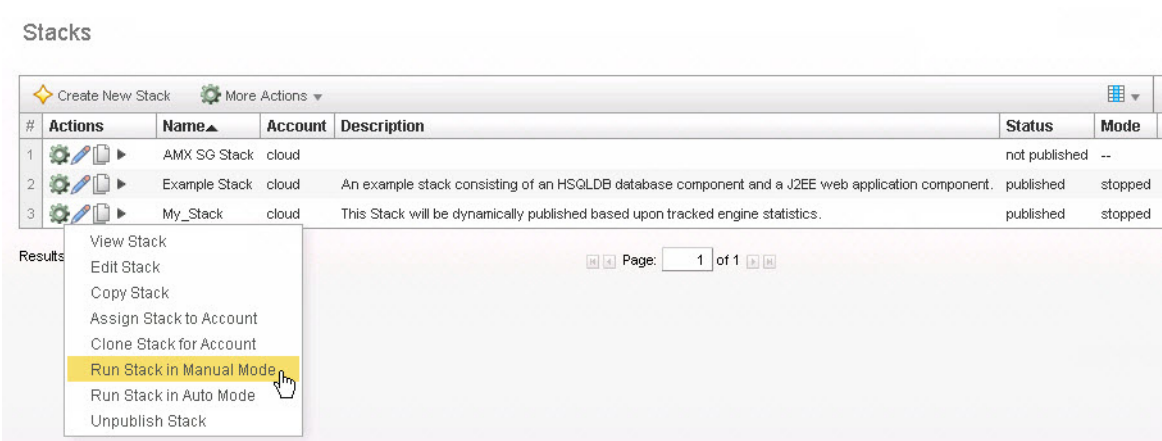
Provided that the component archive has the proper Component Type, TIBCO Silver Fabric can usually find the correct computing environment for scaling up. For example, for an Adapter for Database Component Archive, a selection of the "Same Component Type" ensures that the Silver Fabric Broker finds an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

# Running TIBCO Silver Fabric Enabler for Adapter for Database Stack

After you have created your stack, you must publish it. Use **Publish Stack**, available from the **Actions** drop-down list, to run it.

After publishing, click **Run Stack In Manual Mode** in the **Actions** drop-down list as shown in [Figure 28](#) to run the stack immediately on available resources.

Figure 28 Running a Stack



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

## Updating TIBCO Silver Fabric Enabler for Adapter for Database Stack

---

When a stack is published and running, you can still make changes to the stack. This includes adding other components, changing allocation rules, changing threshold activation rules, or deploying and starting archives on the runtime adapter for the database application instantiated on the engine.

Making changes to the stack is as easy as editing, saving, and publishing those changes to any instance that may be running. Some changes might require restarting 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, click **Save** to 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 Adapter for Database component, stop and restart your entire stack. To deploy, start, stop, and undeploy Adapter for Database project archives, use the following ways:

**Micro-scaling:** Start and stop Adapter for Database archives based on your defined rules when they are already in your component. For more information, refer to: [Create Archive Scaling Rules on page 36](#).

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

### Deploying Archives Directly to Endpoints - Continuous Deployment

In some situations, deploying archives directly to a running TIBCO ActiveMatrix Adapter for Database Component can be useful. For example, when an archive must 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 Database 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 through REST are described in further detail here.

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



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

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

The TIBCO Silver Fabric Enabler for Adapter for Database adds more REST methods to enable control of Adapter for Database 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 may be executed using cURL methods:

- **Deploy:** send an archive to a Silver Fabric Engine running an Adapter for Database Component that meets the criteria specified. The Deploy REST method enables specification of properties files with criteria dictating where and how the archive should be deployed.
- **Start:** 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 Database application archives may 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 "Archives=Archive_A,Archive_B,Archive_X"]
    [-F "configurationFile=YourConfigurationfile.xml"]
    [-F "ForceDeploy=true"]
    [-F "GV=globalVariableA=123,globalVarB=SomeString"]
    [-F "NoDeploy=true"]
    [-F "NoStart=true"]
[-F "VariableProvider=Some_PROVIDER(S)_Name"]
[-F "InstanceSettings.element1.element2=SomeValue"]

```

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

*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:%2F%2FMySFBroker.com:8080%2Flivecluster%2Frest%2Fv1%2Fsf%2Feng
ines%2Farchives -F "archiveFile=@MyProcessOrder.ear" -F
"AppName=MyOrders%2FMyProcOrder"%20-F
"deploymentFile=@MyDeployCriteria.properties"

```

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

Mandatory form-data contain the files necessary for deployment:

**archiveFile:** specifies your Adapter for Database archive file (.zip, .aar, or .ear file) to upload to the Silver Fabric Broker, which then publishes the archive to the appropriate Silver Fabric Engine. Multiple application archives may be deployed in a single archive .zip or .ear file. You can deploy and run them all (default behavior) or you can selectively run a list of archives by specifying that list with the **Archives** form-data field. Archives may also be uploaded using the component wizard.

**deploymentFile:** specifies the properties file that defines 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 archives are 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 a ear file with a name like *MyAppArchive*, varying the AppName specification gives the following behavior:

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

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

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 that case that archive can run without interruption or replacement.

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

All deployment configuration cURL expressions takes the form:

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

Some examples:

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

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

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

- 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.

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



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

**Archives** (optional): form-data parameter that specifies a comma delimited list of archives that are to be deployed within the .zip file that is to be deployed. If an **archives** list is omitted, 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 do not 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 equal (=) sign. For example:

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



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

c. **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.

d. **failureEvents.failureEvent:**

failureType: *ANY, FIRST, SECOND, Subsequent*

If an event is used, type must be specified.

Example syntax:

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

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

description: information that describes this operation.

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

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

alertAction.level: *High, Medium, Low*

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

e. **failureEvents.emailAction:**

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

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

**message:** the message to send.

**to:** a comma-separated list of email addresses to which the messages are sent.

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

**subject:** the subject of the email message.

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

f. **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

g. **suspendProcessEvents.suspendProcessEvent:**

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

**description:** information that describes this operation.

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

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

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

**alertAction.message:** the message that displays when this alert is triggered.

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

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

`emailAction.message`: the message to send.

`emailAction.to`: a comma-separated list of email addresses, to which the messages are sent.

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

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

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

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

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

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

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

#### h. **logEvents.logEvent**:

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

Example syntax:

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

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

`description`: information that describes this operation.

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

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

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

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

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

occurrence.

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

`emailAction.message`: the message to send.

`emailAction.to`: a comma-separated list of email addresses, to which the messages are sent.

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

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

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

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

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

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

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

- i. **failureCount**: The value in this field defines how many restarts should be attempted before resetting the error counter to 0.
- j. **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. The default value is `false`.
- d. **enableVerbose:** when `true`, sets the enabler for verbose tracking for service instances. The default value is `false`.
- e. **threadCount:** specifies the number of threads to use to execute process instances. The number of threads determines how many process instances can execute concurrently.
- f. **prependClassPath:** the values supplied here are prepended to your CLASSPATH environment variable.
- g. **appendClassPath:** the items you supply here are appended to your CLASSPATH environment variable.
- h. **initHeapSize:** specifies the initial size (in MB) for the JVM used for the process engine. The default is 32 MB.
- i. **maxHeapSize:** specifies the maximum size (in MB) for the JVM used for the process engine. The default is 256 MB.
- j. **threadStackSize:** specifies the size of the thread stack. The default is 256 KB.
- k. **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.
- l. **startupType:** specifies the instance service startup type as either: `Automatic`, `Manual`, or `Disabled`.
- m. **login:** specifies the login account for the service, if any. The domain name must be specified as well.
- n. **password:** sets the password for that service, if any.
- o. **checkpoint:** when set to `true`, the process engine waits for all jobs to finish (up to the maximum timeout) before shutting down the engine, rather than remove jobs at their next checkpoint.
- p. **timeout:** the maximum timeout in seconds the process engine will wait for jobs to finish before shutting down the engine. A zero (0) value



means 0 seconds, which effectively turns the graceful shutdown into an immediate shutdown.

- q. **iv** : this element uses a comma-separated string with name-value pairs with each key value pair joined by an equal (=) sign. For example:

**configurationFile** (optional): form-data parameter used to include an XML configuration file created to modify archive properties if needed. Example syntax:

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

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

```
<archiveConfig name="YourArchiveName">
  ...
</archiveConfig>
```

For more information on writing an archive configuration file, see the "Using the Silver Fabric SDK" chapter of the Silver Fabric Enabler for Adapter for Database Developer's Guide.

**ForceDeploy** (optional): redeploy, forces a stop and overwrites preexisting archive or set of archives with the same name. By default, `ForceDeploy` is set to `false` and so a second deployment does not overwrite a pre-existing deployment of the same name. If the archive file is changed, 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, only those archives are stopped, undeployed, and redeployed.

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

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

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

If global variables are not defined with explicit values in the `cURL` statement, then the values in the deployment `configurationFile.xml` is applied.

**NoDeploy**(optional): default value is `false` which means that the archives 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 "ADB" as part of the name which has a domain name of *YourDomain*:

```
ComponentType contains ADB
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

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"><li>• id: The numeric ID unique to each engine, generated upon installation</li><li>• guid: The globally unique identifier for the engine (network card address)</li><li>• instance: The instance of the engine, starting with zero. Multi-CPU hosts may have multiple instances of Engines running concurrently</li><li>• IP: The IP address of the engine</li><li>• username: The username of the engine, which is the hostname of the engine unless installed with tracking properties</li><li>• hostname: The hostname of the engine</li><li>• cpuNo: The number of CPUs on the engine's host computer</li><li>• 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</li><li>• 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</li><li>• 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</li><li>• cpuIdString: The CPUID string of the first physical CPU on the Engine's host computer, if available</li><li>• 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</li></ul>

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

Name	Definition
ActivationInfoProperty( <i>name</i> )	<p>A named property, such as:</p> <ul style="list-style-type: none"> <li>• LoginTime: Constant representing the creation time of the login message property</li> <li>• ActivationTime: Constant representing the activation time property</li> <li>• HTTPS_PORT: Constant representing the name of the HTTPS port property</li> <li>• HTTP_PORT: Constant representing the name of the HTTP port property</li> <li>• Activation Id: Constant representing the name of the activation ID property</li> <li>• Engine Id: Constant representing the name of the engine ID property</li> <li>• HasShutdownDependency: Constant representing the value for the has shutdown dependencies property</li> <li>• EnablerVersion: Constant representing the name of the Enabler version property</li> <li>• ComponentName: Constant representing the name of the Component name property</li> <li>• EnablerName: Constant representing the name of the Enabler name property</li> <li>• EngineInstance: Constant representing the name of the Engine instance property</li> <li>• Hostname: Constant representing the name of the Engine hostname property</li> <li>• IpAddress: Constant representing the name of the Engine IP address property</li> <li>• ContextList: Constant representing the name of the context list property</li> <li>• EnginePort: Constant representing the name of the Engine port property</li> <li>• RoutingPrefix: Constant representing the routing prefix</li> <li>• HTTP_STATIC_ROUTE: Constant representing prefix for static HTTP routes</li> </ul>

Name	Definition
<p>..(continued)</p> <p>ActivationInfoProperty(<i>name</i>)</p>	<ul style="list-style-type: none"> <li>• REDIRECT_TO_ENDPOINTS: Constant representing the VirtualRouter setting for whether to redirect (<code>true</code>) or forward (<code>false</code>) requests to endpoints</li> <li>• DeployedArchives: Constant representing the currently deployed archives property</li> <li>• CD_DeployedArchives: Constant representing the current subset of deployed archives property, originated from Continuous Deployment</li> <li>• RunningArchives: Constant representing the currently running archives property</li> <li>• CD_RunningArchives: Constant representing the current subset of running archives property, originated from Continuous Deployment</li> <li>• ScaledArchives: Constant representing the currently scaled archives property</li> <li>• ArchiveProvider: Constant representing the archive provider property</li> <li>• account: Constant representing the account property</li> <li>• ClusterName: Constant representing the cluster name property</li> <li>• BALANCER_STRATEGY: Constant representing the VirtualRouter setting for the load balancing strategy. Valid settings are "Random", "Round_Robin", "Queue_Depth_Weighted", and "Statistic"</li> <li>• STATISTIC_BASED_ROUTING_STAT_NAME: Constant representing the name of the statistic to use for the "Statistic" balancer strategy.</li> <li>• STATISTIC_BASED_ROUTING_LIMIT: Constant representing the upper or lower limit value for STATISTIC_BASED_ROUTING_STAT_NAME</li> <li>• STATISTIC_BASED_ROUTING_SCALE_DIRECTION: Constant representing whether higher or lower values of the statistic specified by STATISTIC_BASED_ROUTING_STAT_NAME are preferred. Valid settings are "Higher" and "Lower"</li> </ul>

Name	Definition
<div>..(continued)</div> <div>ActivationInfoProperty(<i>name</i>)</div>	<ul style="list-style-type: none"><li>• SESSION_IDENTIFIER: Constant representing the VirtualRouter setting for the type of identifier to use for sessions. Valid settings are "Cookie", "Parameter", and "Header"</li><li>• SESSION_IDENTIFIER_NAME: Constant representing the name of the identifier to identify sessions</li><li>• APPLICATION_SESSION_IDENTIFIER: Constant representing the VirtualRouter setting for the type of identifier to use for application sessions. Valid settings are "Cookie", "Parameter", and "Header".</li><li>• APPLICATION_SESSION_IDENTIFIER_NAME: Constant representing the name of the identifier to identify application sessions.</li><li>• CONTEXT_AWARE_ROUTING_PATTERN: Constant representing the variable defining a context-aware routing pattern</li><li>• ACTIVE_TUPLES: Constant representing the list of active tuples for context aware routing</li><li>• ENABLE_URL_REWRITING: Constant representing whether to enable content rewriting in VirtualRouter</li><li>• ADDITIONAL_REWRITE_CONTENT_TYPES: Constant representing the setting for additional content types to rewrite</li><li>• ADDITIONAL_REWRITE_HOSTS: Constant representing the setting for additional URL rewrite hosts</li><li>• ADDITIONAL_URL_REWRITE_PATTERN: Constant representing the prefix used to denote properties to be used as additional URL rewrite regular expression patterns. The actual URL should be in a capturing group in the pattern</li><li>• SESSION_TAKES_PRECEDENCE_OVER_CONTEXT_AWARE_ROUTING: Constant representing the setting for whether or not VirtualRouter session mappings take precedence over context aware routing pattern matching or if they are considered together</li><li>• Component Instance: Constant representing the variable that defines the component instance number</li></ul>
ImportedVariable( <i>name</i> )	A variable imported into a component



Name	Definition
<code>ExportedVariable(name)</code>	A variable exported from a component.
<code>Statistic(name)</code>	A named Adapter for Database Enabler statistic such as: ADADB New Errors, ADADB Component Uptime, ADADB Messages Received Rate, ADADB Messages Sent Rate, ADADB Mesages Received, ADADB Mesages Sent, ADADB Total Errors, ADADB QueueCount, ADADB MaxQueueSize, ADADB Archive New Errors, ADADB Archive Uptime, ADADB Archive Messages Received Rate, ADADB Archive Messages Sent Rate, ADADB Archive Messages Received, ADADB Archive Messages Sent, ADADB Archive Total Errors, ADADB Archive QueueCount, or ADADB Archive MaxQueueSize
<code>ArchiveStatistic(statName, archiveName)</code>	A named statistic for a specified archive
<code>DependencyComponent</code>	A named dependency on a component
<code>DependencyEngine(component Name)</code>	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 Database"
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 `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 startup 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



Ensure that you do not deploy files larger than 1 GB, 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/{appname}/start"

```

Expressions in the preceding generic form cURL expression above 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.

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

You can use 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 *TIBCO Silver Fabric Administration Tool* at **Admin > REST Services**.

The **{appname}** must be copied from the TIBCO Silver Fabric Administrator > Dashboard > **Scaled Archives** page.



The **{appname}** must be URL-encoded in a cURL statement so that spaces are converted to "%20" and forward slashes "/" are represented by "%2F". Likewise, other special characters must be encoded appropriately.

*Example 3 A cURL example that starts an Archive.*

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

```
/Silver Fabric/SFDB 22E5/DomainMachineName/processOrder/Orders/MyProcOrder/Process_Archive.aar
```

...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%2FSFDB%2
022E5%2FDomainMachineName%2FprocessOrder%2FOrders%2FMyProcOrder/Pr
ocess_Archive.aar/start"
```

## Stop

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

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

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

## Undeploy

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

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

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

**DeleteApp** (optional): This is to be used 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 Silver Fabric Enabler for Adapter for Database Administration Tool. For information on using REST services, see "Using REST Services" in the Silver Fabric Enabler for Adapter for Database 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="sfdb-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=SFDB Example Component
component.description=SFDB Example Component built by Ant
component.type=TIBCO ActiveMatrix Enabler for Adapter for database:3.1.0
enabler.name=TIBCO ActiveMatrix Enabler for Adapter for database container
enabler.version=3.1.0.0
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

---

- [Adapter for Database Log Files, page 69](#)
- [Retained Log Files, page 70](#)

## Adapter for Database Log Files

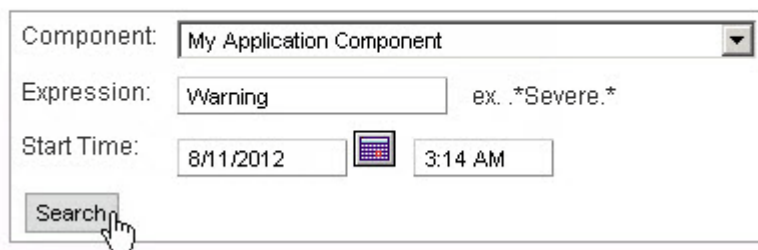
---

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

1. In TIBCO Silver<sup>®</sup> Fabric Administration Tool, select **Engines > Log Search**.
2. Select the component, from which you want to see the log files, as shown in [Figure 29](#).
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 29 Log Files*

### Log Search



The screenshot shows a 'Log Search' dialog box with the following fields and controls:

- Component:** A dropdown menu showing 'My Application Component'.
- Expression:** A text input field containing 'Warning'. To its right is a small icon and the text 'ex. .\*Severe.\*'.
- Start Time:** A date input field showing '8/11/2012' and a time input field showing '3:14 AM'. A small calendar icon is positioned between the date and time fields.
- Search:** A button at the bottom left with a hand cursor pointing to it.

# Retained Log Files

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

- [TIBCO Hawk Agent Log Files, page 70](#)
- [TIBCO Administrator Log Files, page 70](#)
- [Adapter for Database Log Files, page 69](#)

## TIBCO Hawk Agent Log Files



When TIBCO Administrator runs in the Fault Tolerant mode, all files are not located under the TIBCO Silver<sup>®</sup> Fabric *\$ENGINE\_WORK\_DIR* directory. The Hawk<sup>®</sup> log files do not appear in the TIBCO Silver<sup>®</sup> Fabric Administrator GUI.

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

Table 3 Retained TIBCO Hawk<sup>®</sup> Agent Log Files

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

## TIBCO Administrator Log Files



When TIBCO Administrator runs in the Fault Tolerant mode, all files are not located under the TIBCO Silver<sup>®</sup> Fabric *\$ENGINE\_WORK\_DIR* directory. The TIBCO Administrator log files do not appear in the TIBCO Silver<sup>®</sup> Fabric Administration Tool.

**Table 4** lists the retained TIBCO Administrator Log Files.

*Table 4 Retained TIBCO Administrator Log Files*

Name	Location	Purpose of the Log
audit.log	<code>\$ENGINE_WORK_DIR/domaindata/admin/TIBCO_DOMAIN/logs</code>	All information about TIBCO Administrator activities.
tomcat.log	<code>\$ENGINE_WORK_DIR/domaindata/admin/TIBCO_DOMAIN/tomcat/logs</code>	Technical log of TIBCO Administrator that runs on Tomcat.
domainutility.log	<code>\$ENGINE_WORK_DIR/tibco/tra/tra_version_2digits/logs</code>	Log file of the domainUtility command used to create the domain or add the machine. This file is common for all engines.

### Adapter for Database Deployment Log Files

**Table 5** lists the retained Adapter for Database Log Files.

*Table 5 Retained TIBCO Adapter for Database Log Files*

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



# Index

## Symbols

[41](#), [42](#), [51](#), [61](#)

## A

Add Archive Conditions [37](#)

Administrator Component

publish [26](#)

AMI Hawk Daemon [17](#)

AMI Hawk Service [17](#)

AppName [44](#)

Archive Scaling Rules [36](#)

archiveFile [43](#)

Archives [45](#)

ArchiveSettings [45](#)

## C

Component upgrade [27](#)

Component, containers [3](#)

Components [3](#)

Components, creating [7](#)

Configuring the Component, task list [7](#)

Continuous Deployment [41](#), [41](#), [41](#), [42](#)

Create Archive Scaling Rules [36](#)

Creating a Stack [29](#)

Creating an application [29](#)

customer support [xiii](#)

## D

DeleteApp [62](#)

Dependency Requirements [31](#)

dependency, setting [31](#)

Deploying Archives [41](#)

deploymentFile [43](#)

distributions [3](#), [9](#)

Documentation [x](#), [x](#), [xiii](#)

Domain Machine Name [11](#)

## E

EAR File [15](#)

Enablement condition [34](#)

entity [6](#)

environment variable [20](#)

## F

failureEvents [46](#)

Fault Tolerant [32](#), [32](#)

ForceDeploy [51](#)

FTWeight [51](#)

Functionalities [2](#)

## G

global variables [51](#)

GV [51](#)

## H

Hawk AMI Configuration [17](#)

Hawk Application Management Interface [17](#)  
 hbInterval [46](#)

## I

InstanceSettings [50](#)

## L

localRepoInstance [45](#)  
 Log Files  
   Administrator [70](#)  
 Log files [69](#)  
 log files  
   ApplicationManagement.log [71](#)  
   audit.log [71](#)  
   domainutility.log [71](#), [71](#)  
   Hawk.log [70](#)  
   msghma.log [70](#)  
   tomcat.log [71](#)  
   tsm.log [70](#)  
 Log Files, Deployment [71](#)  
 Log Files, Hawk Agent [70](#)  
 Log Files, retained [70](#)  
 logEvents [48](#)  
 LogicalAnd [43](#)

## M

maxDeploymentRevision [45](#)

## N

NoDeploy [51](#)  
 NoStart [52](#)

## P

Product\_HOME [xi](#)

## R

runtime specific context variables [20](#)

## S

SILVERFABRIC\_HOME [xi](#)  
 source ID [34](#)  
 Statistics [34](#)  
 support, contacting [xiii](#)  
 suspendProcessEvents [47](#)  
 system variable [20](#)

## T

Target Component Conditions [38](#)  
 tasks, creating an application [6](#)  
 technical support [xiii](#)  
 Threshold Activation [34](#)  
 TIBCO Administrator Container [3](#)  
 TIBCO Domain Logical Machine Name [11](#)  
 TIBCO Silver Fabric Enabler, updating [41](#)  
 TIBCO Silver Fabric Engine [34](#)  
 TIBCO\_HOME [xi](#)  
 TIBCOCommunity [xiii](#)  
 TLM [11](#)

## U

Upload projects [15](#)



## V

variable, environment [20](#)

variables, encrypted [20](#)

variables, System [20](#)

## X

XML file [15](#), [15](#)

## Z

ZIP File [15](#)

