TIBCO Silver® Fabric

TIBCO Silver® Fabric Concepts

Software Release 5.8.1
August 2017

Two-Second Advantage®

Important Information

Contents

Preface

TIBCO Silver® Fabric combines the flexibility and scalability of the public cloud with the security and control of your own data center. It brings the elasticity of cloud computing to your organization – supporting existing solutions within your current infrastructure while automatically scaling resources to meet demand.

Topics

Related Documentation

This section lists documentation resources you may find useful.

For the latest version of documentation, including any changes or additions made since the last product release, please visit http://docs.tibco.com.

TIBCO Silver Fabric Documentation

The following documentation is included with Silver Fabric in Adobe Acrobat (PDF) format. To view the guides, log in to the Administration Tool and go to **Admin > Documentation**. The PDF files are also on the Broker at *SF_HOME*`/webapps/livecluster/admin/docs`. The following documents form the Silver Fabric documentation set:

- *Silver Fabric Concepts* Contains an introduction to Silver Fabric, including definitions of key concepts and terms, such as Enablers, Stacks, Components, Engines, and Brokers. Read this first if you are new to Silver Fabric.
- *Silver Fabric Installation Guide* Covers installation of Silver Fabric for Windows and Unix, including Brokers, Engines, and pre-installation planning.
- *Silver Fabric Cloud Administration Guide* Covers Silver Fabric cloud administration, configuration of Engines, Enablers, and Components, and configuration and use of Skyway. Also covers security, general maintenance, performance tuning, and database administration.
- *Silver Fabric Developer's Guide* Developer-related topics such as logging and debugging, using the Admin API, and the Enabler SDK.
- *Silver Fabric Developer's Tutorial* Tutorials for developers, such as how to write Enablers and Asset Managers.
- *Silver Fabric User's Guide* Covers Silver Fabric use and operation, including management of Engines, Enablers, Components, and Stacks.
- *Silver Fabric Skyway User's Guide* Covers usage of Skyway, which enables users to quickly and easily provision and manage their Silver Fabric Stacks.
- *Silver Fabric Tomcat Enabler Guide* Covers installation and configuration of applications run on the Tomcat Enabler.
- *Silver Fabric Command Line Enabler Guide* Covers installation and configuration of applications run on the Command Line Enabler.

Other Documentation and Help

Additional help and information is available from the following sources:

- *Silver Fabric Administration Tool Help* Context-sensitive help is provided throughout the Silver Fabric Administration Tool by clicking the Page Help button located on any page.
- *API Reference* Silver Fabric API reference information is available in the Silver Fabric SDK in the `api` directory in JavaDoc format. You can also view and search them from the Silver Fabric Administration Tool; log in to the Administration Tool and go to **Admin > Documentation**.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
TIBCO_HOME	Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as *TIBCO_HOME*. The default value of *TIBCO_HOME* depends on the operating system. For example, on Windows systems, the default value is `C:\tibco`.
SF_HOME	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`.
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [enable | disable]`
`italic font`	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms. For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*

Table 1 General Typographical Conventions (Continued)

Convention	Use
Key combinations	Key names separated by a plus sign indicates 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.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand param1 | param2 | param3`

Table 2 Syntax Typographical Conventions (Continued)

Convention	Use

`{ }`	A logical group of items in a command. Other syntax notations may appear within each logical group.
	
	For example, the following command requires two parameters, which can be either the pair `param1` and `param2`, or the pair `param3` and `param4`.
	
	`MyCommand {param1 param2} | {param3 param4}`
	
	In the next example, the command requires two parameters. The first parameter can be either `param1` or `param2` and the second can be either `param3` or `param4`:
	
	`MyCommand {param1 | param2} {param3 | param4}`
	
	In the next example, the command can accept either two or three parameters. The first parameter must be `param1`. You can optionally include `param2` as the second parameter. And the last parameter is either `param3` or `param4`.
	
	`MyCommand param1 [param2] {param3 | param4}`

Connecting with TIBCO Resources

How to Join TIBCO Community

TIBCO Community is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCO Community offers forums, blogs, and access to a variety of resources including product wikis that provide in-depth information, white papers, and video tutorials. In addition, users can submit and vote on feature requests via the Ideas portal. For a free registration, go to https://community.tibco.com.

How to Access All TIBCO Documentation

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

http://docs.tibco.com

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

Chapter 1 Silver Fabric Overview

TIBCO Silver® Fabric enables creation, configuration, provisioning, management, and monitoring of private clouds for the enterprise.

To meet the highly cyclical demands of large-scale business applications, enterprises traditionally take one of two approaches: deliver high performance by provisioning for peak load or forego service level agreements to a lower cost. Silver Fabric enables IT operations to have the best of both worlds: deliver high performance while optimizing costs.

Silver Fabric is hosted securely within your on-premise data center and seamlessly works with your existing hardware and software — quickly enabling you to create shared pools of computing resources within which your applications' footprints can dynamically expand and contract based on real-time metrics.

Topics

Dynamic Application Elasticity

Dynamic Application Elasticity involves transforming static enterprise applications to dynamically deploy them to hosts within the private cloud. within which your applications' footprints can dynamically expand and contract leverage existing applications and configure them for automated deployment to cloud resources is a minimal amount of reconfiguration.

You can then use an automated system to configure, publish, start, monitor, and manage applications based on policy. Since this eliminates the need to install and configure Stacks on individual machines, this also implies controlling a resource pool in a highly scalable manner.

This type of virtualization involves dynamic expansion or contraction of application capacity, based on schedule or demand. This closed-loop provisioning capability means the environment can automatically respond to changing business needs. The figure below illustrates the three fundamental aspects of application virtualization.

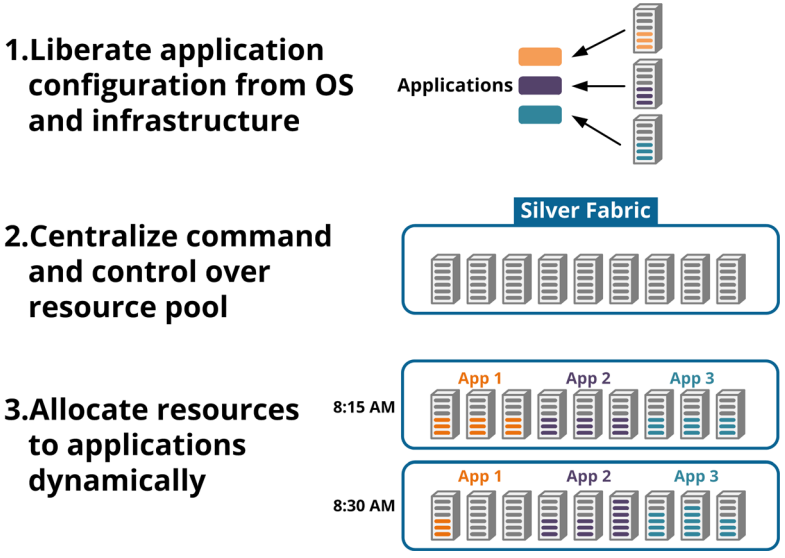

Figure 1 Application Virtualization Concepts.

Uses and Benefits

Application virtualization is a transforming technology for several use cases beyond incremental automation improvement. Silver Fabric automates manual tasks in a new paradigm transforming how application services are delivered to the business. The list below outlines the new capability benefits of application virtualization:

- **Self-Service** — Developers can provision their own platforms and deploy their own applications without IT involvement - reducing workload on your IT resources and speeding up the time it takes to deploy applications.
- **Elasticity** - Application capacity can grow or shrink based on demand. Thresholds for activating new application instances are based on system or application statistics monitoring. Unused applications automatically shrink to their minimum required level of capacity.
- **Shared Pool of Resources** — By pooling resources and automatically allocating resources based on demand or schedule, enterprises can dramatically reduce the amount of over-provisioning that is seen in data center. Silver Fabric increases infrastructure utilization and reduces the total number of servers required — ultimately reducing the management and overhead associated with application stacks.
- **Leverage Existing Resources** — Silver Fabric does not rely on virtualization technology and can leverage your existing infrastructure resources. Silver Fabric can have resources that can be a mix of virtual machines, native OS hosts and public cloud resources allowing you to gain maximum value from your existing investments.

Automated Application Management

Silver Fabric creates a utility computing operation by centralizing and automating all aspects of application provisioning, activation, monitoring, and administration. Different people within an organization are tasked with different activities. A utility operating model must provide the different roles and entitlements, but it must also provide transparency and reporting for creating diagnostic methodologies and charge back models.

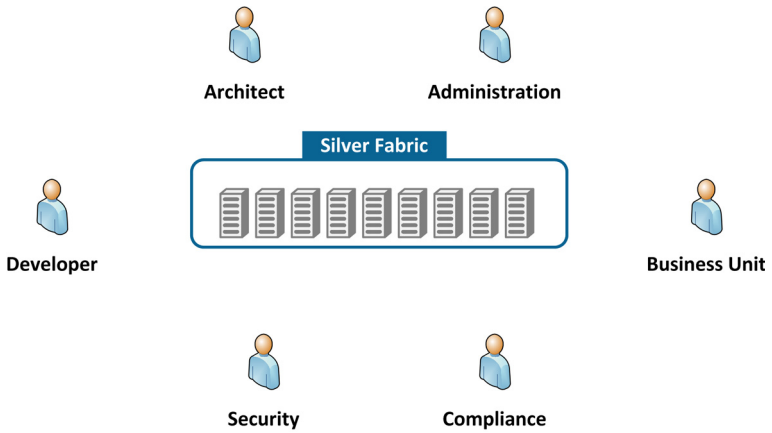

Figure 2 Multiple users and roles require different entitlements.

Configuration managers create and standardize distributions for runtimes, platforms, and application artifacts. Architects must understand the resource requirements and constraints for a given application, and create appropriate runtime policies. Administrators must monitor applications with transparency into the operations on remote machines. Therefore, uniform logging, auditing, and exception tracking are critical. Scheduling and coordination becomes an important capability to address with a utility computing model. Lastly, thorough reporting on usage is important for internal charge back and IT visibility.

Skyway

Skyway is a web-based interface that enables a developer to obtain a desired complex development platform to build franchise applications. Skyway users are provided Templates, each of which defines a Stack that can be easily provisioned to run an application. You can configure Templates before provisioning, and also capture changes made in the Stack's Components.

Skyway runs on each Broker. It uses the Broker's definitions of Stacks, Templates, and users.

Skyway can also be used as a templating system to enable operators to easily deploy applications to a private cloud. An application running in the cloud can be captured as a template which can then be modified and later provisioned. This enables operators to repeatedly instantiate templates to environments after making minor changes.

The Operations Management Environment

In addition to providing a virtualization model and runtime environment for applications, Silver Fabric also includes a highly manageable operating environment featuring a web-based console, the Silver Fabric Administration Tool. This lets you define, deploy and register Stacks and Components, manage the workloads running in the cloud, and configure the Silver Fabric environment. Uploading binaries and controlling versions is also straightforward.

The Administration Tool provides graphical runtime monitors to quantify cloud usage and current operations. Diagnostic tools also provide logging and error information for remote debugging purposes.

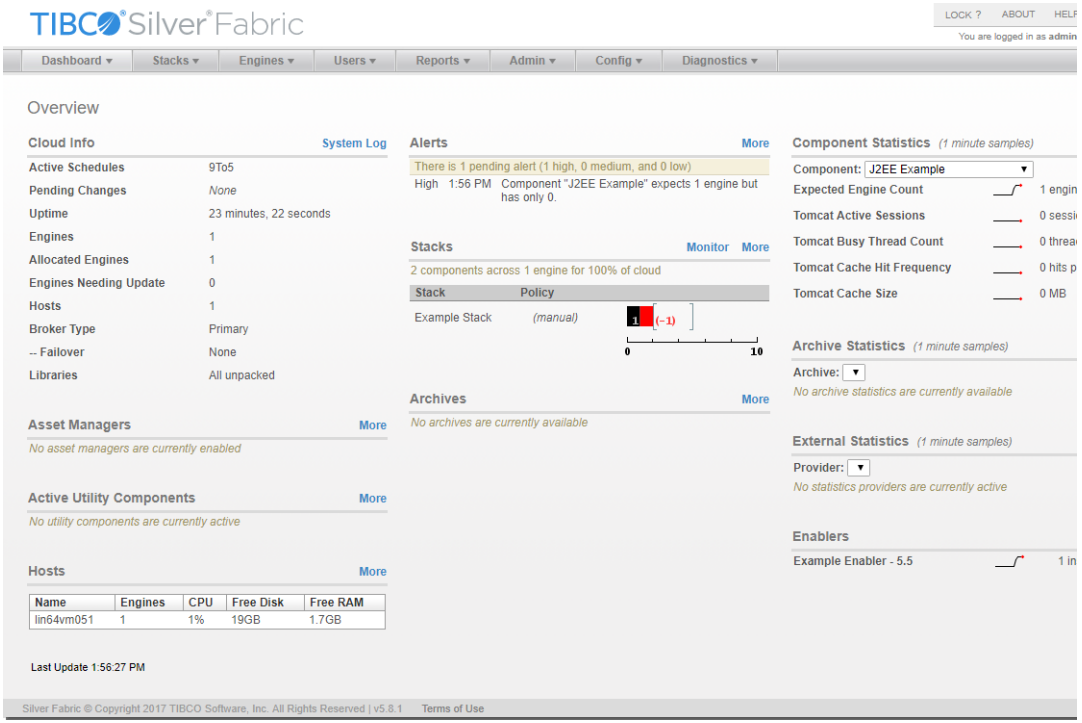

Figure 3 The Silver Fabric Administration Tool Dashboard.

TIBCO Silver® Fabric provides monitoring and management facilities for key metrics such as throughput, latency, resource usage, and exceptions. Simple user interfaces and subscription services provide access to current and historical information as well as Service Level Agreement (SLA) warnings and breaches.

Resource and Privilege Segmentation

Silver Fabric provides the ability to segment user privileges and resource access in two different ways: by account and by role.

Assets such as applications and compute nodes can be segmented vertically, by groups of users called accounts.

Accounts are a collection of users, with access limited to a subset of assets on the cloud. Accounts are typically used to "silo" users in different business units to their own set of objects in an enterprise-wide cloud. Users in an account can only access objects that belong to their account, which prevents them from interfering with other applications or data in the cloud.

A special account, simply named "cloud", exists by default and contains admin-level users that have access to all objects.

You can also segment user privileges horizontally by the use of **roles**. A role is a set of permissions assigned to a user that governs the ability to use, see, or access a Silver Fabric resource. An Administer role allows administrators access to any part of the cloud. Four other predefined roles allow more limited access, or you can create custom roles, and add or remove what permissions are be available to user accounts assigned that role.

Users, accounts, and roles can all be defined and administered in Silver Fabric's built-in authentication database. You can also map users and groups in your existing LDAP server to integrate with your current enterprise authentication system.

High-Level Silver Fabric Architecture

Silver Fabric virtualizes applications across a shared computing infrastructure. You can dynamically change the amount of resources dedicated to an application based on policy. This lets you utilize and share your excess capacity and grant priority access to the most important applications.

At the highest level, Silver Fabric consists of the following components:

- **Component** — one of the executable pieces, such as a utility service or J2EE web application, that makes up a Stack.
- **Engine** — the process that provisions and runs Components.
- **Enabler** — a controlling "wrapper" around an application, application server, or application platform that enables you to run it on Silver Fabric.
- **Stack** — a Component or a bundle of Components that is activated as a unit, plus one or more policies specifying the resource needs, priorities and constraints of those Components.
- **Broker** — the central component that deploys and activates applications on the remote machines of the utility infrastructure. The Broker also manages the policies you create to specify how and when the applications are activated on the utility resources.

For More Information

Management of the Silver Fabric platform is described in more detail in the *Silver Fabric Cloud Administration Guide*. Monitoring is described in more detail in Statistics and Monitoring on page 37.

Chapter 2 Enablers

An *Enabler* is a wrapper around an executable, typically an application server, and is responsible for life-cycle management of deployed applications. Enablers host Components, and configure, start, monitor, and stop Component processes. They also collect performance statistics pertaining to those Component processes.

A typical Enabler is a wrapper around a J2EE application server, such as Tomcat, JBoss, or WebLogic. However, Enablers are not J2EE-specific. Enablers have been developed to wrap Apache Web Server, Microsoft IIS, and TIBCO BusinessWorks, among many others.

The Silver Fabric installation includes the Tomcat Enabler by default. Additional Enablers can be obtained from TIBCO. Contact TIBCO for information about available Enablers.

You can also use the Enabler SDK to develop additional, custom Enablers and add them to existing Silver Fabric installations.

Topics

Enabler Contents

A Silver Fabric Enabler consists of an Enabler runtime and, optionally, one or more additional Enabler distribution archives.

The Enabler runtime is a package containing Silver Fabric-specific configuration and implementation. It contains a configuration file that defines changes to occur when an Enabler is activated on an Engine. For example, at run-time, the Enabler runtime may dynamically change URLs, port numbers, or other information in an application server's startup scripts, in order to virtualize the server to any machine.

The Enabler distribution is one or more packages containing the application server or other application that's run to host Components. Some Enablers, such as the Command Line Enabler, only require an Enabler runtime. In other Enablers, such as the included Tomcat Enabler, the Enabler distribution consists of the installation directory used to run the application server in a standalone configuration on a single machine, with minor changes, such as the removal of redundant files or changes to statically configured items.

Components, Enablers, and Distributions are packaged as *grid libraries* and uploaded to a Silver Fabric Broker and deployed to Engines. Grid libraries are compressed files and distributions that may be published by the Silver Fabric Broker to Silver Fabric Engines with specific configurations and policies defined by an Enabler and a Stack. Grid libraries, also known as gridlibs or archives, are stored in a central repository for reuse when a Component is published to a new Engine.

Enablers and Component Types

Enablers host Components on an Engine. Each Enabler supports one or more *Component Types*. A given Component can be launched by any Enabler that supports its type.

J2EE Components

A J2EE Component is a web application or web service that may require multiple running instances and is deployed as an EAR or WAR application archive into a J2EE application server Enabler.

For example, an existing J2EE application server cluster could be represented as a J2EE Component, with the size of the cluster varying between two to five nodes, based on policy. The physical locations of the Component instances are determined by the infrastructure at runtime and the resources are provided dynamically. Silver Fabric then instantiates this user-defined Component on these resources and centrally monitors its performance. Stack policies dictate how many instances to create, the kinds of resources to use for them, and when to add or remove instances.

Silver Fabric includes by default a Tomcat Enabler that can be used to run J2EE Components. Enablers that wrap certain versions of other J2EE application servers (including WebLogic, WebSphere, and JBoss) can be obtained from TIBCO. If you have multiple J2EE Enablers, you can separately associate each J2EE Component with one of the available J2EE Enablers as part of configuring the J2EE Component.

Command Line Components

Any executable that can be run from a command line can be virtualized as a Component by way of the Command Line Enabler, which ships as part of Silver Fabric. This provides a quick way to obtain many of the advantages of application virtualization with a minimum of effort.

Other Component Types

Specialized Enablers can sometimes prove useful. Silver Fabric's Enabler SDK is a toolkit that allows you to develop custom Enablers.

For More Information

For more information on developing Enablers, see the *Silver Fabric Developer's Guide*.

Chapter 3 Components

TIBCO Silver Fabric *Components* are configured executables that have been virtualized with Enabler configurations. Components can then be published as parts of a Stack for running on Silver Fabric Engines. Many TIBCO products, utility services, and J2EE web applications have been productized as distinct components for publishing multiple scalable instances on Engines as parts of the TIBCO Silver Fabric Cloud.

A policy specifies the minimum and maximum number of processes to run at a given time. If a Component has two running instances, it means that two physical processes are running (automatically activated on different machines if possible), which are load balanced by an external or internal load balancer. Silver Fabric automatically reconfigures each instance of a Component to reflect aspects of the environment (such as host names, port numbers, or work directories) that are unique to that instance.

Topics

Creating and Deploying Components

Silver Fabric simplifies application management by storing all information pertaining to application servers, versions, customizations, and application code in one logical place, the Broker. Multiple configuration files for different computers or cluster members are not required as Silver Fabric supports a one-to-many paradigm. All Component customizing is done in the Silver Fabric Administration Tool, using the Component Wizard.

Features

Each Enabler defines set of behaviors it supports; each supported item is called a *feature*. For example, the Tomcat Enabler supports HTTP routing as a feature by default.

When you create a Component, you can then specify which Enabler features you will use. For example, if your Enabler supports clustering as a feature, you can add clustering support and configure which cluster your Component will use.

Variables

Enablers and Components feature *runtime context variables*, which can be used to help you virtualize an application. Enablers define a set of string and environment variables, which are inherited by each Component. If a Component has a dependency on another Component, it will also inherit that Component's exported variables. These variables can then be used by the Enabler or Component to ensure the application can be easily virtualized.

For example, the Tomcat Enabler provides variables for the Tomcat instance name, HTTP port, JDK name and version, and more. If you create a Component for your J2EE application running in the Tomcat Enabler, it will inherit all of the Enabler's variables. If you add a dependency on a database defined in another Component which contains the database's JDBC URL in an exported variable, you can then use that variable in the application's Component to point it to the database.

You can also create an encrypted context variable instead of hard-coding passwords in Enabler or Component configuration files; it is stored encrypted on the Broker and decrypted when used on the Engine. There is also an auto-increment variable type. This enables you to define a variable that has the Engine instance number added or appended to the value. For example, you could define a port number as being 8080 plus the Engine instance number, so running instances of an Enabler would listen on ports 8080, 8081, and so on.

You can easily add, edit, or remove variables in the Silver Fabric Administration Tool, either at the Enabler level, where they are inherited by all Components running in that Enabler, or at the Component level.

Allocation

Allocation is the number of CPUs or Engine instances given to running published Components. Allocation constraints may be set to limit the number of Engines or CPUs dedicated to running a particular Component on a schedule or based on metrics.

Configure the default allocation for Components and Components within Stacks to specify how many instances and the relative priority that the Component can have. You can set allocation constraints on dependencies (on other Components or resource types in the runtime environment) and you can also specify conditions or schedules that can change the allocation to increase or reduce the availability of resources.

Deployment

Once you create and configure a Component, you can publish it with a single click. It's also easy to modify and re-publish Components on demand, making for highly scalable application management.

For more information on Components, see the *Silver Fabric Cloud Administrator's Guide*.

Chapter 4 Stacks

In some contexts, an executable standing alone is useful in its own right. More often, however, what's really useful is something that's built up out of several tiers of executables running simultaneously and in cooperation. A classic example of this might be a Code-Cache-Data web-based application, with a J2EE web application as its front-end, a database as its back-end and an in-memory cache between them. While there are tasks you might perform that focus on just one or another of these pieces, in most contexts you'd like to work with them as a unit.

Topics

Stacks

In Silver Fabric, each piece of an application, such as a database or J2EE web application, is defined with a Component. The unit tying these Components together is called a *Stack*.

At its simplest, a Silver Fabric Stack is just a bundle of Components. The value in this initial kind of bundling is that it provides joint activation. When Silver Fabric runs a Stack, it provisions default numbers of the various Components to Engines and activates those Components. When Silver Fabric stops a Stack, it deactivates those Components and makes the Engines on which they were running available to other Stacks. When you define Components, you specify default allocation rules, such as the number of Engines to allocate for the Component, and those will be copied into and used by each Stack including that Component.

Policies and Schedules

A Stack isn't limited to using Components' default allocation settings. You can change those settings at the Stack level by defining a *policy*. Policies are rules that can define component allocation settings within a stack. Policies can also set a dependency, enablement condition, resource preference, threshold activation, engine group minimum or maximum.

For instance, each Component has a default allocation priority, but when you add a Component to a Stack, you might want it to have a different priority in that Stack. You can do that by creating a policy that gives it a different priority within that Stack, while leaving its default priority intact for all other Stacks.

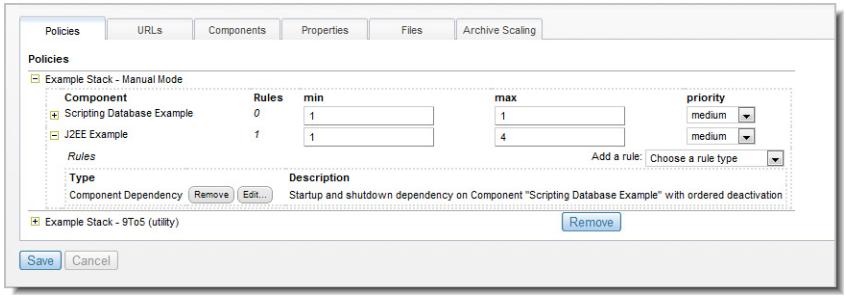

Figure 4 Defining policies for a Stack.

Policies can also contain rules that define advanced Engine activation behavior. For example, you can add an allocation rule stating that a Component cannot be activated until another Component in the Stack is already running. Or you can specify that a Component needs or prefers to run on Engines of a specific OS or hardware type.

If you need for your Stack to run differently at different times of day or different parts of the week or month, you can create *Schedules* that describe those different periods of interest. Schedules are periods of time that you define based on hourly, daily, weekly, or monthly periods.You should define distinct Schedules for each period of time for which you want different Stack behavior so that different policies will apply to each specifically defined Schedule. Once you define the intervals and recurrence within a Schedule, you can also share the definitions across any Stacks.

For example, you can define a Schedule to specify the intervals of 9:00 AM to 5:00 PM Monday through Friday, and another Schedule that indicates the weekend, such as all day Saturday and Sunday. One Stack can then add policies to the weekday Schedule to allocate more Engines during the business day, while another Stack can take more Engines during the weekend off-hours.

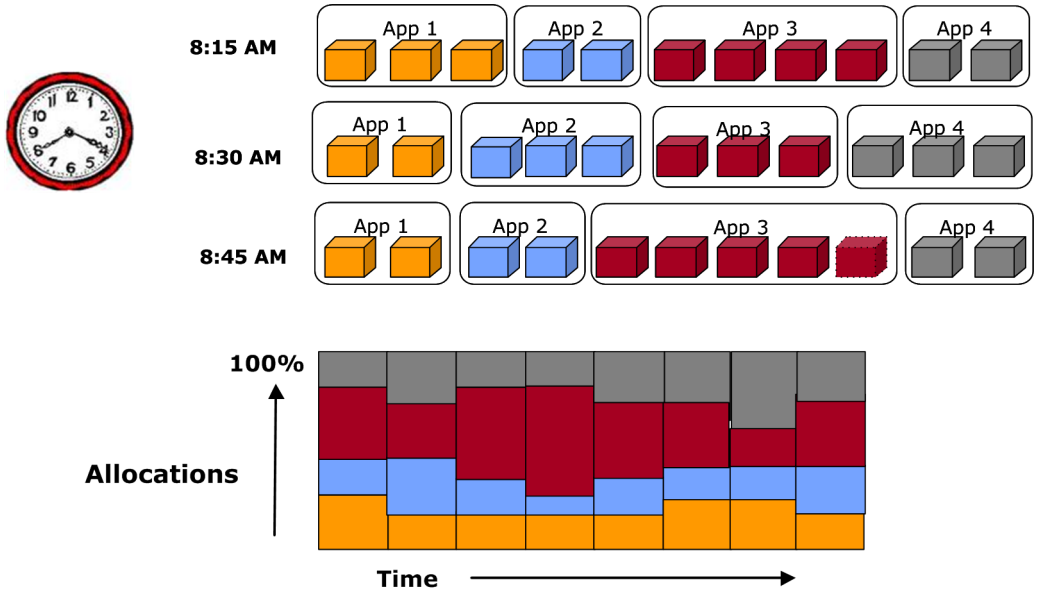

Figure 5 Scheduling changes in resource allocation policy.

Some components also support archive scaling and policies may be set to manage and scale component archives based on statistics gathered on the project archive level. Archive scaling is an elasticity framework for scaling archives or services within an Enabler. It enables you to deploy, start, stop, and undeploy application archives to Components without restarting the Components and disrupting the running applications. Archive Scaling is also known as micro-scaling.

For More Information

For more information on Stacks, see the *Silver Fabric Cloud Administrator's Guide*.

Chapter 5 Inside the Engine

The TIBCO Silver® Fabric *Engine* is a lightweight application that deploys, monitors and manages Stacks. Any given host can be used to run one or more Engine instances (and so, potentially, Component instances). Engine instances are started and managed by an agent that runs on the host called the *Engine Daemon*. The Engine Daemon starts Engine instances, monitors them, and restarts them if there are failures or reconfigurations.

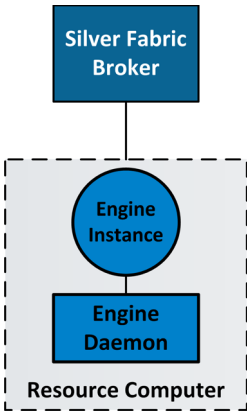

Figure 6 The TIBCO Silver® Fabric Engine.

Topics

The Engine Daemon

The *Engine Daemon* manages Engine instances on resource computers. It is the program you install on a computer to maintain Engine instances on that machine. It runs as a separate process from the Engine instances. One Engine Daemon runs per machine. It starts and stops one or more Engine instances based on configuration and machine state.

Aside from starting Engine instances, the Engine Daemon also performs configuration for Engine instances. The Engine Configuration is centrally managed on the Broker. When an administrator changes the Engine Configuration, the changes automatically propagate to Engine Daemon, which then apply the changes to Engine instances.

Installation and Configuration

Engine instances are only installed on a machine once. Engine configuration is centrally managed, and Engine instances can be automatically upgraded.

A 1-click installation is available from the Silver Fabric Administration Tool for 32-bit Windows machines. For 64-bit Windows machines, or as an alternative to the 1-click installation on a 32-bit Windows machine, you can download a Windows installation executable, or use SMS to run an automated installation. For Unix, an archive file is available for download to a target machine.

Once Engine instances are installed, you can manage Engine configuration from anywhere on the network with the Silver Fabric Administration Tool. You can also create configuration profiles for use by multiple machines to synchronize configurations.

For More Information

For more information on installing Engines, see “Windows Engine Installation” and “Unix Engine Installation” in the *Silver Fabric Installation Guide*.

Chapter 6 Inside the Broker

The Silver Fabric Broker provides policy-driven resource allocation and monitoring. The focal point of Silver Fabric, the Broker manages Stacks and Engines to ensure service and manage load.

The Silver Fabric administration service that manages and provides policy-driven resource allocation and monitoring. The Silver Fabric Administrator provides an HTTP accessible GUI for management and configuration of the Broker.

Topics

Broker Roles

A Broker's roles include routing and authentication, Engine allocation, Engine provisioning, and resource management. It can also serve as a virtual endpoint for J2EE Components.

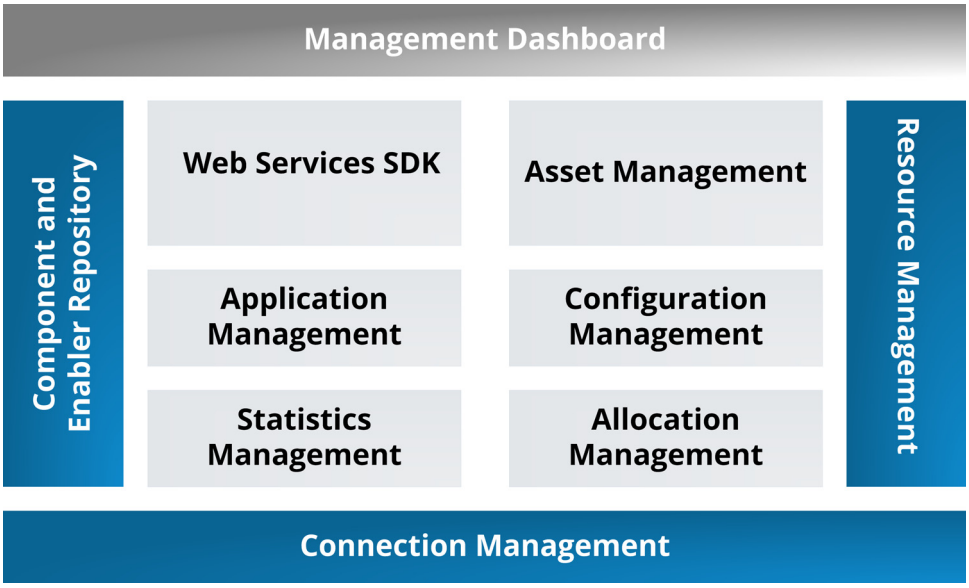

Figure 7 Roles of the Silver Fabric Broker.

Connection Management

Connection managers establish connections with Engines and other Brokers (for failover). The remote control of applications and statistics collection is made through the connection managers. The HTTP/HTTPS communication protocol provides a highly scalable, secure, firewall-friendly management plane. Heartbeats flow between components through the connections to provide robust operation with failover/failback capabilities.

Statistics Management

The Broker gathers statistics from Engines, Enablers, and Components. Statistics are tracked metrics from Engines, Components, or Archives that are reported and aggregated for historical reporting and for dynamic provisioning of incremental application resources. System statistics such as CPU usage and memory footprint

can be used with application statistics such as threads, sessions, throughput, and latency. External statistics can be implemented to provide information about conditions outside Silver Fabric's scope. You can also use scripts to provide statistics about a running Enabler or Component.

A limited amount of history is stored in memory for allocation decisions; an optional external relational database can be used for long-term history storage of system and application statistics, alerts, user events, and other tracked data. Silver Fabric supports standard RDBMS vendors. The Silver Fabric Administration Tool enables you to get simple reports or create and store custom queries.

Allocation Management

The Broker determines how many instances of each Component to run, and on which machines. The allocation algorithm, derived from Stack policies and captured statistics, provides priority-based optimization of Component allocation across machines. Policy rules govern constraints (such as OS or memory requirements), dependencies and the addition and subtraction of resources in response to runtime conditions. Blacklisting ensures that Components that repeatedly fail to start on certain machines are not continually activated on those machines.

Stack Management

The Stack manager enables you to create, modify, deploy, and activate Stacks. Silver Fabric Stacks are clusters of Components, combined with one or more policies (sets of settings and rules that dictate Engine allocation for each Component in the Stack). Where a Component corresponds to a virtualized executable, a Stack corresponds to the kind of Component bundle you'd naturally think of as a unit. For example, a typical Code-Cache-Data web-based application might be deployed in Silver Fabric as a Stack combining a Component for each tier (one for the J2EE web application "Code" front-end, a second for the in-memory cache and a third for the database back-end).

Component and Enabler Repository

The Component and Enabler repository is the central location that stores the libraries and distributions used by Engines to run Components. You can use the Silver Fabric Administration Tool to stage and deploy these archives in controlled and roles-based fashion.

Management Dashboard

The Silver Fabric Administration Tool Dashboard provides dynamic updates of quick view operational data, applications, resources, and statistics. Event notifications appear on the Dashboard and are available for capture by external tools and systems through SNMP.

Web Services SDK

Most of the tasks you perform through the Silver Fabric Administration Tool are also achievable through web services (REST/JMX), thus enabling other applications or management systems to control Silver Fabric. In addition, a command line tool and a set of ant tasks allow common activities to be easily integrated into existing operational frameworks.

Asset Management

Provisioning applications dynamically sometimes involves more than just instantiating Components on existing Engines. Depending on the structure of your data center, it could involve spinning up new machine images, powering on hardware or reconfiguring the network. In Silver Fabric's Asset Management framework, an Asset Manager is something you'd add to a Broker that would be responsible for managing external services to perform tasks like these. For instance, the VMware Asset Manager that ships with Silver Fabric enables Silver Fabric to use VMware APIs to start, stop and monitor VMware virtual machines.

The Asset Manager framework provides an API that enables an Asset Manager to react to events on the Broker and to post events to the Broker (allowing other components to react in turn). It also includes an Asset Manager management interface in the Silver Fabric Administration Tool.

Configuration Management

The Broker stores and configures many Silver Fabric settings, including users and passwords, Component information, routing properties, and Engine configuration. You configure these settings with the Silver Fabric Administration Tool.

Resource Management

Enabler and Component resources are staged on the Broker and published to Engines.

Publishing resources like Stacks, Enablers, Components, Component Archives, and Schedules moves those selected resources as staged changes to the running installation.

You can use the Silver Fabric Administration Tool to upload files to Components and publish/unpublish the Components as needed.

VirtualRouter

Silver Fabric VirtualRouter is a web application that routes requests between Components and HTTP clients such as Web browsers. The Broker continually sends Engine routing information to the VirtualRouter. In the case of J2EE Components, HTTP clients connect to VirtualRouter and are forwarded to the Component hosting the J2EE web application.

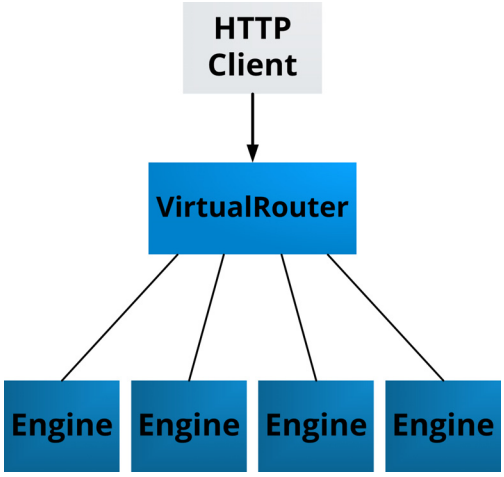

Figure 8 Silver Fabric VirtualRouter.

By default, an embedded instance of VirtualRouter runs on the Broker. The number of simultaneous requests it can handle is controlled by the number of threads available in the application server running Silver Fabric. You can run additional instances of VirtualRouter externally, on other application servers. This provides greater overall performance, as VirtualRouter can handle greater loads without affecting general Broker performance. It also provides greater system stability in the event of a Broker failure, as VirtualRouter continues to function with its last known state. Multiple VirtualRouters may be run for redundancy and load sharing. They can also be partitioned such that some VirtualRouters only forward to a certain set of Components and other VirtualRouters to other Components.

Skyway

Skyway is a web-based interface that enables a developer to obtain a desired complex development platform to build franchise applications. Skyway users are provided Templates, each of which defines a Stack that can be easily provisioned to run an application. You can configure Templates before provisioning, and also capture changes made in the Stack's Components.

Skyway runs on each Broker. It uses the Broker's definitions of Stacks, Templates, and users.

Skyway can also be used as a templating system to enable operators to easily deploy applications to a private cloud. An application running in the cloud can be captured as a template which can then be modified and later provisioned. This enables operators to repeatedly instantiate templates to environments after making minor changes.

For more information on Skyway, see the *Silver Fabric Cloud Administration Guide*.

For More Information

For more information, see “Broker Installation” in the *Silver Fabric Installation Guide*.

Chapter 7 Statistics and Monitoring

TIBCO Silver® Fabric provides robust, built-in support for monitoring and statistics collection. Monitoring is a primary tool for system health tracking, optimization and automated error-prevention. The Broker monitors the cloud as it communicates directly with each Engine. Baseline performance statistics such as free memory and CPU utilization as well as allocation statistics—number of Engines expected or allocated—are collected and tracked for each Component. Components can be configured to have additional statistics, such as throughput or request frequency, tracked as desired.

Topics

Statistics Collection

Silver Fabric monitors key statistics for IT intelligence as well as for dynamically adjusting allocation resources to applications. Engines contain statistics collection mechanisms. The Broker collects the latest average statistics at regular time intervals. The data collected includes the invoking user, Component, service and method signature. Additionally, the record contains the average values of the collected statistics.

In the figure below, the Engine collects statistics and reports them to the Broker. The Broker uses these statistics as a metric to determine whether to allocate or deallocate Engines. This creates a continuous feedback loop.

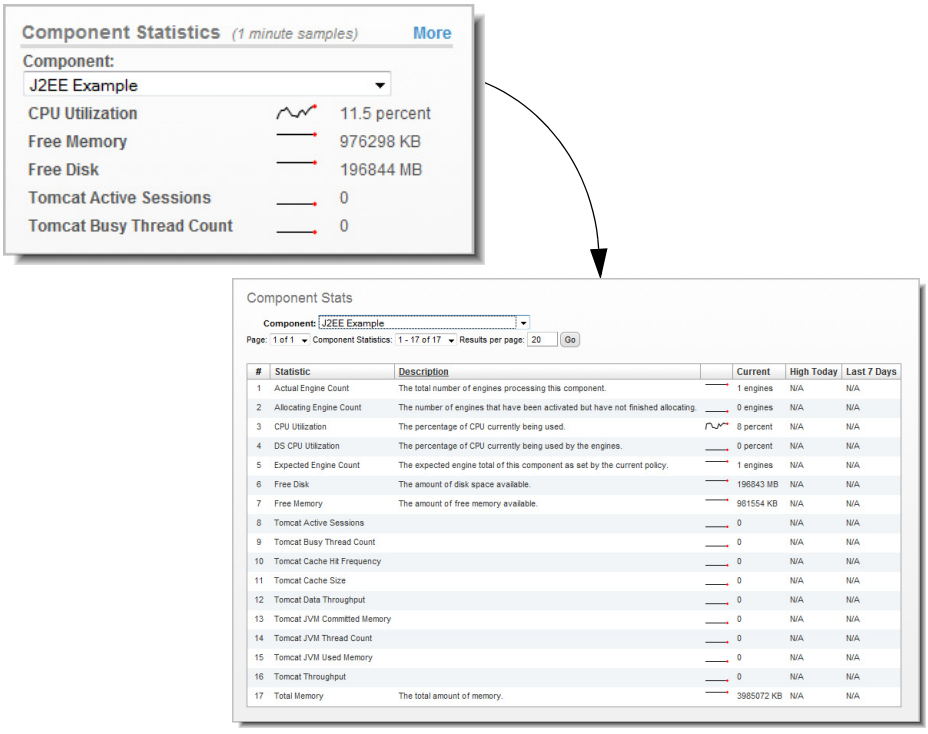

Figure 9 Statistics gathering.

Silver Fabric uses Java Management Extensions (JMX) to provide statistics and monitor enabled applications. Measuring the right JMX statistics to determine load and system performance is critical for enforcing and meeting SLAs. For example, the load on a Java application server and the commensurate user response time is not best assessed through CPU utilization. JMX attributes such as queue depth, service throughput, or thread count are better alternatives.

Silver Fabric collects Engine statistics on a regular basis. Since the collection occurs frequently for every Engine in the cloud, the resulting amount of data requires consolidation through an averaging strategy. For every Engine process, data accumulates in the Broker's in-memory statistics queue. Each interval, the Broker averages the data for each Engine, calculates the statistics for the entire cloud, and saves the resulting records into the database.

Viewing Statistics

You can view statistics in a variety of ways to monitor the effectiveness and general health of your Silver Fabric system.

The Dashboard

The Silver Fabric dashboard is the initial page of the Silver Fabric Administration Tool. The dashboard displays updated information on the system, including Engine allocation, alerts, cloud information, and more.

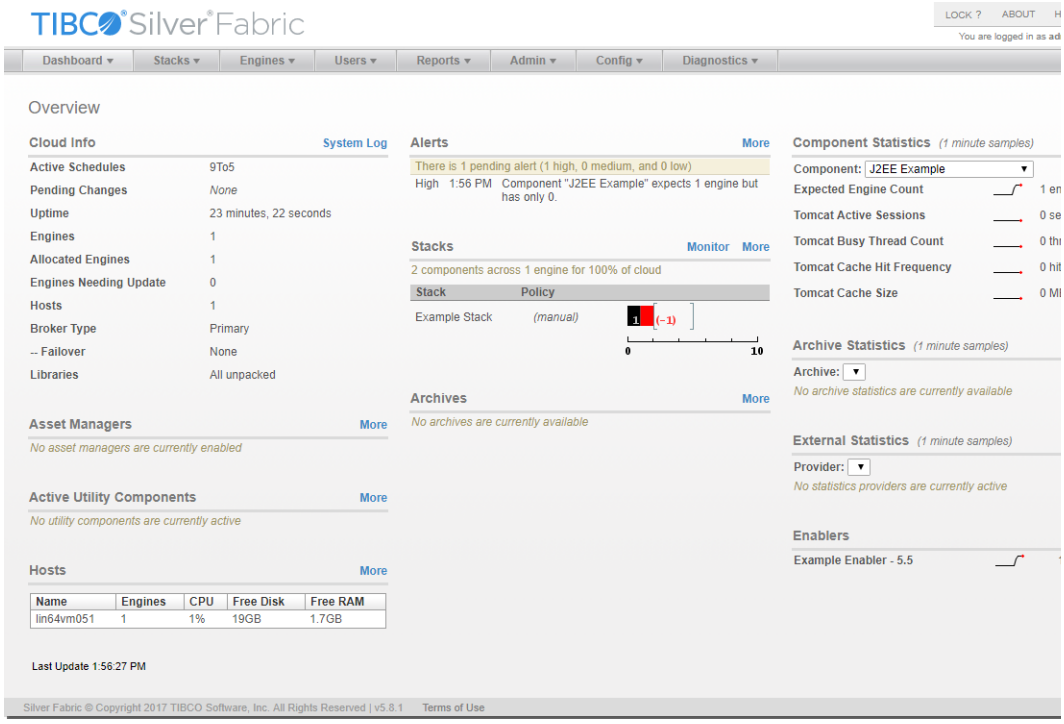

Figure 10 The Silver Fabric Dashboard.

The Reports Tab

The Silver Fabric Administration Tool provides several methods to view collected data on the Reports tab. These include an Alerts page and an Allocation Stats page, both of which are summarized on the dashboard. The Reports tab also contains pages for Component and Engine Charts. The charts pages enable you to graph a variety of statistics.

If you make available an enterprise-grade JDBC-enabled database, Silver Fabric can use it to store data including allocation events and Component, Enabler, and Engine statistics.

Glossary

A

Accounts

A collection of users, with access limited to a subset of the cloud and its objects. Accounts are typically used to "silo" users in different business units to their own set of objects in an enterprise-wide cloud. A special account, simply named "cloud", exists by default and contains admin-level users that have access to all objects.

Administration Tool

A web application used to monitor and manage a Silver Fabric installation. The interface can be used to administer the Broker, its Engines and Clients, and the associated Stacks, Components, Enablers, and Schedules.

Allocation

The number of CPUs or Engine instances given to running published Components. Allocation constraints may be set to limit the number of Engines or CPUs dedicated to running a particular Component.

Applications

Applications are now called Stacks. See Stacks.

Application Components

Application Components are now called Components. See Components.

Archives

Compressed libraries and distributions that may be published by the Silver Fabric Broker to Silver Fabric Engines with specific configurations. Archives are stored in a central repository for reuse when a Component is published to a new Engine. See also Grid Library.

Archive Scaling

An elasticity framework for scaling archives or services within an Enabler. It enables you to deploy, start, stop, and undeploy application archives to Components without restarting the Components and disrupting the running applications. Archive Scaling is also known as micro-scaling.

Assets

Applications and compute nodes are assets that can be segmented vertically by groups of users called accounts.

B

Broker

The Silver Fabric administration service that manages and provides policy-driven resource allocation and monitoring. The Silver Fabric Administrator provides an HTTP accessible GUI for management and configuration of the Broker.

C

Cloud

A collection of TIBCO Silver Fabric Engines, Brokers, and Components. Your Silver Fabric Broker, Engines, and Components running on your network is your private cloud hosted on-premise.

Component

An executable, virtualized by an Enabler, that can be published as part of a Stack . Many TIBCO products, utility services, and J2EE web applications have been productized as distinct components for publishing as part of the TIBCO Silver Fabric Cloud.

Component Type

Components with a similar set of characteristics. Components belonging to the same Component Type will have the same Component configuration wizard.

Containers

Containers are now called Enablers. See Enablers.

D

Deployment

Deployment is now called publishing. See Publish

Distribution

The Distribution contains the application server or program used for the Enabler.

E

Enabler

A wrapper around an external application or application platform, such as a J2EE application server.

Engine

The process that provisions and runs a Component instance.

Engine Configuration

A centralized profile that controls Engine Daemon behavior. Engine instances use the configuration of their Engine Daemon. Changes made to an Engine Configuration on a Broker will propagate to all Engine Daemons using that configuration.

Engine Daemon

Engine Daemon - the agent that starts, manages, and stops Engine instances on the Engine host.

G

Grid Library

Compressed files and distributions that may be published by the Silver Fabric Broker to Silver Fabric Engines with specific configurations and policies defined by an Enabler and a Stack. Grid libraries, also known as gridlibs or archives, are stored in a central repository for reuse when a Component is published to a new Engine.

P

Policy

Rules can define component allocation settings within a stack. Policies can also set a dependency, enablement condition, resource preference, threshold activation, engine group minimum or maximum.

Publishing

To move staged changes made to Enablers, Components, Component Archives, Stacks, and Schedules to the running installation.

R

Role

A set of permissions assigned to a user, each of which approves them to use, see, or access a Silver Fabric resource. Roles can be used to segment permissions horizontally, and limit actions available to a given class of user. There are five default roles, or you can also create custom roles.

Runtime

Refers to published components and stacks running on Engines.

Runtime Context Variables

Enabler settings and values that help to virtualize a component. Enablers define a set of string and environment variables, which are inherited by each Component at runtime.

S

Schedule

Defined periods of time based on hourly, daily, weekly, or monthly periods.

Skyway

A web-based application used by cloud consumers to run platforms for franchise applications. Skyway users are provided Templates, which are Stacks that can be easily provisioned to run an application.
Skyway runs on each Broker. It uses the Broker's definitions of Stacks, Templates, and users.

Stack

A bundle of Components that is activated as a unit, plus one or more policies specifying the resource needs, priorities and constraints of those Components.

Statistic

Tracked metrics of an Engine, Component, or Archive that are reported and aggregated for historical reporting and dynamic provisioning of incremental application resources.

T

Template

A Silver Fabric Stack which can be configured and provisioned from Skyway.
Any Stack can be designated as a template in the Silver Fabric Administration Tool.

V

VirtualRouter

A web application that routes requests between Components and HTTP clients such as Web browsers.

Index

R

S

T

U

V

W