TIBCO Silver® Fabric Enabler for Apache Tomcat User's Guide

Software Release 6.0
December 2017

Two-Second Advantage®

TIBCO®

Important Information

Contents

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

Documentation for TIBCO Silver® Fabric Enabler for Apache Tomcat is available on the TIBCO Silver Fabric Enabler for Apache Tomcat Product Documentation page.

Product-Specific Documentation

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

- *Silver Fabric Concepts Guide*
- *Silver Fabric Installation Guide*
- *Silver Fabric Cloud Administration Guide*
- *Silver Fabric Developer's Guide*
- *Silver Fabric Developer's Tutorial*
- *Silver Fabric User's Guide*
- *Silver Fabric Skyway User's Guide*
- *Silver Fabric Enabler for Apache Tomcat User's Guide*
- *Silver Fabric Command Line Enabler Guide*
- *Silver Fabric Administration Tool Help*
- *API Reference*

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit http://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to https://community.tibco.com.

About the Apache Tomcat Enabler

Overview

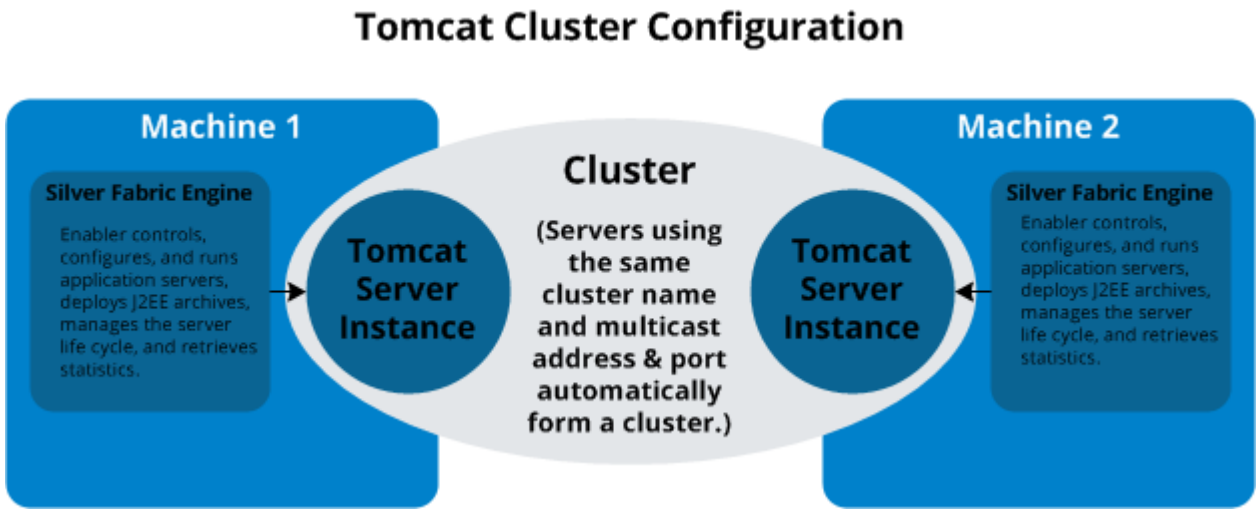
A Silver Fabric Enabler allows an external application or application platform, such as an application server, to run in a TIBCO Silver® Fabric software environment. The TIBCO Silver Fabric enabler for Apache Tomcat provides integration between Silver Fabric and the Tomcat Servlet/JSP Server.

The enabler automatically provisions, orchestrates, controls and manages a Tomcat standalone or clustered environment. Tomcat server instances are distributed and run on Silver Fabric engines, either as standalone servers or as servers belonging to a Tomcat partition (cluster). The management of the Tomcat environment through Silver Fabric is dynamic, centralized, and largely automated. This centralized management adapts the structure of the Tomcat environment, and manages the life cycle process state of server instances, as specified in Silver Fabric by Policies that are directed by business objectives associated with the target applications.

Architecture

The Silver Fabric engine manages a Tomcat standalone or clustered server instance. The Silver Fabric broker and the Silver Fabric engine instance collaborate to perform the following actions:

- Automatically provision all Tomcat-related and Java JDK software to machines with Silver Fabric engines. This software is encapsulated within archive libraries (.zip files) and distributed to computers running Silver Fabric engines.
- Automatically start and stop Tomcat servers on a node, as required by policies specified within Silver Fabric.
- Monitor the health of the Tomcat Server and retrieve statistics.



Installing the Tomcat enabler

The following procedure describes how to install the Tomcat enabler:

Prerequisites

- This guide presumes a strong familiarity with your particular version of Tomcat. If you are uncertain of how to achieve a particular task, consult your version specific Tomcat documentation.
- This guide assumes a Silver Fabric broker is running with at least one engine installed, and that you have the broker's host name, a user name, and password for the Silver Fabric Administration Tool. If this is not true, see the *Silver Fabric Installation Guide*, or contact your administrator.
- Refer the included Silver Fabric Readme for the latest prerequisites required for this enabler.

Procedure

1. Get the required grid libraries for the installation.

   The Tomcat enabler consists of an enabler runtime grid library and a distribution grid library. The enabler runtime contains information specific to a Silver Fabric version that is used to integrate the enabler, and the distribution contains the application server or program used for the enabler.

   Enabler runtime grid libraries are downloaded from the TIBCO download site. Distribution grid libraries are created by referring the directions in Creating distribution Grid Libraries. Note that Distribution Grid Libraries are not provided by TIBCO.

   The following required grid libraries for each version of the Tomcat enabler are listed:

Tomcat Version	Required Grid Libraries
8.5	`TIB_sftm_6.0.0_tomcat_enabler.tar.gz` `tomcat_8.5_distribution_gridlib.zip`

2. If you are upgrading enabler, stop any active components that is currently using the Tomcat enabler.
3. Copy the desired Tomcat enabler version grid library files to the *SF_HOME*`/webapps/livecluster/deploy/resources/gridlib` directory in the following order:
   - Distribution
   - Enabler

   Copying the grid libraries files to this directory also extracts them to the deployed directory *SF_HOME*`/webapps/livecluster/deploy/expanded/`. This overwrites any changes to the existing grid library in the staging directory. TIBCO Silver® Fabric installs into a directory within *TIBCO_HOME*. This directory is referenced in documentation as *SF_HOME*. The default value of *SF_HOME* depends on the operating system. For example on Windows systems, the default value is `C:\tibco\fabric`. *TIBCO_HOME* refers to a common location where TIBCO products are installed.
4. Verify successful installation by selecting **Stacks > Enablers** in the Silver Fabric Administration tool and ensuring that the enabler is displayed in the list.

Configuring the Tomcat Enabler

The following topics describe how to set up Silver Fabric to run Tomcat applications.

Running Tomcat Applications in Standalone Mode

The following procedure describes how to run a Tomcat component in standalone mode. In this mode, Silver Fabric engines run individual self-contained servers that do not communicate with each other. On activation, the engine deploys archives contained in the component to the server.

Procedure

1. Enter your Silver Fabric host and port in your browser and log in to the Silver Fabric Administration tool.
2. Go to **Stacks > Components**.
3. Select **Create New J2EE Component** from the **Global Actions** list.
4. If prompted, select the **Tomcat Enabler** and desired **Enabler Version** from the available enablers.
5. Enter a component name in the **Name** field and an optional description in the **Description** field.
6. Click **Next**
   The **Configure Component Features** page is displayed.
7. HTTP Support, Archive Management Support, and Application Logging Support are added to the feature list by default. Make sure that the Clustering Support option is not selected. You can further customize component features as needed by selecting the feature and clicking **Edit**. When finished, click **Next**.
   The **Configure Component Options** screen is displayed.
8. Click **Next** until you reach the **Upload, remove, or reorder archive files** page.
9. Click **Add** and upload the archives (WAR file only) that you want to deploy and run on the Tomcat server.
10. Configure the options on the remaining screens as needed. Additional options are also configurable through context variables.
11. Click **Finish**.
12. Go to **Stacks > Components**.
13. Select **Publish Component** from the corresponding Actions list of the component you created.
14. Create a stack and add the component to the stack as described in the *Silver Fabric Cloud Administration Guide*. Start the stack when required.

Running Clustered Tomcat Applications

You can configure Silver Fabric to run with multiple Tomcat Partitions (clusters). In this configuration, engines running a Tomcat J2EE component act as nodes in a cluster. Membership in a particular cluster is determined by a node's `CLUSTER_MCAST_ADDR` and `CLUSTER_MCAST_PORT`. The cluster name provided while creating a component using the Component Wizard is provided as a system property to the Tomcat process and is the name used in the `cluster.xml` file. Naming is a convenience to the administrator, but does not determine cluster membership.

When running multiple clustered Tomcat components concurrently, configure each component with a unique cluster name, and a unique multicast address and port. To change the multicast port and address, you can set the `CLUSTER_MCAST_ADDR` and `CLUSTER_MCAST_PORT` runtime context variables on the **Add/Edit Enabler and Component-specific Runtime Context Variables** page in the Component Wizard. See Tomcat Runtime Context Variables for the default values of these variables.

Procedure

1. Start Silver Fabric and log into the Silver Fabric Administration Tool.
2. Go to **Stacks > Components**.
3. Select **Create New J2EE Component** from the **Global Actions** list.
4. If prompted, select the Tomcat enabler and desired enabler version from the available enablers.
5. Enter a component name.
6. Click **Next**.
   The **Configure Component features** screen is displayed.
7. Select the **Clustering support** option from the **Add feature** list. The **Enter feature values** screen is displayed.
8. Enter the cluster name in the field and click **OK**.
9. Click **Next** again.
   The **Configure Component options** screen is displayed.
10. Select the desired options.
11. Click **Next** until you reach the **Add/Edit Archive Files** screen.
12. Click **Add and upload the archives** (WAR file only) that you want to deploy and run on the Tomcat server cluster.
13. Configure the options on the remaining screens as needed. Additional options are also configurable through context variables.
14. Click **Finish**.
15. Select **Publish Component** from the Actions list to deploy the component.

Configuring SSL for Tomcat Components

You can configure SSL on your component to specify whether the Tomcat Server listens on HTTP or HTTPS. The Tomcat enabler included with Silver Fabric uses a default keystore with certificates that have been signed by the TIBCO certificate. Since the certificates are in the same certificate chain, they are trusted by the Engines without modification.

By default, the location of the server keystore and server trust store are in the files, at

`${CATALINA_HOME}/conf/tomcat.keystore` and `${CATALINA_HOME}/conf/tomcat.truststore`. This allows for all Silver Fabric components to trust the application server as well.

When broker-engine communication is over SSL, provide Java SSL properties before the engine starts. Go to **Config > Engines** to specify engine JVM command-line arguments. For example,

`-Djavax.net.ssl.keyStore=yourKeystoreFile`.

Configuring One-Way SSL

One-way SSL is the most common, standard implementation of SSL in client / server connections. In this mode, when a client attempts to connect with the server, the server offers the client a signed certificate. This certificate can be self-signed or signed by a Certificate Authority (CA). If the CA is trusted by the client in its local trust store, and the certificate is validated, or if the client is configured to accept the self-signed certificate, the connection is established.

To enable SSL in your Silver Fabric Component:

Procedure

1. On the Silver Fabric Administration Tool, go to **Stacks** > **Components**.
2. Select the **Edit Component** action for to your Component.
3. Select **Add/Edit Component Features**.
4. Select the **HTTP Support** feature and click **Edit**.
5. Select the **HTTPS Enabled** option and clear the **HTTP Enabled** option to create a pure SSL configuration.
6. Click **OK**.
7. The server uses the demo keystore bundled with Tomcat by default. If this is sufficient for your needs, click **Finish** and you are done.
8. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables** and configure the following environment variables as necessary to use a custom certificate:

   - `SERVER_KEY_STORE_FILE` — Server keystore file location for incoming SSL connections. In the definition, you can use the variable `${CATALINA_HOME}`, which expands to the Tomcat application server home directory for the enabler at run time.
   - `SERVER_KEY_STORE_PASSWORD` — Password for the server keystore.
   - `CLIENT_TRUST_STORE_FILE` — Trust store file used when connecting to the JMX server.
   - `CLIENT_TRUST_STORE_PASSWORD` — Client trust store password used when connecting to the JMX server.

   Click **OK** when finished. You can now upload your keystore files.
9. Select **Add/Override/Customize Enabler and Component-Specific Content Files**.
10. Click **Upload**.
    The **Add File** screen is displayed.
11. Complete the screen as follows:
    a) Enter a name for your server keystore file in the **Name** field.
    b) Enter the path you used for `SERVER_KEY_STORE_FILE` above in the **Relative Path** field.
    c) Enter your server keystore file in the **File** field.
    d) Click **OK**.
12. Repeat the previous two steps to upload `CLIENT_TRUST_STORE_FILE`.
13. Click **Finish**.
14. Select **Publish Component** from the Actions list to deploy the component.

Configuring Two-Way SSL

In two-way SSL, the Tomcat server additionally tries to establish trust with the connecting client by requesting a certificate from the client, and either accepting or rejecting it based on its own trust settings.

Follow the applicable instructions later in the section, to enable two way SSL in your Silver Fabric component.

Procedure

1. If a VirtualRouter instance is forwarding requests to this enabler, ensure it has SSL enabled.
2. On the Silver Fabric Administration Tool, go to **Stacks** > **Components**.

3. Select the **Edit Component** action for to your component.
4. Select **Add/Edit Component Features**.
5. Select the **HTTP Support** feature and click Edit.
6. Select the **HTTPS Enabled** option and clear the **HTTP Enabled** option to create a pure SSL configuration.
7. Click **OK**, then click Menu.
8. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables**.
9. Select **Add From Enabler**.
10. Select `TWO_WAY_SSL_ENABLED` from the list and click **OK**.
11. Click `TWO_WAY_SSL_ENABLED`, click **Edit**, change the value to true and click **OK**.
12. The server uses the demo keystore bundled with Tomcat by default. If this is sufficient for your needs, click **Finish** and you are done.
13. Configure the following environment variables as necessary to configure a custom certificate:
    - `SERVER_KEY_STORE_FILE` : Server keystore file location for incoming SSL connections.
    - `SERVER_KEY_STORE_PASSWORD` : Password for the server keystore.
    - `SERVER_TRUST_STORE_FILE` : Server trust store file location for outgoing SSL connections.
    - `SERVER_TRUST_STORE_PASSWORD` : Password for the server trust store.
    - `CLIENT_KEY_STORE_FILE` : Client keystore file used when connecting to the JMX server when `TWO_WAY_SSL_ENABLED` is true.
    - `CLIENT_TRUST_STORE_FILE`: Trust store file used when connecting to the JMX server.
    - `CLIENT_KEY_STORE_PASSWORD` : Client keystore password used when connecting to the JMX server when `TWO_WAY_SSL_ENABLED` is true.

    In the definitions, you can use the variable `${CATALINA_HOME}`, which expands to the Tomcat application server home directory for the enabler at run time.

    Click **OK** when finished. You can now upload your keystore file.
14. Select **Add/Override/Customize Enabler and Component-Specific Content Files**.
15. Click **Upload**.
    The **Add File** screen is displayed.
16. Complete the screen as follows:
    a) Enter a name for your server keystore file in the **Name** field.
    b) Enter the path you used for `SERVER_KEY_STORE_FILE` above in the **Relative Path** field.
    c) Enter your server keystore file in the **File** field.
    d) Click **OK**.
17. Repeat the previous two steps to upload `SERVER_TRUST_STORE_FILE`, `CLIENT_TRUST_STORE_FILE`, and `CLIENT_KEY_STORE_FILE`.
18. Click **Finish**.
19. Select the **Publish Changes** action.

Configuring HTTP/2 Support

Prerequisites

The Apache Portable Runtime (APR) Library has to be installed on the engine host. The default location of the library is

- Linux: `/user/local/apr/lib`
- Windows: `$CATALINA_HOME/bin`

If the APR library is installed on non-default location, *APR_LIB_PATH* Runtime context variable can be set to point to correct location. For more information on installation of APR, refer: Apache Tomcat Native Library Documentation.

Perform the following steps to configure HTTP/2 support:

Procedure

1. On the Silver Fabric Administration Tool, go to **Stacks** > **Components**.
2. Select the **Edit Component** action for to your component.
3. Click **Add/Edit Component Features**.
4. Select the **HTTP Support** feature and click **Edit**.
5. Select **HTTP/2 Enabled** check box on the pop-up that is displayed. Clear any other HTTP configuration check boxes then press **OK**.
6. Select **Add/Override/Edit Enabler and Component-Specific Runtime Context Variables** and configure the following environment variables as necessary to use a custom certificate:

   - `CERTIFICATE_FILE` — File location of the server certificate.
   - `CERTIFICATE_KEY_FILE` — File location of the server certificate key.
   - `APR_LIB_PATH` — HTTP/2 protocol requires the Apache Portable Runtime (APR) library installed on the enabler to work. The variable points to the directory where the APR library is installed. This variable is set with default values `/usr/local/apr/lib` for Linux and Solaris, and it is set as `$ {CATALINA_HOME}/bin` for Windows. New values can be set for this variable to point it to a custom directory.

   Click **OK** when finished. You can now upload your keystore files.
7. Select **Add/Override/Customize Enabler and Component-Specific Content Files**.
8. Click **Upload**.
   The **Add File** screen is displayed.

About Archive Management

The Tomcat enabler can dynamically manage the set of application archives that are deployed and running on Tomcat servers. This consists of archive detection, context URL detection, archive scaling, and continuous deployment operations.

- Archive Detection - The Tomcat enabler regularly detects the current set of archives that are deployed and running on the server. On the broker, this list is reflected in the ActivationInfo associated with the engine. The `ARCHIVE_DETECTION_ENABLED` and `ARCHIVE_DETECTION_FREQUENCY` runtime context variables can be modified to control whether or not polling of archives is performed, and how often this detection happens.
- Context URL Detection - The Tomcat JMX server is queried for the current set of context URLs associated with running web applications. This set of URLs is reported to the broker and updated in VirtualRouter.
- Archive Scaling - Archive scaling is a method of dynamically adding or removing archives from running environments, without affecting what is already running; these archives can then scale up or down as per archive-specific allocation rules. See the *Silver Fabric Administration Guide* for more details on creating Scaling rules.

- Continuous Deployment - Users can deploy, start, stop, and undeploy application archives to Tomcat by using the archive-level API operations provided by the Silver Fabric broker through REST, Ant, and the Command Line Interface (CLI).

About Using the Tomcat Manager

Tomcat distribution includes manager web application.

A default user with the user name tomcat and password tomcat is created during installation. It belongs to the roles `manager-gui`, `manager-status`, `manager-script`, and `manager-jmx`.

For information on using the manager app, see the Apache Tomcat documentation, available at the Apache's website.

Tomcat's manager application are installed in the `ROOT` webapps directory. When a component explicitly defines an application to be a `ROOT` context application by setting the `ROOT_CONTEXT_WEBAPP` variable, that application deletes and overwrites the default `ROOT` directory. The following warning is logged when a component deletes the `ROOT` application directory:

```
"WARNING: [TomcatContainer] Webapp directory ROOT already exists, it will be
overwritten with <ROOT_CONTEXT_WEBAPP> as the root context"
```

Post Configuration

After configuring and activating your Silver Fabric Tomcat Component, you can access the applications running on the Tomcat Servers. You can also add load balancing to your Component , if required.

Verifying Component Configurations

Verify your component configurations by performing the following steps:

Procedure

1. Go to **Engines > Engines** in the Silver Fabric Administration Tool and ensure that at least one instance of your component is running on an engine.
2. Go to **Admin > VirtualRouter**. The VirtualRouter page contains a table for each VirtualRouter client that is currently running. Select the `VirtualRouter Properties` action next to each client to show the component each host is running.
3. When you find the client running your component, select the **Status Page** action. The **VirtualRouter Status page** is displayed.
4. In the Relative URLs column, entries for each web application are listed. Click a URL to connect to the web application. HTTP requests made when the URL is clicked in the component-level Relative URLs column, are sent to VirtualRouter, which forwards the request to an engine running the component. HTTP requests made when the URL is clicked on the engine-level Relative URLs column are sent directly to that engine.

    - Alternatively, you can mimic the behavior of the VirtualRouter Status page by directing your browser to the address/port of your Silver Fabric broker with a relative URL from your component, such as `http://myserver:8080/myapp/index.html` (where *myapp* is one of the deployed applications). If you cannot access your Web application, verify your component configuration. See Configuring the Tomcat Enabler.
    - For testing purposes, you can also directly access applications running on Silver Fabric engines. The listen port of the server is generally the value of the relevant run time context variable for the component (`HTTP_PORT` or `HTTPS_PORT`) plus the engine instance number. For example, `HTTP_PORT` is by default 9090, so a server running on an engine `mymachine-2` is listening on port 9092.

About Load Balancing

After configuring and activating your Silver Fabric Tomcat Component, you can access the applications running on the Tomcat Servers. You can also add load balancing to your Component , if required.

You can use load balancing for proxying requests from clients and routing them to engines running the appropriate component. Load balancing is achieved through mappings between relative URLs and components.

VirtualRouter

VirtualRouter is load balancer for HTTP-enabled components. An instance of VirtualRouter runs by default on each Silver Fabric broker, and can also run externally. Additionally, VirtualRouter maintains HTTP Session mappings between clients and Tomcat servers by examining the standard `JSESSIONID` cookie. For more information about using VirtualRouter, see the *TIBCO Silver Fabric Installation Guide*

Custom External Load Balancer

You can create a custom load balancer to distribute requests across your component. Accommodating the dynamic nature of Silver Fabric allocation requires a mechanism to update the routing list of the

custom balancer when the Silver Fabric allocation changes. Notification of activation or deactivation events can occur in the following ways:

- ServerHook events
- SNMP traps
- Web Service calls

For more information on configuring and using these methods refer to the *Silver Fabric Developer's Guide*.

Statistics

The following statistics are the default statistics supported by the Tomcat enabler. The enabler uses JMX to retrieve statistic values from MBean attributes on the Tomcat server. You can select and track these statistics from the Component Wizard. Tracked statistics are available for report output. You can also create policy rules based on any tracked statistic.

Apache Tomcat 8.0 Statistics

The following statistics are available from Apache Tomcat application servers:

Name & Description	MBean/Attribute	Unit
Tomcat Busy Thread Count The number of busy Threads	Aggregated from the following three underlying statistics: Catalina:type=ThreadPool,name=http-nio-$ {container.get BindAddressPrefixForMBeans()}$ {container.getHTTPPort( )}/ currentThreadsBusy Catalina:type=ThreadPool,name=http-nio-$ {container.get BindAddressPrefixForMBeans()}$ {container.getHTTPSPor t()} / currentThreadsBusy Catalina:type=ThreadPool,name=ajp-nio-$ {container.getBi ndAddressPrefixForMBeans()}$ {container.getAJPPort()} / currentThreadsBusy	Threads
Tomcat Throughput The number of requests processed per second.	Aggregated from the following three underlying statistics: Catalina:type=GlobalRequestProcessor,name=htt p-nio-$ {container.getBindAddressPrefixForMBeans()}$ {container.getHTTPPort()} / requestCount Catalina:type=GlobalRequestProcessor,name=htt p-nio-$ {container.getBindAddressPrefixForMBeans()}$ {container.getHTTPPort()} / requestCount Catalina:type=GlobalRequestProcessor,name=ajp -nio-$ {container.getBindAddressPrefixForMBeans()}$ {container.get AJPPort()} / requestCount	Requests per second

Name & Description	MBean/Attribute	Unit
Tomcat Data Throughput The number of bytes sent per Second.	Aggregated from the following three underlying statistics Catalina:type=GlobalRequestProcessor,name=http-nio-${container.getBindAddressPrefixForMBeans()}${container.getHTTPPort()} / bytesSent Catalina:type=GlobalRequestProcessor,name=http-nio-${container.getBindAddressPrefixForMBeans()}${container.getHTTPSPort()} / bytesSent Catalina:type=GlobalRequestProcessor,name=ajp-nio-${container.getBindAddressPrefixForMBeans()}${container.get AJPPort()} / bytesSent	Bytes per second
Tomcat Active Sessions The number of active sessions.	Catalina:type=Manager,* / activeSessions	Sessions
Tomcat Cache Size The size of the cache.	Catalina:type=Cache,host=${VIRTUAL_HOST_NAME},* / cacheSize	MB
Tomcat Cache Hit Frequency The number of cache hits per second.	Catalina:type=Cache,host=${VIRTUAL_HOST_NAME},* / hitsCount	Hits per second
Tomcat JVM Used Memory Used heap memory.	java.lang:type=Memory / HeapMemoryUsage:used	MB
Tomcat JVM Committed Memory Committed heap memory.	java.lang:type=Memory / HeapMemoryUsage:committed	MB
Tomcat JVM Thread Count The number of threads.	java.lang:type=Threading / ThreadCount	Threads

Apache Tomcat 8.5 Statistics

The following statistics are available from Apache Tomcat application servers:

Name & Description	MBean/Attribute	Unit
Tomcat Busy Thread Count The number of busy Threads	Aggregated from the following three underlying statistics: Catalina:type=ThreadPool,name=${container.getHTTPProtocolHandlerName()} / currentThreadsBusy Catalina:type=ThreadPool,name=${container.getHTTPSProtocolHandlerName()} / currentThreadsBusy Catalina:type=ThreadPool,name=${container.getAJPProtocolHandlerName()} / currentThreadsBusy	Threads
Tomcat Throughput The number of requests processed per second.	Aggregated from the following three underlying statistics: Catalina:type=GlobalRequestProcessor,name=${container.getHTTPProtocolHandlerName()} / requestCount Catalina:type=GlobalRequestProcessor,name=${container.getHTTPSProtocolHandlerName()} / requestCount Catalina:type=GlobalRequestProcessor,name=${container.getAJPProtocolHandlerName()} / requestCount	Requests per second
Tomcat Data Throughput The number of bytes sent per Second.	Aggregated from the following three underlying statistics: Catalina:type=GlobalRequestProcessor,name=${container.getHTTPProtocolHandlerName()} / bytesSent Catalina:type=GlobalRequestProcessor,name=${container.getHTTPSProtocolHandlerName()} / bytesSent Catalina:type=GlobalRequestProcessor,name=${container.getAJPProtocolHandlerName()} / bytesSent	Bytes per second
Tomcat JVM Used Memory Used heap memory.	java.lang:type=Memory / HeapMemoryUsage:used	MB
Tomcat JVM Committed Memory Committed heap memory.	java.lang:type=Memory / HeapMemoryUsage:committed	MB

Name & Description	MBean/Attribute	Unit
Tomcat JVM Thread Count The number of threads.	java.lang:type=Threading / ThreadCount	Threads

Archive Level Statistics

The following statistics are reported at the archive level for finer grain reporting information that can be used for archive scaling:

Name & Description	MBean/Attribute	Unit
Tomcat Archive Throughput	Catalina:j2eeType=Servlet,name=<servlet>,WebModule=<webmodule>,J2EEApplication=none,J2EEServer=none / requestCount	Requests per second
Tomcat Active Archive Sessions	Catalina:type=Manager,context=<path>,host=<listen address> / activeSessions	sessions

Runtime Context Variables

The following is a comprehensive list of all runtime context variables used by the Tomcat enabler.

Variable	Type	Description
ALLOW_NAME_ONLY_COOKIE_FLAG	Environment	Specify if non-specification compliant name-only cookies are accepted by Tomcat. (7.0 and later version only)
APR_LIB_PATH	Environment	HTTP/2 protocol requires the Apache Portable Runtime (APR) library installed on the enabler to work. The variable points to the directory where the APR library is installed. This variable is set with default values `/usr/local/apr/lib` for Linux and Solaris, and it is set as `${CATALINA_HOME}/bin` for Windows. New values can be set for this variable to point it to a custom directory.
ARCHIVE_DEPLOYMENT_TIMEOUT	String	The amount of time in seconds to allow for each archive deployment to finish in standalone mode. Valid only if VERIFY_ARCHIVE_DEPLOYMENT_SUCCESS is true.
ARCHIVE_DETECTION_ENABLED	String	Archive detection is enabled only if this variable is defined and set to true.
ARCHIVE_DETECTION_FREQUENCY	String	Frequency in seconds at which archive detection is triggered. This variable is used only when ARCHIVE_DETECTION_ENABLED is true. Minimum allowed value is 30. If the value specified is lower than the minimum allowed, it resets to the minimum (30 seconds).
BIND_ON_ALL_LOCAL_ADDRESSES	Environment	Set to true to have Tomcat bind to all local IP addresses.
CAPTURE_AS_PROPERTY	String	A file ending with one of these comma separated list of trailing names is considered a properties file at capture time.
CAPTURE_INCLUDES	String	Directories that require to be captured. Generally these directories contain configuration, applications, and so forth.

Variable	Type	Description
CATALINA_HOME	Environment	Tomcat App server home
CATALINA_OPTS	Environment	Additional Catalina options to be used in startup and shutdown scripts.
CERTIFICATE_FILE	Environment	File location of the server certificate.
CERTIFICATE_KEY_FILE	Environment	File location of the server certificate key.
CLIENT_KEY_STORE_FILE	String	Client keystore file used when connecting to the JMX server when TWO_WAY_SSL_ENABLED is true.
CLIENT_KEY_STORE_PASSWORD	String	Client keystore password used when connecting to the JMX server when TWO_WAY_SSL_ENABLED is true.
CLIENT_TRUST_STORE_FILE	String	Trust store file used when connecting to the JMX server
CLIENT_TRUST_STORE_PASSWORD	String	Client trust store password used when connecting to the JMX server.
CLUSTER_DISTRIBUTABLE_FLAG	String	True or False flag which enables/ disables session replication at the global context level.
CLUSTER_MCAST_ADDR	String	MCast address to use for cluster membership broadcasts.
CLUSTER_MCAST_PORT	String	Port to use with MCast address for cluster membership broadcast.
CLUSTER_REPL_LISTENER_PORT	String	Port to use when listening for session replication requests.
CLUSTER_STICKY_SESSION_SUPPORT_ ATTR	String	The attribute to be added to the engine element with name=Catalina to allow/disallow sticky session.
COMPONENT_NAME	Environment	Tomcat Component name
DEFAULT_SSL_ALGORITHM	String	Default algorithm for the SSL/TLS Connector
DISTRIBUTION_NAME	Environment	Use this variable to change the distribution name on grid-library.xml.

Variable	Type	Description
DISTRIBUTION_VERSION	Environment	Use this variable to change the distribution version on `grid-library.xml`
ENABLE_JMX_AUTH	Environment	True if security is used when connecting to Tomcat JMX server instance.
HOST_AJP_PORT	String	IP port number used for AJP/1.3 protocol
HOST_JMX_PORT	Environment	IP port number used for JMX connector
HOST_SHUTDOWN_PORT	String	IP port number used to shut down app server
HTTPS_PORT	Environment	HTTPS listen port
HTTP_PORT	Environment	HTTP listen port
HTTP2_ENABLED	Environment	If `HTTP2` protocol is supported
J2EE_ARCHIVE_DEPLOY_DIRECTORY	String	Location of deployed applications
JDK_NAME	String	The name of the required JDK
JDK_VERSION	String	The version of the required JDK
JMX_MONITOR_ROLE	String	Role name to use when connecting to the Tomcat JMX server instance.
JMX_MONITOR_ROLE_PWD	String	Role name password to use when connecting to the Tomcat JMX server instance.
LISTEN_ADDRESS_NET_MASK	Environment	A comma-delimited list of net masks in CIDR notation. The first IP address found that matches one of the net masks is used as the listen address. Note that `BIND_ON_ALL_LOCAL_ADDRESSES` overrides this setting.
ROOT_CONTEXT_WEBAPP	String	Specify the name of the archive, including file extension, to run as the `ROOT` context.
SERVER_KEY_STORE_FILE	Environment	Server keystore file location for incoming SSL connections.
SERVER_KEY_STORE_PASSWORD	Environment	Password for the server keystore

Variable	Type	Description
SERVER_TRUST_STORE_FILE	Environment	Server trust store file location for outgoing SSL connections
SERVER_TRUST_STORE_PASSWORD	Environment	Password for the server trust store
SESSION_COOKIE_NAME	Environment	Specify the name of the cookie used for session tracking
TWO_WAY_SSL_ENABLED	Environment	If clientAuth is enabled in the SSL or TLS Connector
USE_HTTP_ONLY_COOKIE_FLAG	String	Specify if the HttpOnly flag is set on session cookies to prevent client-side script accessing session ID. (7.0 and later version only)
VERIFY_ARCHIVE_DEPLOYMENT_SUCCESS	String	Whether or not to verify successful deployment of J2EE archives. This must be set to `true` for archive ordering.
VIRTUAL_HOST_NAME	String	The name associated with the `server.xml` host element

Creating Distribution Grid Libraries

TIBCO does not provide the full installation of the Tomcat software. You must create the distribution by using your own copy of Tomcat.

Prerequisites

These files are required for creating the distribution grid libraries:

- Apache Tomcat Software: TIBCO does not provide the full installation of the Tomcat software to build the distribution. You must create the distribution by using your own copy of the Tomcat software downloaded from the Apache Tomcat website https://tomcat.apache.org/download-80.cgi .
- grid-libray.xml file: Refer example Tomcat Distribution `Grid-library.xml` file for a sample grid-library.xml file.

To create a distribution grid library:

Procedure

1. Create a temporary directory named `tomcat-x.y.z-distribution-gridlib`.
2. Download the full installation of the Tomcat software, and extract the .zip file into the temporary directory.
3. The extracted .zip file download of the Tomcat software contains a single directory with the name `apache-tomcat-x.y.z`. Rename this directory `tomcat`.
4. Create a `grid-library.xml` file by referring Example Tomcat Distribution Grid-library.xml File.
5. Put the `grid-library.xml` into the top level of the temporary directory.
   The temporary directory now contains the `grid-library.xml` and a `tomcat` directory.
6. Create an archive of the temporary directory. It can be a .zip file archive, or a gzipped TAR archive. The archive should contain the `grid-library.xml` and `tomcat` directory at the top level, and not the temporary directory itself.
7. Ensure that you name the archive as `tomcat-x.y.z-distribution-gridlib.zip` or `.tar.gz`. This is the completed distribution grid library for installation on your broker.

Example Tomcat Distribution Grid-library.xml File

The following is an example of a `grid-library.xml` file used in the creation of a distribution gridlibrary for Tomcat :

```
<?xml version="1.0" encoding="UTF-8"?>
<grid-library>
<grid-library-name>tomcat-distribution</grid-library-name>
<grid-library-version>8.5</grid-library-version>
</grid-library>
```