



# **TIBCO ActiveMatrix® Service Grid**

## **Concepts**

Version 3.4.3 | February 2025



# Contents

---

<b>Contents</b>	<b>2</b>
<b>Introduction</b>	<b>5</b>
Challenges of Service Delivery	6
<b>Design</b>	<b>8</b>
Design Activities	8
<b>Runtime</b>	<b>10</b>
Runtime Overview	10
Hosts	11
Nodes	12
Features	14
Resource Templates	15
Resource Instances	15
<b>Administration</b>	<b>17</b>
Overview of ActiveMatrix Administrator	17
TIBCO ActiveMatrix Administrator	18
Enterprise	20
Environments	21
Messaging Bus	23
Governance	23
Identity Management	23
Policy Management	24
<b>Service Components</b>	<b>25</b>
Composites	25
Components	26

Component Implementation .....	27
Services and References .....	27
Interfaces .....	28
Bindings .....	28
Properties .....	30
Message Exchange Patterns .....	32
Java Components .....	33
Composite Components .....	33
Mediation Components .....	34
WebApp Components .....	37
Spring Components .....	37
REST Bindings .....	38
<b>TIBCO ActiveMatrix Policy Director Governance .....</b>	<b>40</b>
Advantages over Hard-Coded Policies .....	41
Features .....	41
High-Level Architecture .....	43
Policy Director Governance Administration .....	44
Governance Agent .....	45
Policy Distribution Engine .....	45
Policy Enforcement .....	47
Policy Enforcement Point Identifiers .....	49
Order of Action .....	49
Governance Agent's Merge Behavior .....	50
Governance Definitions and Concepts .....	52
Governance Controls .....	52
Governance Control Templates .....	52
Governed Object .....	53
Object Groups .....	54
Shared Resources .....	55
Policy Enforcement Host .....	56

Policy Enforcement Point .....	56
Policy Decision Point .....	56
Policy Action .....	56
Policy Runtime Store .....	57
Policy Error .....	57
Policy Conflict .....	57
<b>Glossary .....</b>	<b>58</b>
<b>TIBCO Documentation and Support Services .....</b>	<b>70</b>
<b>Legal and Third-Party Notices .....</b>	<b>72</b>

# Introduction

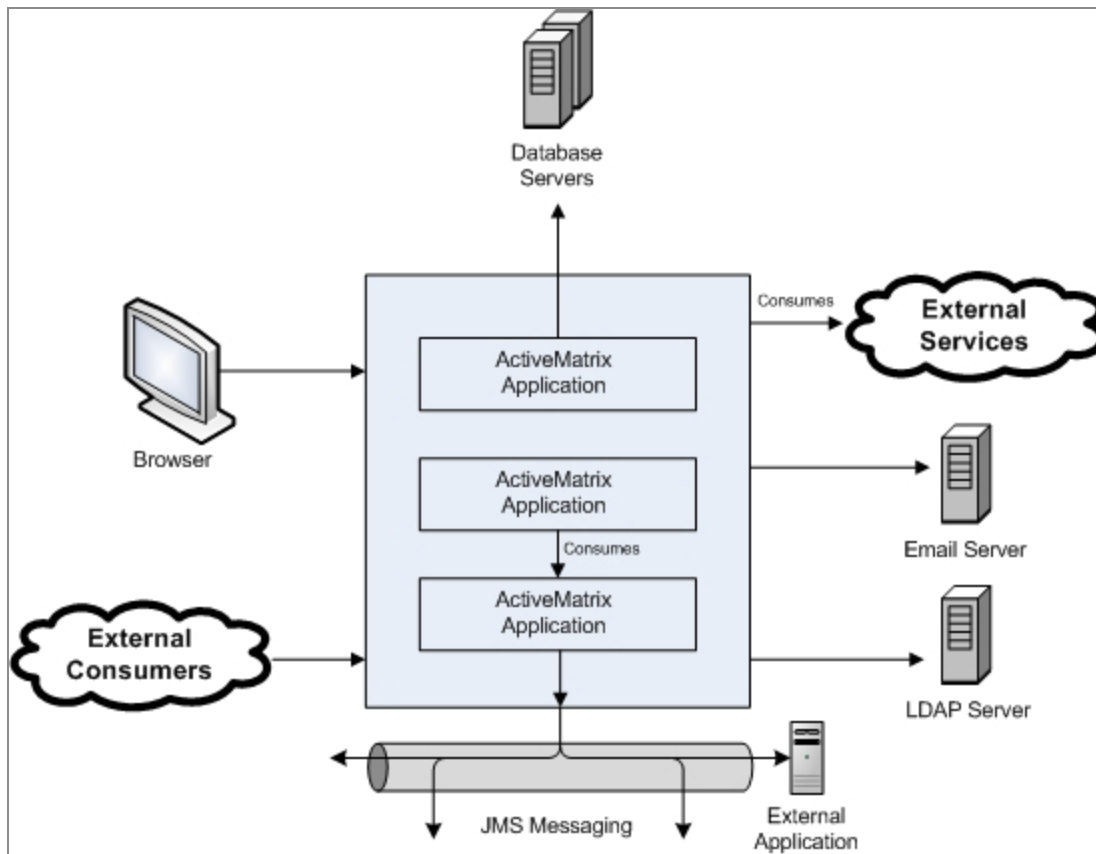
---

TIBCO ActiveMatrix provides tools for developing and packaging distributed applications, a distributed service execution environment, and tools for managing the runtime environment, applications, and the services the applications provide.

Managing a large number of distributed business applications—starting from deployment, integration, scaling, flexibility for future changes, and monitoring—pose a challenge to information technology departments. The TIBCO ActiveMatrix products help solve many of these challenges.

TIBCO ActiveMatrix Service Grid and TIBCO ActiveMatrix BusinessWorks form the core of the TIBCO Service Oriented Architecture (SOA) design. Expanding the capabilities of these products are the adapters that support interactions with non-TIBCO components. TIBCO ActiveMatrix goes a step further to consolidate runtime platforms and administration. Moreover, TIBCO ActiveMatrix uses the standardized SOA modeling based on Service Component Architecture (SCA) and SOA deployment based on OSGi specifications.

TIBCO ActiveMatrix software architecture is shown in the following figure:



TIBCO ActiveMatrix addresses and provides solutions to the following scenarios:

- Capability to build applications once and reuse
- Provides a framework for the application lifecycle
- Has the ability to host multi-tenancy
- Supports multi-domain

In addition, various types of authorization, authentication, and encryption policies can be dynamically configured to control cloud deployments. TIBCO ActiveMatrix includes complex event processing technology to dynamically scale and shrink application resources based on service-level agreements.

## Challenges of Service Delivery

The biggest challenge to service delivery is coping with the fast evolving nature of business requirements.

As business requirements, models, and priorities changes, businesses need to:

- Add new services, modify existing services, and replace or retire out-dated ones.
- Provide legacy services to new consumers and new services to legacy consumers.
- Expand the range of available services.

To scale services and meet the business needs of providing services, Information Technology departments have to tackle a series of challenges such as:

- Accommodating heterogeneous software assets
- Achieving service-consumer compatibility
- Reusing services and common features
- Managing different versions of services
- Minimizing the disruptions of transitions
- Easing of development and maintenance

Services developed and run using the TIBCO ActiveMatrix software address all of these challenges.

# Design

---

TIBCO ActiveMatrix Business Studio provides a common modeling, implementation, and deployment environment for different types of applications. TIBCO Business Studio provides an Eclipse-based design environment which:

1. Business analysts can use to capture, design, and model all aspects of a business process, including the organization and data models that underpin it.
2. Solution designers can implement the process as an executable application, then deploy the application to the TIBCO ActiveMatrix runtime for execution.

Applications developed using the TIBCO ActiveMatrix design environment conform to the Service Component Architecture (SCA), which is a model for developing applications based on the service-oriented architecture (SOA).

For more information, see [Service Components](#) section.

## Design Activities

TIBCO ActiveMatrix design activities are performed in TIBCO Business Studio, an extension of the Eclipse SDK Workbench.

In TIBCO Business Studio, analysts, architects, and developers, design, implement, configure, test, and package TIBCO ActiveMatrix applications. Beyond the standard Eclipse Workbench features, TIBCO Business Studio provides:

- Wizards and editors for creating projects and applications.
- Resource template editors for resources such as LDAP and JDBC connections, security providers, and so on.
- Editors for specifying intents and creating policies for governed objects.
- A distribution editor for specifying constraints on how application components are distributed across nodes.
- Rapid application deployment features that support deploying and testing applications on a local deployment environment.



- Tool for generating scripts to deploy applications to a remote deployment environment.
- Support for debugging applications running in a remote deployment environment.

For more information on TIBCO Business Studio, see *TIBCO ActiveMatrix® Service Grid Composite Development*.

# Runtime

---

The basic runtime elements of the TIBCO ActiveMatrix product suite are the Node, Host, and the EMS server.

The Node is the container where components, bindings, and resources are deployed. The Host is installed on every machine and represents ActiveMatrix on that machine.

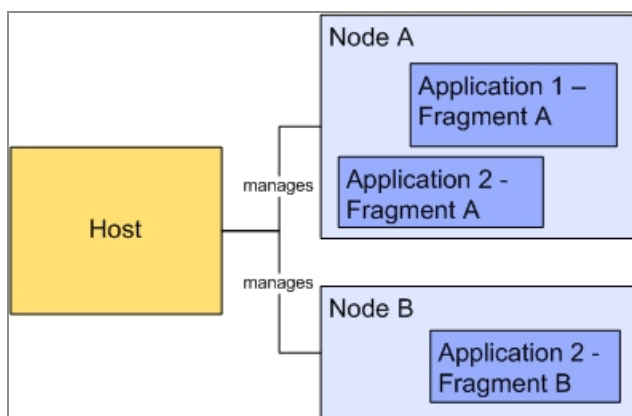
The TIBCO ActiveMatrix Administrator provides the deployment and run-time management for ActiveMatrix Environments. In reality, the ActiveMatrix Administrator too is an ActiveMatrix component running on a dedicated node.

## Runtime Overview

TIBCO ActiveMatrix employs a three-level runtime environment consisting of Hosts, Nodes, and Application fragments.

The following figure illustrates a possible configuration of Hosts, Nodes, and Application fragments. The Host manages Node A and Node B. Application 1- Fragment A runs on Node A. Application 2's fragments are distributed over Nodes A and B.

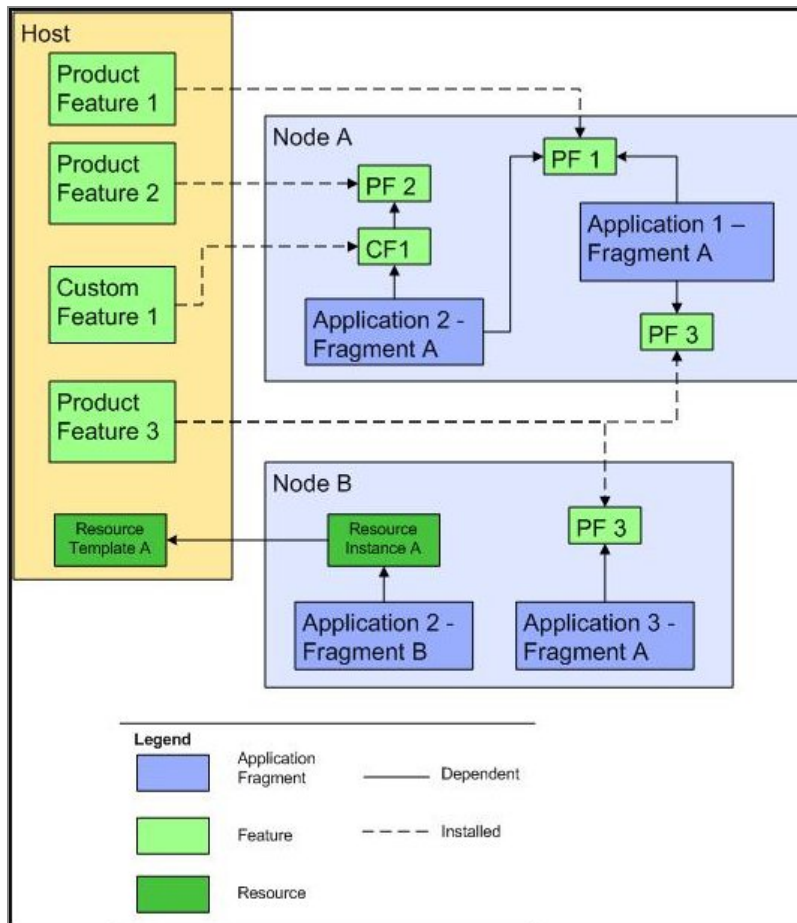
TIBCO ActiveMatrix Runtime



The following figure illustrates a possible configuration of Hosts, Nodes, features, Resource Instances, and Application fragments. In the figure, the Host has three product features: PF1, PF2, and PF3, one custom feature CF1, and one Resource Template A. Custom feature

CF1 is dependent on product feature PF2. On Node A, all the available features are installed. Node B has one installed product feature and one installed Resource Instance. Application 1- Fragment A on Node A is dependent on two product features: PF1 and PF3. Application 2- Fragment A is dependent on custom feature CF1 and product feature PF1. Application 2- Fragment B is dependent on Resource Instance A. Application3 - Fragment A is dependent on the product feature PF3.

### Runtime Configuration



## Hosts

Within the Enterprise, Hosts are used to manage Nodes. A Host can contain Nodes from more than one Environment. Hosts are responsible for deploying Applications and for other administrative tasks. Each Host has a software repository that contains the application templates, features, and resource adapters available to the Nodes managed by that Host.

Hosts have types, but the only type currently supported is TIBCO Host.

A Host is bound to a single ActiveMatrix Administrator server at a time. Hosts can contain Nodes from multiple Environments within one ActiveMatrix Administrator server.

ActiveMatrix enables isolation of Hosts so that Enterprises can separate different groups and services. The features are given below:

- A Host can be part of any number of [Environments](#).
- If the Host association is changed from **All Environments** to **Specific Environments**, support for Nodes already running on the Host continues.
- Node creation on the isolated Host is limited to users in the associated Environment or group of Environments. Host not associated with an Environment is hidden.
- It is not possible to remove a Host from an Environment if Nodes for that Environment are running on the Host.

### **SystemHost**

The Host that manages the SystemNode Node on which the ActiveMatrix Administrator server runs is named SystemHost.

### **RemoteHost**

In an enterprise, a host that is not a SystemHost is a RemoteHost.

## **Nodes**

A *Node* is the Runtime Environment for applications. Nodes exist in an Environment and are managed by hosts. When managed by a host, a Node runs with its own OS process and JVM. You can configure a host with multiple nodes. Nodes act as sandboxes for applications.

The reasons to use multiple nodes include:

- Increase throughput.
- Run different versions of software and limit the set of affected application fragments when updating software versions.
- Allow applications to use different Resource Instance configurations of the same name.

- Enable fault tolerance.
- Implement various security policies by limiting access to certain Nodes and resources.

The reasons to share a node include:

- Share Resource Instances between applications such as thread pools and database connection pools.
- Communication between components in a node avoids serialization overheads.
- Reduced overall memory utilization.

Application fragments are components or bindings of an application that are distributed and deployed to nodes. A fragment can be distributed to many nodes, and a single Node can run many fragments. To increase throughput for a component or binding, you can deploy multiple copies of the fragment to multiple Nodes.

A Node has a set of product and features shared by Resource Instances and Application fragments running on the Node. You can upgrade or downgrade the features to match the feature versions to those available in the software repository.

### **SystemNode**

The Node on which the Administrator server runs is named SystemNode. A SystemNode is a special node on which you deploy the Administration application to manage the enterprise.

### **RemoteNode**

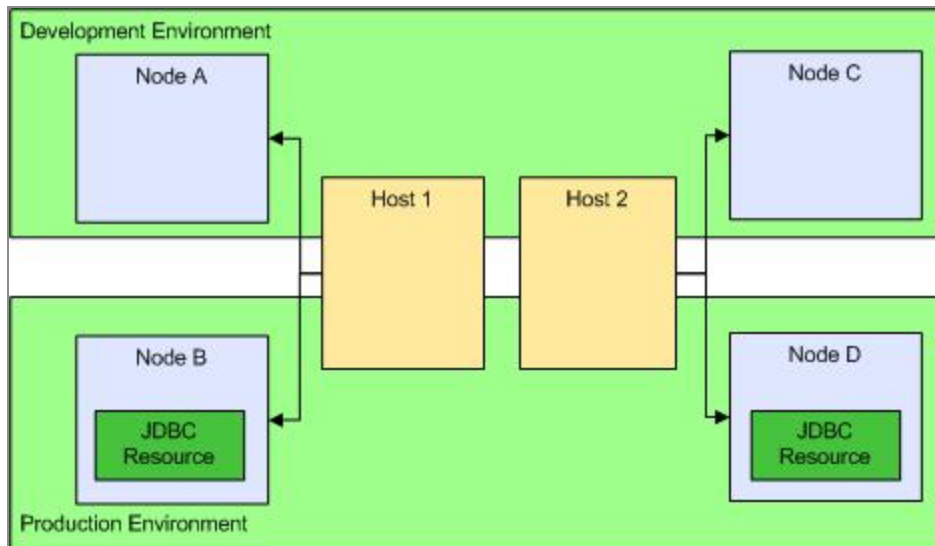
In an enterprise, a node that is managed by a RemoteHost is a RemoteNode.

To ensure stability of the ActiveMatrix Administrator, it is critical to avoid deploying any user Application to SystemNode. The same rule applies to replicated ActiveMatrix Administrator like SystemNodeReplica in a high-availability Enterprise.

The following figure depicts a configuration of Environments, Hosts, and Nodes that shows the flexibility achieved with a multi-node setup. The two Environments are assigned to groups of users that have responsibility for a specific phase of the application life cycle: Development and production. Isolation between the groups is achieved by creating two nodes on each host and assigning them to different Environments. Nodes A and B are located on Host 1 and Nodes C and D are located on Host 2. Nodes A and C are managed by the Development Environment and Nodes B and D are managed by the Production Environment. In addition, access to a JDBC resource is restricted to the nodes in the

Production Environment.

Multiple Node Scenario



## Features

A *feature* is a software package that contains plug-ins, which in turn contain component implementations and libraries. A feature is identified by an ID, a multi-part version, and its dependencies on other features. There are two types of features: system and shared library.

System features are part of a TIBCO ActiveMatrix product or contain the drivers that are installed using TIBCO Configuration Tool (TCT). Shared library features contain component implementations and libraries. When you create a Distributed Application Archive (DAA) containing a composite, you can package the composite's required features in the Application Archive or you can package the features as a standalone DAA.

When you upload a DAA containing a composite in ActiveMatrix Administrator, you can optionally import the features contained in the archive into the ActiveMatrix Administrator repository. When you deploy an application, ActiveMatrix Administrator automatically distributes the features (and any features that it depends on) to the host that manages the Nodes on which the application is distributed and installs the features on those Nodes. You can also manually install features on the other nodes managed by that host.

# Resource Templates

A *Resource Template* specifies configuration details for resources.

One Resource Template can be used to configure many Resource Instances. Resource Instances allow sharing of resources such as connection pools. They also eliminate the need to provide such details in services, component implementations, and references. Instead, you specify a property of the type of required resource in the Service, Component, or Reference. While configuring an application for deployment, the property of a Resource Instance in the Node is mapped to the application.

Resource Templates are defined at the Enterprise, Environment, or Application level. A Resource Template defined at Enterprise level is available to all environments and applications in the Enterprise. A Resource Template defined at an Environment level is available only to applications in that Environment. A Resource Template defined at an application level is available only to that application.



**Note:** TIBCO ActiveMatrix Policy Director supports only global scope that is applicable at the Enterprise level.

You can create Resource Templates in two ways:

- Manually using the command-line and Web interfaces.
- Automatically when you import the Resource Templates while creating an application.

In either case, you must have Enterprise permission to create a Resource Template.

## Resource Instances

A *Resource Instance* is a runtime object that represents a resource, such as an HTTP, JDBC, or LDAP connection.

A Resource Instance instantiates the configuration defined in a Resource Template and makes it available to Services running on a Node.

Applications, Components, Bindings, Logging Appenders, and Resource Templates can have properties whose value is the name of a resource. For example, an HTTP client Resource Template's SSL property configuration includes a property whose value is the name of SSL Client Provider resource. [TIBCO Business Studio Resource Instance](#) and [ActiveMatrix](#)

[Administrator Resource Instance](#) show how to set a resource property in TIBCO Business Studio and ActiveMatrix Administrator respectively.

### TIBCO Business Studio Resource Instance



The screenshot shows the 'SSL' configuration section in TIBCO Business Studio. It features a checkbox labeled 'Enable SSL' which is checked. Below it, the 'SSL Client Provider' is set to 'SSLClientProviderResourceTemplate'. To the right of the text box are two small buttons: an ellipsis (...) and a red 'X'.

### ActiveMatrix Administrator Resource Instance



The screenshot shows the 'Enable SSL (optional)' section in ActiveMatrix Administrator. It includes a checkbox labeled 'Yes' which is checked. Below this, the 'SSL Client Provider' is set to 'SSLClientProviderResourceTemplate'. To the right of the text box is a blue 'new' link and a small pencil icon.



# Administration

---

TIBCO ActiveMatrix Administrator is a central administration server that lets you create, deploy, and manage applications in TIBCO ActiveMatrix Administrator using a browser-based interface. There is also a set of command-line utilities available for creating EAR files and deploying applications.

## Overview of ActiveMatrix Administrator

TIBCO ActiveMatrix Administrator provides a UI interface and runtime interface.

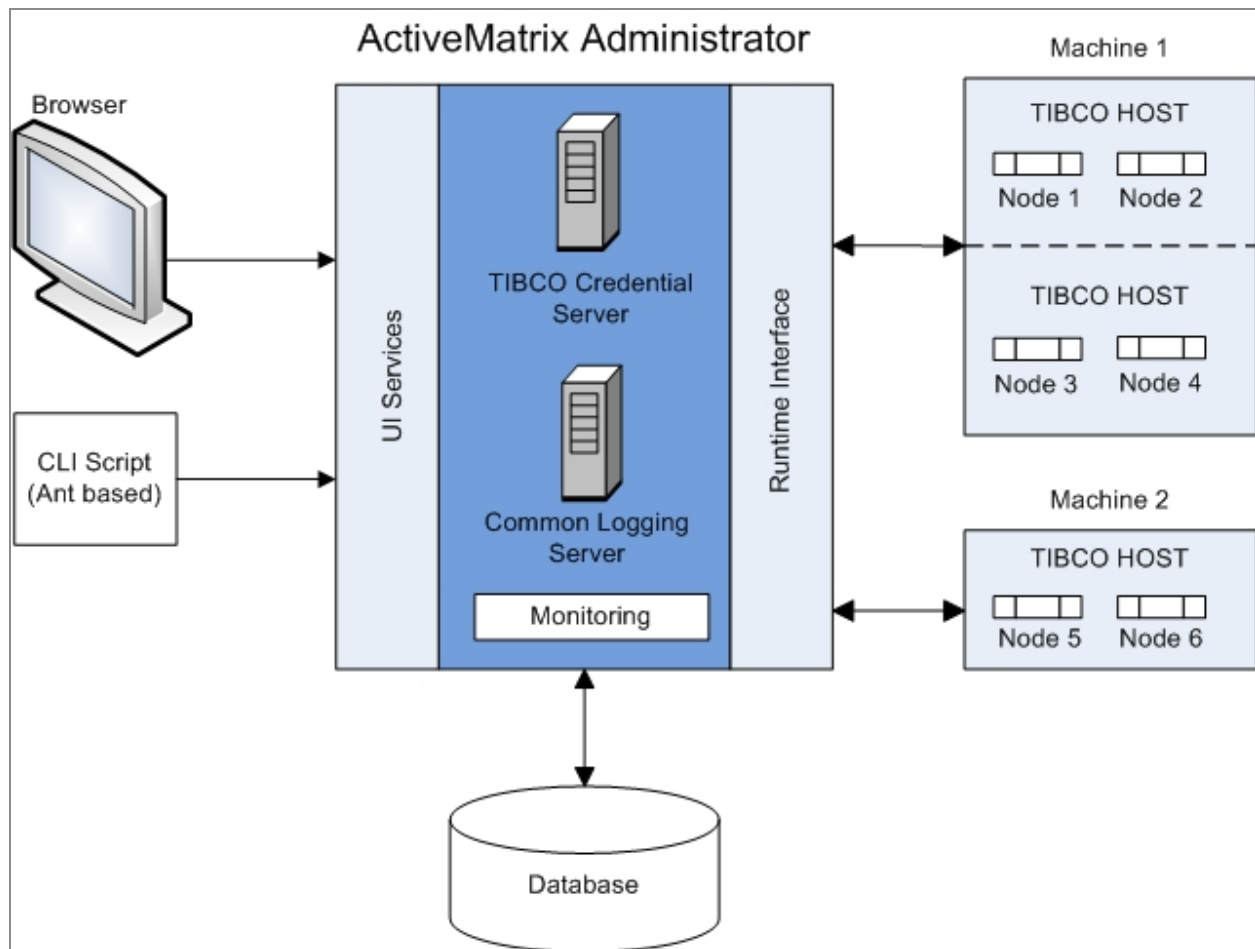
TIBCO ActiveMatrix administration is supported by TIBCO ActiveMatrix Administrator and TIBCO Business Studio. System administrators can do the following:

- Configure Environments and Messaging Buses
- Register hosts and associate them with environments
- Provision Nodes with features and resources
- Provision Hosts
- Deploy, configure, and manage applications

In TIBCO Business Studio developers, deploy and debug applications.

The following figure illustrates the relationship between TIBCO ActiveMatrix Administrator and the objects that it manages. This section provides an overview of the Environment and Messaging Bus. [Runtime](#) discusses hosts, nodes, applications, features, and resources.

## TIBCO ActiveMatrix Administration



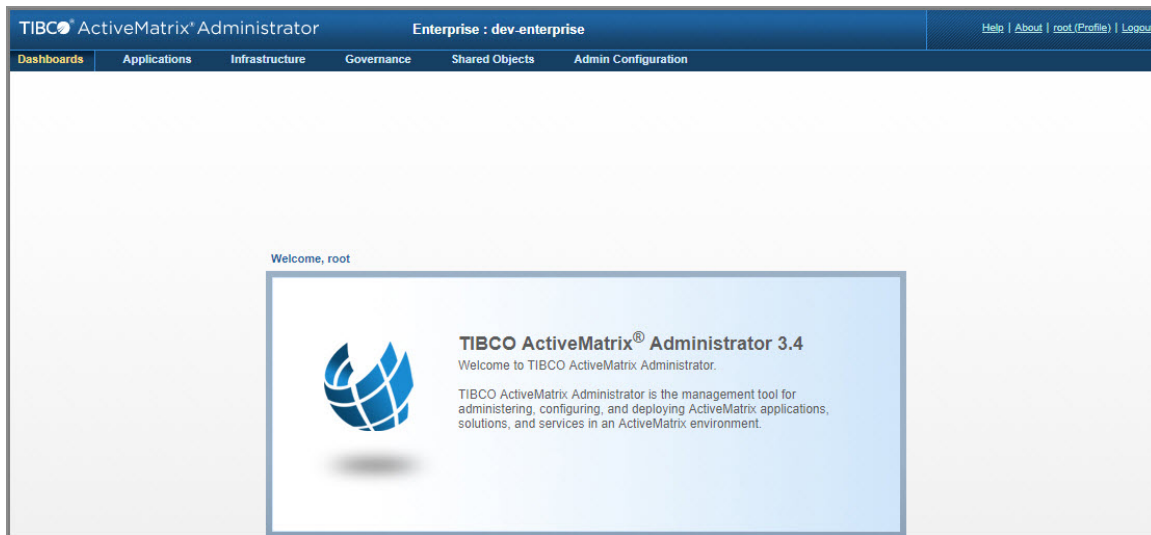
## TIBCO ActiveMatrix Administrator

TIBCO ActiveMatrix Administrator consists of the following components:

- Administrator server
- Administrator server clients
  - Administrator web interface
  - Administrator command-line interface
  - TIBCO Business Studio

In the following figure, the ActiveMatrix Administrator web interface is opened to the welcome screen.

## TIBCO ActiveMatrix Administrator



The communication channel between the ActiveMatrix Administrator server and its clients can be secured with SSL.

The administrator server hosts TIBCO Credential Server, which provides credentials to secure communication between the ActiveMatrix Administrator server, Hosts, and Nodes using SSL.

The Node on which the ActiveMatrix Administrator server runs product applications that provide various platform services:

- Log - Aggregates log data from nodes and saves to a persistent store.
- Payload - Stores and retrieves large payloads for log entries.
- Monitoring -Aggregates performance data from nodes and saves to a persistent store.

## Servers

Administrator servers interact with other servers:

- Database - maintains Administrator server configuration, performance, log, and payload data
- Authentication realm - maintains user data
- Notification - propagates status messages between Administrator server, hosts, and Nodes
- Messaging Bus - propagates messages between Applications

The communication channels between Administrator servers and other servers can be secured with SSL. For information on SSL support, see *TIBCO ActiveMatrix® Service Grid Installation and Configuration*.

For information on ActiveMatrix Administrator, see *TIBCO ActiveMatrix® Service Grid Administration*.

## Enterprise

In TIBCO ActiveMatrix terms, the Enterprise is the top-level object in the hierarchy of administration. The Enterprise contains the hierarchy of Environments, Hosts, and Nodes. And also contains the Shared objects that are available across the Enterprise. The objects are available in the ActiveMatrix Administrator UI.

In the runtime, Enterprise refers to the collection of runtime objects that share the Enterprise Message Service server that functions as the notification server. The Enterprise is identified by a name specified when you create an ActiveMatrix Administrator server.

The following table lists the Enterprise objects and their location in the ActiveMatrix Administrator UI.

### *Enterprise Objects*

<b>Enterprise Object</b>	<b>Location</b>
Application Template	<b>Infrastructure &gt; Software Management &gt; Application Templates</b>
Environment	<b>Infrastructure &gt; Environments</b>
Feature	<b>Infrastructure &gt; Software Management &gt; Features</b>
Host	<b>Infrastructure &gt; Hosts</b>
Logging Appender	<b>Shared Objects &gt; Logging Appenders</b>
Resource Template	<b>Shared Objects &gt; Resource Templates</b>

Enterprise Object	Location
Substitution variable	<b>Shared Objects &gt; Substitution Variables</b>
User, super user, and group	<b>Governance &gt; Users and Groups</b>
Enterprise permission	<b>Governance &gt; Enterprise Permissions</b>

## Environments

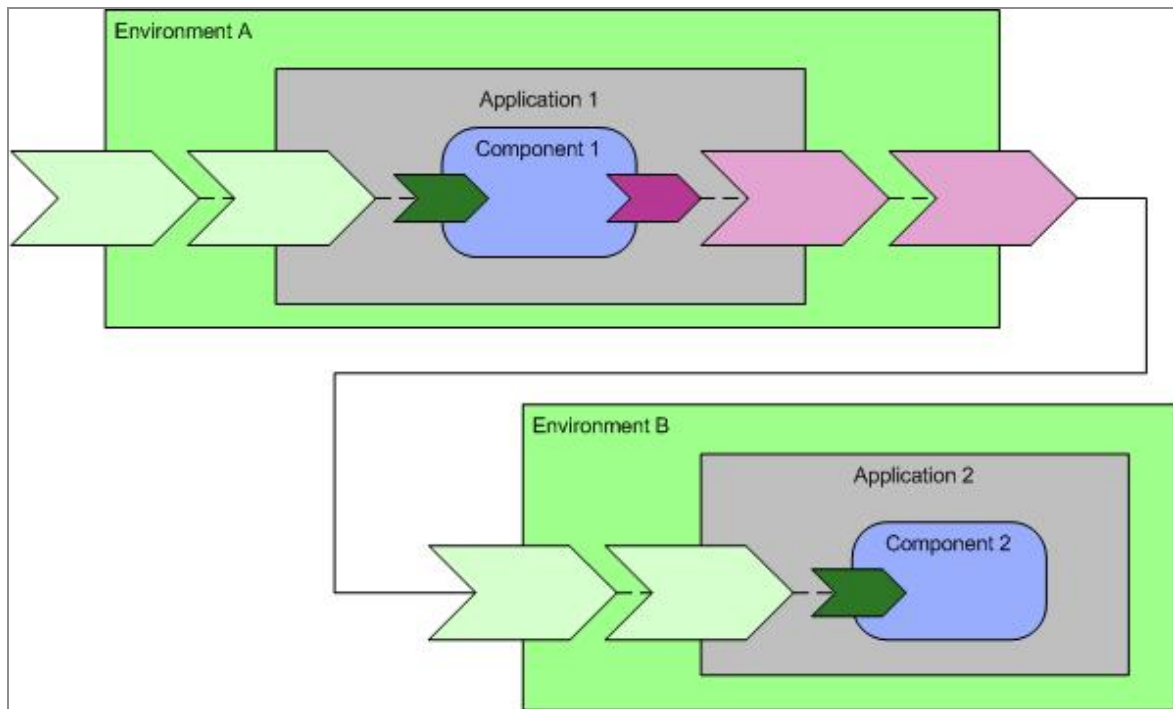
An *environment* is a logical grouping of applications and nodes. An Administrator server can have multiple environments. For example, you can define environments distinguished by product life-cycle functions such as development and production, by geographical area, or by business unit.

Environments provide a way to isolate one group of applications and nodes from another. This is useful for security, optimizing network traffic (each Environment has its own Enterprise Message Service server for service bus communication), and visual organization in the ActiveMatrix Administrator UI. [Hosts](#) can also be isolated and associated with one or more Environments.

Environments contain the following types of objects:

- **Applications:** The Services and references defined by an application can be promoted to the Application's Environment. Services and references promoted to the Environment level can be wired to each other. The following figure illustrates a Service and Reference exposed by a component, promoted to the composite level, promoted again to the Environment level, and wired between the promoted Reference and Service.

Cross Environment Wires



- **Nodes:** Runtime sandboxes that run application logic. Node names must be unique within an Environment and within a host.
- **Messaging Bus configuration**

## Administrator Environment

The Environment containing the node on which the ActiveMatrix Administrator server runs, SystemNode, is named SystemEnvironment. In a high-availability enterprise, the SystemEnvironment contains the SystemNodeReplica as well.

**i Note:** SystemEnvironment is solely for ActiveMatrix platform applications. Do not deploy any user application to SystemEnvironment.

## Development Environment

When you create an ActiveMatrix Administrator server, you have the option to create a Development Environment and Node. By default, the Environment is named DevEnvironment and the node is named DevNode.

# Messaging Bus

An Environment's *Messaging Bus* is the communications backbone that mediates message exchange between service consumers and providers. When a consumer makes a service request, it is the responsibility of the Messaging Bus to locate providers that offer the service and deliver the message to a provider.

Messaging Bus enables service virtualization. With service virtualization, a reference does not need to know about the binding details of the service with which it is communicating. It only needs to know the name of the service. Service virtualization allows applications within an environment to communicate without requiring the Applications' promoted Services and References to have bindings.

# Governance

Governance is a concept to exercise organizational control over the development, deployment, and operations of services.

In service-oriented architecture, operational governance assures service execution to ensure that services behave according to the specified mandates and guidelines. It can include service monitoring, resource optimization, fault tolerance and access control. Several features in the design, runtime, and administration components of the TIBCO ActiveMatrix platform support operational governance. At design-time, TIBCO Business Studio allows application developers to specify intents and policy sets. At administration time, TIBCO ActiveMatrix Administrator supports credentials that support identity management.

# Identity Management

Identity management is supported by Security Resource Instances that you define in TIBCO ActiveMatrix Administrator and install in runtime nodes. The Resource Instances are invoked when policies are enforced by the governance agent.

## Policy Management

Policy management involves specifying a capability or constraint on a governed object—Composite, Component, Service, or Reference—to affect runtime behavior such as security, reliability, transactions, threading, and quality of service.

Constraints and capabilities are managed separately from the core business logic of your application. To enable policy management, an application designer specifies intents and policy sets.

Policies specified at design-time are packaged into the deployment archive and enforced via a governance agent embedded in the TIBCO ActiveMatrix runtime.

An *intent* describes abstract constraints on the behavior of a component, or on interactions between Components. Intents let application designers specify requirements in a high-level, abstract form, independent of the configuration details of the runtime and bindings. Intents guide administrators as they configure bindings, policies, and runtime details.

A *policy* is a configuration that specifies the details that TIBCO ActiveMatrix needs to enforce a constraint declared in an intent. A policy can also be specified without an intent.

A *policy set* contains one or more policies. Adding a policy set to a governed object applies its Policies to the object.

TIBCO ActiveMatrix support for intents, policy sets, and policies conforms to the [SCA Policy](#) specification. TIBCO ActiveMatrix also uses TIBCO ActiveMatrix Policy Director Governance, which is a governance solution to manage and enforce cross-functional requirements such as security, monitoring, and compliance independent of their implementation and deployment. For more details, see "TIBCO ActiveMatrix Policy Director Governance" in *TIBCO ActiveMatrix BPM SOA Concepts*.



# Service Components

---

Service Component Architecture (SCA) defines a model for developing Applications based on a service-oriented architecture.

[Service Component Architecture \(SCA\)](#) is the foundation of TIBCO Silver ActiveMatrix support for service-oriented Applications. Business function is provided as a set of Components assembled into a structure called a Composite.

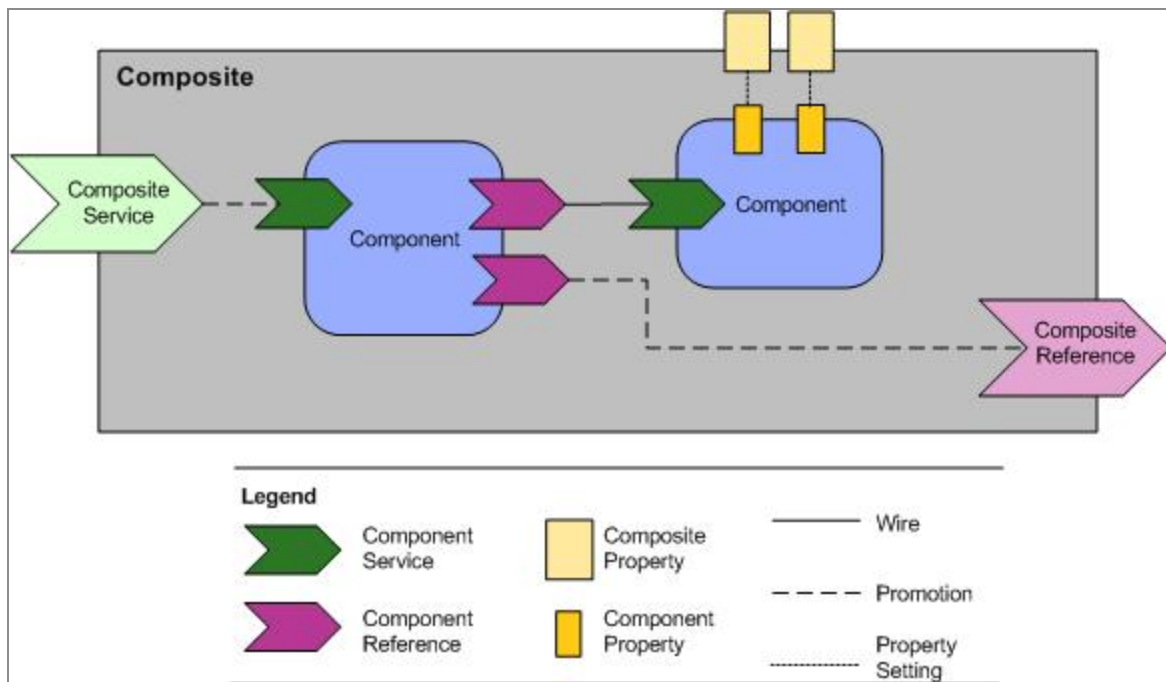
## Composites

A *composite* is a configuration of services comprising an application that conforms to a service-oriented architecture. A Composite contains Components, Services, References, the wires that interconnect them, and properties that are used to configure the components. Composites can be nested (contained by other composites). A root Composite equates to an SCA application.

For information on service-oriented architectures, see the [SCA Assembly](#) specification.

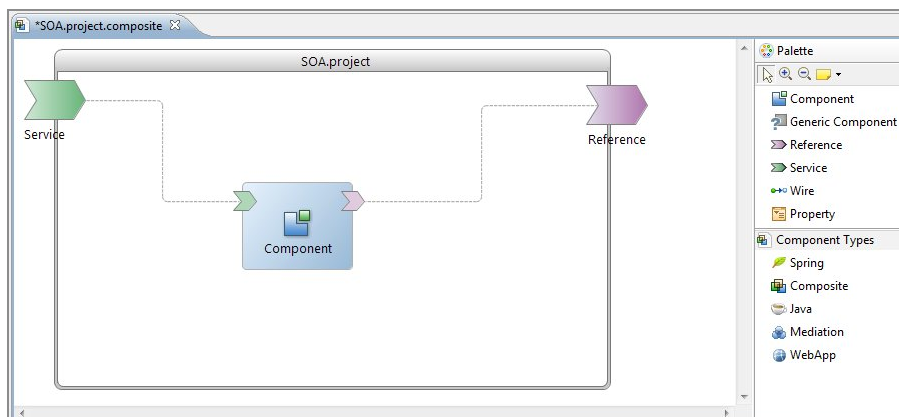
The constituents of a composite can run in a single process on a single computer or be distributed across multiple processes on multiple computers. A complete application might be constructed from just one composite or it could combine several different composites. The components making up each composite can be implemented using different technologies.

## Composite



You can edit Composites in the TIBCO Business Studio Composite Editor shown in the following screenshot.

## Composite Editor



# Components

A *Component* is the basic element of a business function. It is defined at design-time.

Components are configured instances of implementations. More than one Component can use and configure the same implementation.

Components can have Services, References, and properties. All of these can be promoted to the Composite level during design-time. Promotion enables an ActiveMatrix Administrator to wire or configure Services, References, and properties when the Application is deployed. Services, References, and properties that are not promoted are private to the Application and are set at design-time only.

Components can have several different types of dependencies. Components can express dependencies on product features, custom features, other Components, and resources. All of a Component's dependencies must be satisfied for it to be deployed to a Node.

Components can be deployed to multiple nodes for fault tolerance or load balancing.

## Component Implementation

A Component's *implementation* concretely provides the business function.

The TIBCO ActiveMatrix family of products supports different implementation types in different products. For example: Java, Mediation, WebApp, and Spring.

An *abstract component* is a component whose implementation type is unspecified but whose interfaces and connections to services, other components, and references are defined. Abstract components can be used by system architects to defer the choice of implementation type while specifying the relationship of a component to other composite elements. Abstract components cannot be packaged or deployed.

## Services and References

Applications interact via services and references. A *service* is a set of operations and the messages required by the operations. A *reference* identifies the service consumed by a component or composite. Applications offer services and invoke references to other services.

An application's services and references are promoted from the services and references of the Components that it contains.

Component services can be consumed by other components within the Composite or promoted as composite services for use by consumers outside the Composite. A composite service has an interface and one or more bindings.

Component references consume services provided by other components in the same Composite or services provided outside the Composite. A Composite Reference has an interface and one binding.

## Interfaces

An *Interface* defines the contract for Services and References. Services and References can interact only when they have the same Interface. An Interface defines one or more operations and each operation has zero or one Request (input) message and zero or one Response (output) message. The Request and Response messages may be simple types such as strings and integers or they may be complex types. In the current release, TIBCO ActiveMatrix supports WSDL 1.1 port type Interfaces.

## Bindings

A *binding* specifies how communication happens between a reference and a service. A Service binding describes the mechanism that a client uses to access a service. A Reference binding describes the access mechanism a reference uses to invoke a service. References can have at most one binding.

TIBCO ActiveMatrix supports the following Binding Types (BT):

- Virtualization
- REST
- SOAP
- JMS

Virtualization Bindings connect Services and References to the Messaging Bus. Virtualization Bindings are automatically created for every Composite Service and every wired component service and reference. At design-time, Virtualization Bindings of Component Services and References are implicit; their properties cannot be viewed.

There are two types of Virtualization Bindings: internal and external. An *internal binding* is associated with a Component Service or reference. An *external binding* is associated with a


service or reference promoted to the root composite. Administrators can create or modify wires connected to external bindings and can monitor, start, and stop external bindings.

The following bindings are explicitly created by architects and developers only on promoted services and references:

- SOAP
- JMS
- REST

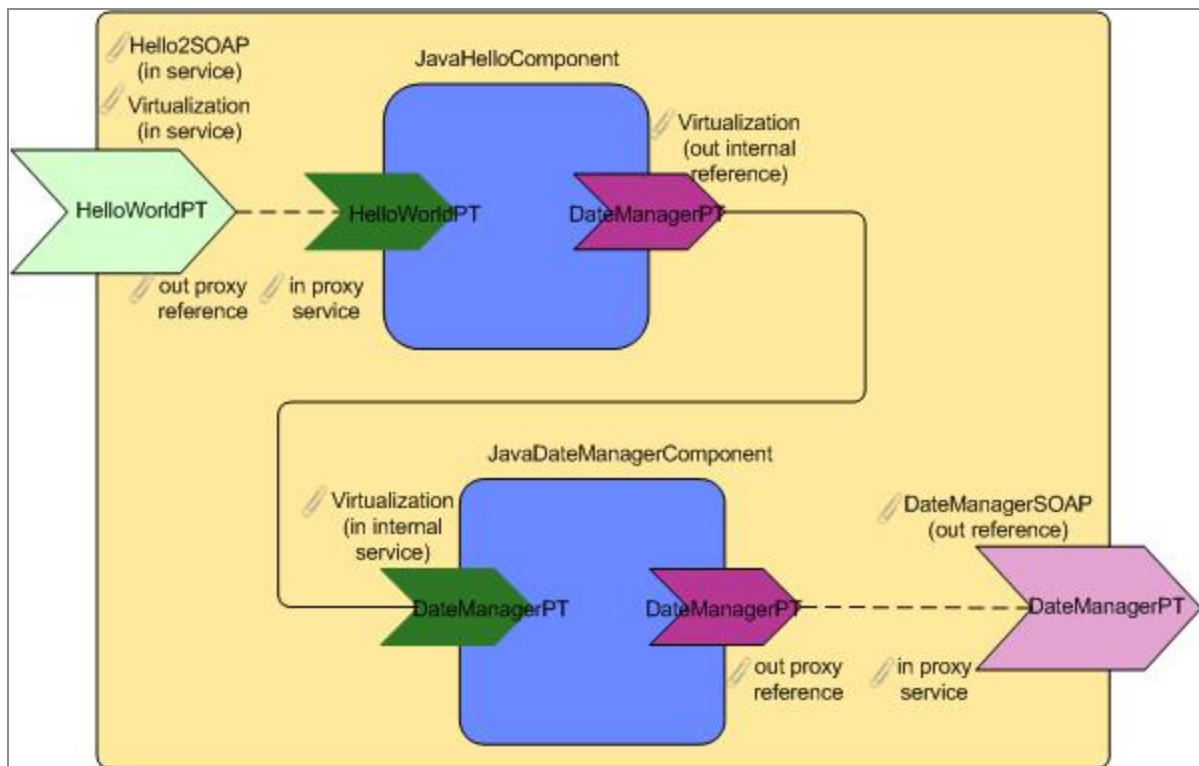
**i Note:** TIBCO Business Studio and TIBCO ActiveMatrix Administrator provide the option to choose between TIBCO's SOAP/JMS and W3C SOAP/JMS for SOAP Binding Type while adding a Binding to a Service.

**i Note:** SOAP Bindings support both HTTP and JMS transport types.

The following figure, Bindings are indicated by a  icon. The promoted Service HelloWorldPT has a SOAP and external Virtualization Binding. The components have internal Virtualization Bindings. The promoted reference DateManagerPT has a SOAP binding. In addition, anytime a Service or Reference has a binding of type other than Virtualization, a pair of proxy (Virtualization) bindings are created to connect the Service or Reference to the component to which the Service or Reference Service is wired.

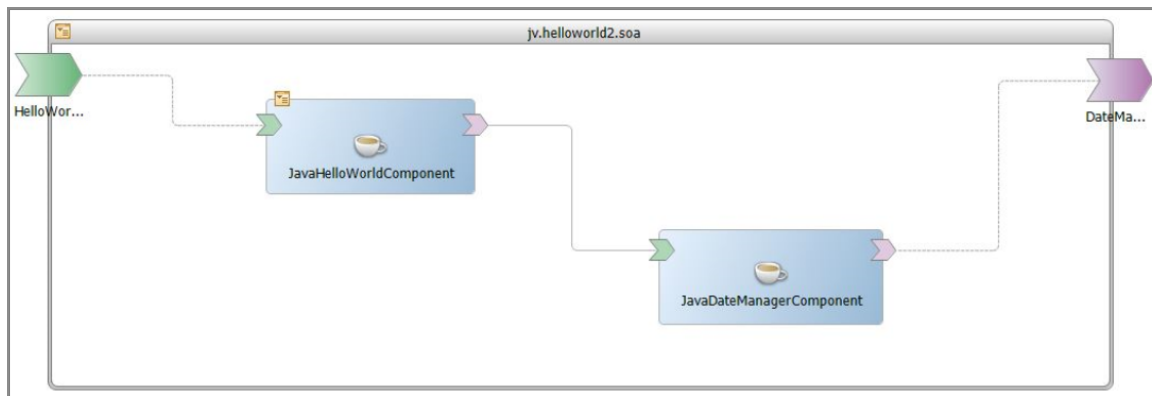
This figure is a representation of the TIBCO\_HOME/samples/java/helloworld2.zip sample in the installation. You can run the sample to get a better understanding of the Bindings.

## Bindings



Here is an example of how this scenario looks in TIBCO Business Studio:

## Bindings in Business Studio



## Properties

A *property* is an externally visible data value. Properties enable object behavior to be configured at deployment time.

A property has a type, which may be either simple or complex. Implementations, components, composites, bindings, logging configurations and appenders, and resource templates can have properties. Implementation, component, and composite properties are defined in TIBCO Business Studio. Binding, logging configuration and logging appender, and resource template properties are defined by the TIBCO ActiveMatrix platform.

Properties can have explicit values or may be bound to substitution variables, which can be set at deployment time in various scopes. Depending on the object possessing the property, the property value can be bound at design-time, deployment time, or both:

- At design-time you can provide default values and indicate whether a composite or component property value must be set at deployment time.
- Some properties can be bound to substitution variables.

At design-time, a composite property value can be set to a constant or bound to a substitution variable. Either type of binding can be overridden at administration time. However, only the properties of the root composite of an application or those on bindings associated with application-level services and references can be overridden. If there are nested composites (components of type composite), then their property values cannot be changed by an administrator.

A composite property is specific to an application. Often the same property may be defined in more than one application. For business reasons or, ease of use an administrator may want to define the value only once and have it be used by more than one composite property. This is achieved by binding the composite property to a substitution variable, which can be defined at the enterprise, host, environment, node, application, and application fragment levels.

The Owner column displays more contextual information about the owner of the property. Properties display a prefix indicating the context as follows:

- Application-level properties display with the prefix [Application].
- Binding level properties display with the prefix [Service] or [Reference].
- Component level properties display with the prefix [Component].
- Properties of nested composites display with the prefix [Composite].
- Properties for certain policy set such as Threading policy display with any of the preceding prefixes depending on where the policy set was added.

Editable Properties   <a href="#">Non-editable Properties</a>   <a href="#">Policy Set Properties</a>			
Owner	Property Name	Property Type	Property Value
[Application]	test	string	test
	MEDIATION_VALIDATE_MESSAGE_DATA	boolean	false
[Service] Sample > SOAPService_Binding1	HttpInboundConnectionConfig	HttpConnector	httpConnector
[Reference] Sample1 > SOAPReference_Binding1	HttpOutboundConnectionConfig	HTTP Client	HttpClient_SampleSOAP
[Service] Sample > SOAPService_Binding1 > ThreadingPolicy_S	threadpool	Thread Pool	TH-Test

A component may be deployed to more than one node and you may want to have different values passed for a component property in every node. In such cases you would set the component property to a substitution variable, and set the substitution variable to different values on each node.

## Message Exchange Patterns

A provider generates and responds to messages according to the operations defined in the interface it offers. The interface is always written from the perspective of the provider. That is, if an interface says that the messages are input and then output, the provider first receives a message and then sends a message. A consumer uses a service, and interprets an interface to consume a service. The consumer handles messages in the opposite direction from the provider.

A message exchange pattern (MEP) defines the sequence and cardinality of messages sent between the provider and the consumer. MEPs contain both normal and fault messages. TIBCO ActiveMatrix software supports the following MEPs:

- One-Way (In-Only): A consumer sends a message to a provider.
- Request-Response (In-Out): A consumer sends a message to a provider, with an expectation of a response. The provider sends a response message. The provider may generate a fault if it fails to process the message.

Faults are errors that can occur at any point during the processing of a message. Faults can also be thrown by the target service while processing messages. In service-oriented applications, clients expect specific fault responses to be returned when errors occur. For example, SOAP clients expect a SOAP fault message to be returned when an error occurs during processing. Each implementation type supports methods for generating faults in response to error conditions.



# Java Components

Java components integrate Java classes into the TIBCO ActiveMatrix platform.

The integration conforms to [SCA-J](#) specifications. Java components support service implementation using the flexibility and power of a general-purpose programming language.

TIBCO Business Studio facilitates Java component implementation by providing a rich set of automatic code generation and synchronization features. TIBCO Business Studio supports both WSDL-first and code-first development.

You can develop Java components and generate classes that conform to the WSDL interface specification of the component's services and references. When you add a service, reference, or property to a Java component and regenerate the implementation, TIBCO Business Studio adds fields and methods that represent the service, reference, or property to the component's implementation class.

You can also configure an existing Java class as the implementation of a Java component and update component properties to match the implementation.

For information on Java components, see *TIBCO ActiveMatrix® Service Grid Java Component Development*.

# Composite Components

A composite may serve as the component implementation for a higher-level composite. Composite components enable architects to structure complex applications as a hierarchical collection of parent and child composites.

When a composite is used as a component implementation, the components within that composite cannot be referenced directly by the using component. In other words, the internals of the composite are invisible to the using component. The using component can only connect wires to the services and references of the referenced composite and set values for properties of the composite. The services, references, and properties of the composite define a contract that relies on using the component.

# Mediation Components

Mediation is the process of resolving differences between two entities, for example, when bridging transport or interface differences.

Mediation components support enterprise service bus (ESB) features and manage interactions between service consumers and service providers. TIBCO ActiveMatrix Mediation components are implemented with Mediation Flows, which are created using the Mediation Flow Editor. The TIBCO ActiveMatrix Mediation Flow Editor provides a zero coding graphical tool that allows you to build mediation flows between service consumers and service providers that are described using WSDL files. The primary building block of a mediation flow is called a mediation task. Mediation tasks are primitives that implement ESB functions such as logging, data transformation, routing, and others.

Mediation flows:

- Map requests to one or more service providers. For example, route requests based on the message content, the message context, or both.
- Send back response messages received from service providers with or without transforming them.
- Manage faults. Mediation flows can throw faults, catch faults from service providers, and transform them to the faults expected by service consumers.
- Construct and send reply messages to service consumers without invoking the service providers.
- Provide access to metadata such as security context, SOAP headers and other message context details.
- Use custom mediation tasks, which can be developed with custom mediation task wizards. TIBCO ActiveMatrix mediation features provide wizards and a public API for developing custom mediation tasks.

Composites containing mediation components can address the following ESB scenarios:

- Service virtualization
- Transport bridging
- Message exchange pattern bridging
- Message content and context-based routing
- Static and dynamic routing

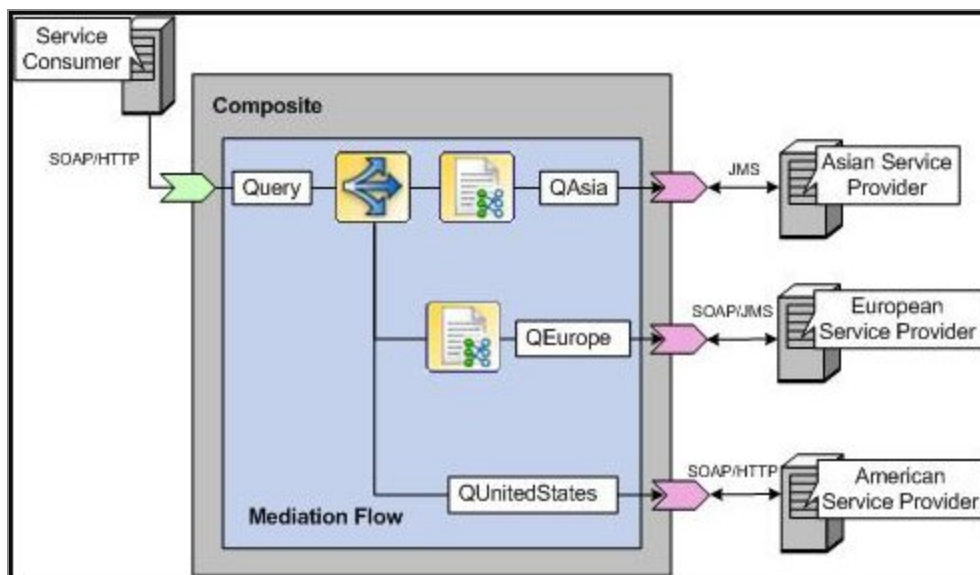
- Data transformation
- Data enrichment
- Data validation
- Message filtering
- Log message and context data

By applying one or more ESB scenarios, you can implement ESB patterns such as:

- Gateway (Route)
- VETO (Validate, Enrich, Transform, Operate)
- VETRO (Validate, Enrich, Transform, Route, Operate)

The following figure shows how a service consumer invokes a mediation flow and how the mediation flow interacts with target services.

### Mediation



In this figure:

### Procedure

1. The service consumer, accessing the mediation service over SOAP/HTTP, invokes the query operation in the mediation service to request information.
2. Based on the contents of the message, a route task directs the message to one of

three target operations, provided by web services in Asia, Europe, and the United States. Transport and interaction-protocol bridging allow communication with the target service providers to proceed.

3. For Asia and Europe, transform tasks transform the message structure and contents provided by the service consumer to ones that the target service providers can accept.

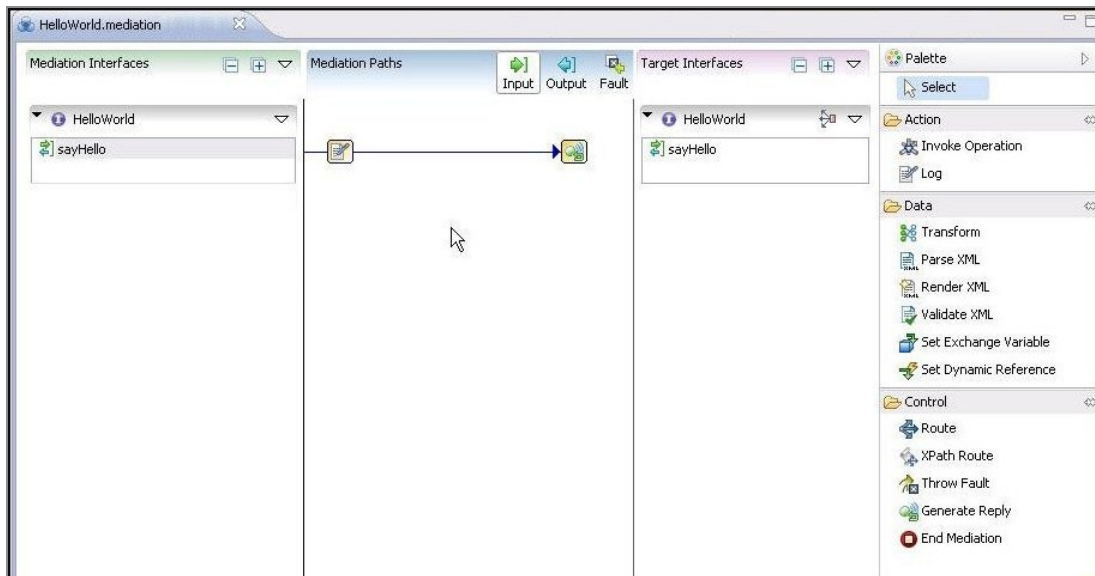
In summary, TIBCO ActiveMatrix mediation technology provides:

- **Service virtualization:** A mediation service hides the location of service providers and details of how the services are provided (for example, the transport protocol, message format, and schema) from service consumers. Virtualization enables:
  - **Location transparency:** The location of the service providers is hidden from service consumers.
  - **Transport bridging:** A composite application containing one or more mediation components can provide a bridge between service consumers and service providers that use different transport protocols.
  - **Connections between mediation operations and target operations:** Mediation flow associate each mediation operation with one or more target operations.
- **Content and context-based routing:** A routing task placed on the input path of a mediation flow can route service requests to alternative target services based on the message content, message context, mediation flow parameters or all of these. A routing task can also route service requests to Throw Fault tasks, as a means of rejecting requests.
- **Data transformation:** When routing a service request to alternative service providers, it might be necessary to transform the message structure, data types, or contents used by the service consumer to the ones expected by the service provider, and conversely. Transform tasks perform these transformations.
- **Fault management:** Mediation flows provide the ability to map fault types reported by service providers to ones understood by service consumers. A mediation flow can also throw faults based on routing cases, rather than sending every message to a service provider. Finally, mediation flows handle runtime faults that occur in the mediation flow itself.
- **Logging:** Log tasks can log elements of the message content, message context, mediation flow context or all of these elements.

- **Custom mediation tasks:** To provide a mediation feature not present in pre-defined mediation tasks, you can write code that performs a custom mediation task, and incorporate the task in the Mediation Flow Editor using wizards.

The following figure shows a Hello World mediation flow containing a log task opened in the Mediation Editor.

Mediation Flow



For information on Mediation components, see *TIBCO ActiveMatrix® Service Grid Mediation Component Development*.

## WebApp Components

A *web application* delivers services and content over the Internet.

WebApp components integrate Java EE web applications into the TIBCO ActiveMatrix platform. The integration conforms to [SCA-J](#) specifications.

For information on WebApp components, see *TIBCO ActiveMatrix® Service Grid WebApp Component Development*.

## Spring Components

Spring components integrate Spring Beans into the TIBCO ActiveMatrix platform.

A Spring component is very similar to a Java component, but its implementation can consist of more than one Java class. The classes are specified in a Spring Bean configuration file. Each Spring Bean corresponds to a Java class. In Spring components, each service, reference, and property are associated with a Bean (as opposed to all being associated with the same Java class in the case of Java components).

For information on Spring components, see *TIBCO ActiveMatrix® Service Grid Spring Component Development*.

## REST Bindings

REST components map your SCA services to REST. This allows you integrate your SCA services with clients that use HTTP instead of SOAP.

ActiveMatrix service development typically starts with a WSDL, which defines the service interfaces. Developers expose SOAP or JMS services by adding SOAP or JMS bindings on a promoted component service.

The REST component allows you to expose those services as REST services that can consume Badger fish JSON, Standard JSON, or XML. You can add multiple bindings and multiple types of bindings on the same composite service. That means the same service can expose SOAP, JMS, and REST interfaces to service consumers at the same time.

Typical use cases include the following:

- Mobile clients have to consume an ActiveMatrix SCA service.
- Web clients or scripting clients (thin clients) participate in SCA.
- Mashups or websites have to expose services as APIs.

The REST bindings are especially helpful in the following situations:

- Mobile devices need to interact with back-end applications and services.
- Mobile application developers find it difficult to program the SOAP stack on the client side.
- Developers use modern scripting languages like JavaScript and Ruby, which provide first-class support for JSON and XML processing.

In these cases, the REST component allows the clients to easily invoke ActiveMatrix SCA services.

For information on REST components, see *TIBCO ActiveMatrix® Service Grid REST Binding Development*.

# TIBCO ActiveMatrix Policy Director Governance

---

TIBCO ActiveMatrix Policy Director Governance is a governance solution to manage and enforce cross-functional requirements such as security, monitoring, and compliance independent of their implementation and deployment.

Using both embedded agents and intermediary proxy applications, TIBCO ActiveMatrix Policy Director Governance can inspect, secure, log, and forward messages.

TIBCO ActiveMatrix Policy Director Governance can change the behavior of a service without changing the application code that results in a dynamic, policy-based service-oriented architecture (SOA) governance. For example, when you create, deploy, and manage simple object access protocol (SOAP) web services and representational state transfer (REST) services, security, logging, monitoring, and compliance could be involved in the process. The governance functions are independent of the service life cycle and ideally those services are not interrupted to change a policy.

The following are examples of policies provided in TIBCO ActiveMatrix Policy Director Governance:

Policy	Description
Authentication	Validates the username and password credentials from request messages.
Authorization	Filters the service request messages by checking that the requester has appropriate access permissions.
Encryption	Enforces the encryption on all messages between two applications (encrypting messages as they exit one application, and decrypting them as they enter the other application).
Credential Mapping	Automatically attaches appropriate credentials to request messages before they reach services.
Logging	When a request results in a fault message, it stores log details for later analysis.



## Advantages over Hard-Coded Policies

There are several advantages of declaring policies at run time than hard-coding policies into functional components.

### *Advantages of Declaring Policies at Run Time*

Advantage	Description
Division of Labor	Development teams can focus on implementing functionality while management, administration, and operations teams focus on formulating and implementing policies.
Leverage	Declarative policies generalize frequently used solutions into conventions (called policy templates). You gain leverage by using them many times.
Concise specification	You can use declarative policies to configure policy templates with a small number of parameters, which modifies its behavior to a specific business situation.
Comprehension	<ul style="list-style-type: none"><li>• Declarative policies are concise and based on standardized conventions, therefore they are easier to understand and verify than procedural code.</li><li>• The declarative language of policies specifies the desired result, and the distribution engine delivers operational details of policies.</li></ul>
Flexibility	<ul style="list-style-type: none"><li>• Declarative policies are easier to change than procedural code and respond quickly to changing business conditions.</li><li>• Declarative policies can change quickly to bring the enterprise into compliance. Instead of waiting for code changes, the distribution engine immediately applies and enforces new or modified policies.</li></ul>

## Features

TIBCO ActiveMatrix Policy Director Governance provides uniform cross-domain governance for TIBCO as well as non-TIBCO application platforms.


You can define a single policy in TIBCO ActiveMatrix Policy Director Governance and enforce it uniformly for the services running on TIBCO ActiveMatrix Service Grid and in the case of SOAP, other application platforms too.

## Embedded governance agents

By running in the same Java Virtual Machine as a TIBCO ActiveMatrix Service Grid node, embedded governance agents enforce security and other policies to provide last-mile security for SOAP and REST services. Last-mile security protects messages delivered to your service up to the point where TIBCO ActiveMatrix Service Grid application consumes them.

## Intermediary proxies

Intermediary proxies hide internal endpoints behind a firewall and expose proxy endpoints to external clients. The TIBCO ActiveMatrix Policy Director Governance proxy applications run on the TIBCO ActiveMatrix Service Grid nodes with embedded governance agents. You can enforce policies on these proxies the same way as you would on your internal TIBCO ActiveMatrix Service Grid applications.

 **Note:** Embedded governance agents are not available for other application platforms. Therefore, TIBCO ActiveMatrix Policy Director Governance proxies are the solution to enforce policies uniformly on your external SOAP services.

You can use TIBCO ActiveMatrix Policy Director Governance to create intermediary proxies to TIBCO ActiveMatrix Service Grid.

## Policy enforcement

Using both embedded agents and intermediary proxy applications, TIBCO ActiveMatrix Policy Director Governance can inspect, reject, log, transform, and redirect messages. Utilizing these capabilities TIBCO ActiveMatrix Policy Director Governance can enforce a broad range of policies.

You can perform the following actions with TIBCO ActiveMatrix Policy Director Governance:

- Discover services running in TIBCO ActiveMatrix Service Grid nodes.
- Register Web Services Description Language (WSDL) for the SOAP services running in any platform and create proxies for them.

- Create object groups of services or partner references using fixed membership or dynamic criteria. These groups are used as the targets for policies.
- Define resources such as LDAP, Keystores, and HTTP connections for use in policies.
- Define polices (known as Governance Controls in TIBCO ActiveMatrix Policy Director Governance) and deploy them to TIBCO ActiveMatrix Service Grid nodes for uniform policy enforcement.

You can complete Governance Control configurations such as an update to an authorization policy or a keystore password, in TIBCO ActiveMatrix Policy Director Governance and redeploy them to all runtime processes without updating an individual process or machine.

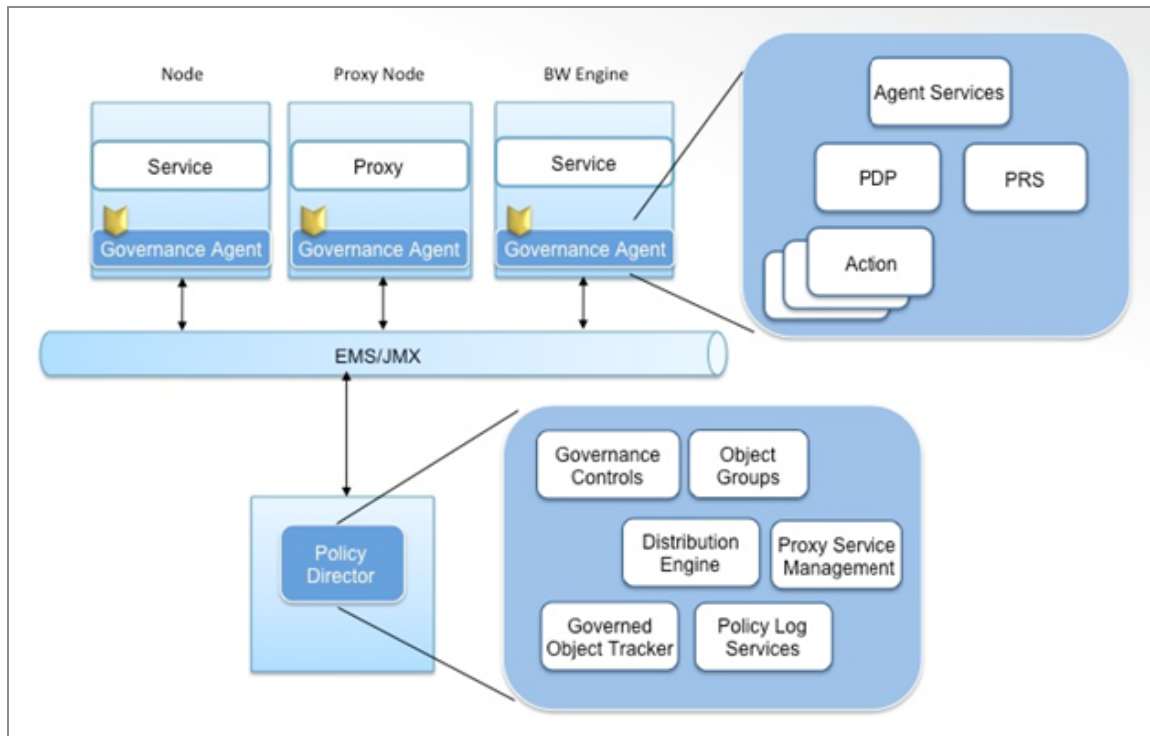
## Administration

TIBCO ActiveMatrix Policy Director Governance includes TIBCO ActiveMatrix Administrator, a graphical user interface for performing all of the prior tasks. It also has a comprehensive command-line interface for performing the same tasks.

# High-Level Architecture

TIBCO ActiveMatrix Policy Director Governance has two major components: administration and runtime enforcement.

## TIBCO ActiveMatrix Policy Director Governance High-Level Architecture



## Policy Director Governance Administration

TIBCO ActiveMatrix Policy Director Governance administration provides services to define and manage governance controls, object groups, shared resources, external services, and proxy applications.

TIBCO ActiveMatrix Policy Director Governance is typically, but not necessarily, installed and configured on a separate machine from your runtime engines. The administration component includes both TIBCO ActiveMatrix Administrator's browser-based and command-line interface for scripting policy configuration and management tasks.

One of the core administration components, the distribution engine, distributes governance control to different governance agents based on the selected object groups and where the governed objects run. The distribution engine uses the Enterprise Messaging Service (EMS) and Java Messaging Services (JMS) to communicate with governance agents to receive start-up events, deploy policy and resource configurations, and receive deployment status messages.

After the policies are deployed to the governance agent, you can take the administration components offline without affecting policy enforcement on each of the TIBCO ActiveMatrix Service Grid nodes.

## Governance Agent

A governance agent is a TIBCO ActiveMatrix Policy Director Governance component that runs in an application's Java processes such as the TIBCO ActiveMatrix Service Grid node.

Proxy endpoints are deployed on TIBCO ActiveMatrix Service Grid nodes designated as Proxy Nodes that also have an embedded governance agent for enforcing proxy policies.

The Governance Agent has core services to send information on start-up events, receive policy and resource configurations to TIBCO ActiveMatrix Policy Director Governance.

The Governance Agent also comprises the Policy Decision Point (PDP), Policy Runtime Store and Action components for enforcing policies on TIBCO ActiveMatrix Service Grid services and references.

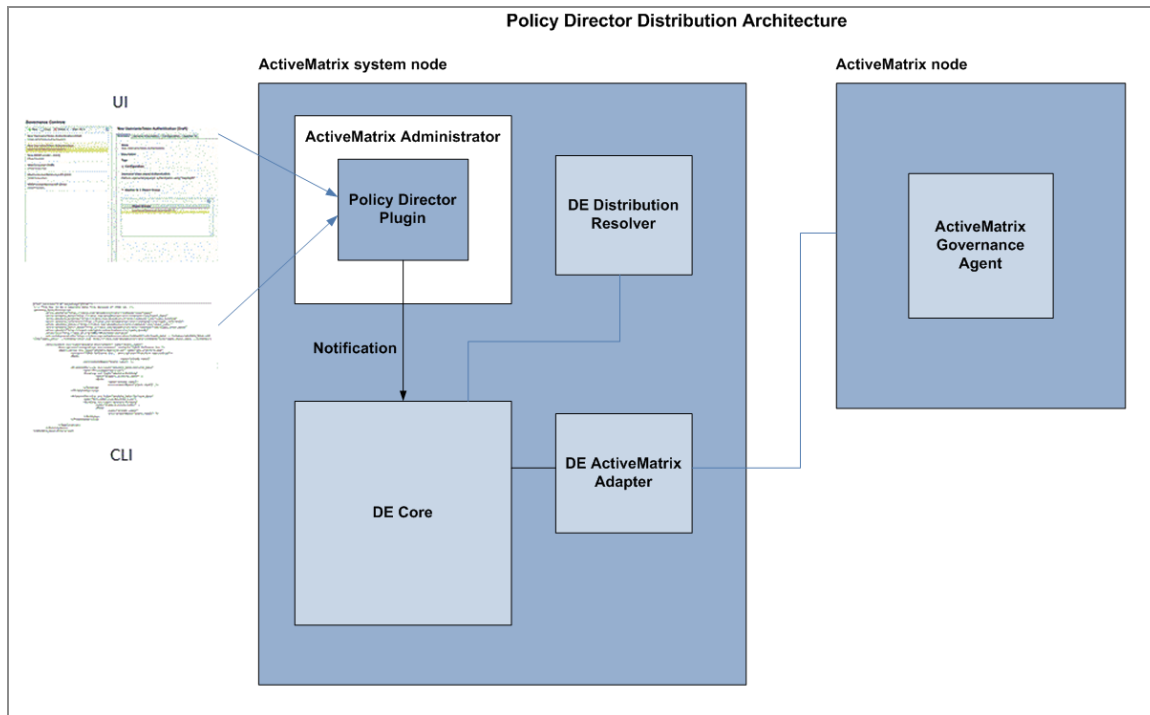
Policy Enforcement Point (PEP) is also installed as part of the Governance Agent. This component is specific to each platform supported by TIBCO ActiveMatrix Policy Director Governance since it is responsible for intercepting or consuming platform-specific events and interacting with PDP.

## Policy Distribution Engine

TIBCO ActiveMatrix Policy Director Governance distribution engine is designed for dynamic distribution of policies and resources to multiple governance agents.

The distribution engine is composed of TIBCO ActiveMatrix Service Grid components and deployed as a TIBCO ActiveMatrix Service Grid Application on the system node. It uses the TIBCO ActiveMatrix Grid Administrator's replication feature to support an environment with fault tolerance and load balance.

## TIBCO ActiveMatrix Policy Director Governance Distribution Architecture



You can use Policy Director Governance to define a single governance control and apply it to multiple Object Groups across the ActiveMatrix Service Grid and also non-TIBCO SOA platforms. The Distribution Engine's job comprises the following:

- Process all the object groups
- Resolve the Governed Object members
- Determine the Governance Agents on which each Governed object is running
- Compile appropriate policies for each Governance Agent

The distribution engine uses the TIBCO ActiveMatrix Grid Administrator's replication feature to support setups with fault tolerance and load balance.

## Extensions

The adapter and resolver extensions support the distribution engine, providing flexibility and scaling up for future Policy Enforcement Host platforms.

Adapter extensions communicate on the right channels with governance agents running on Policy Enforcement Hosts such as the TIBCO ActiveMatrix Service Grid. For example, a combination of TIBCO ActiveMatrix Service Grid protocols and TIBCO Enterprise Message

Service™ (EMS) is used to communicate with TIBCO ActiveMatrix Service Grid Governance Agent.

Resolver extensions resolve different kinds of governed objects and the governance agents responsible for enforcing policies on them.

The distribution engine also resolves the shared resource dependencies of the governance controls and deploys them to governance agents as needed. For example, if a resource configuration is changed, the distribution engine detects it and updates the governance agents based on the resource's governance control dependencies.

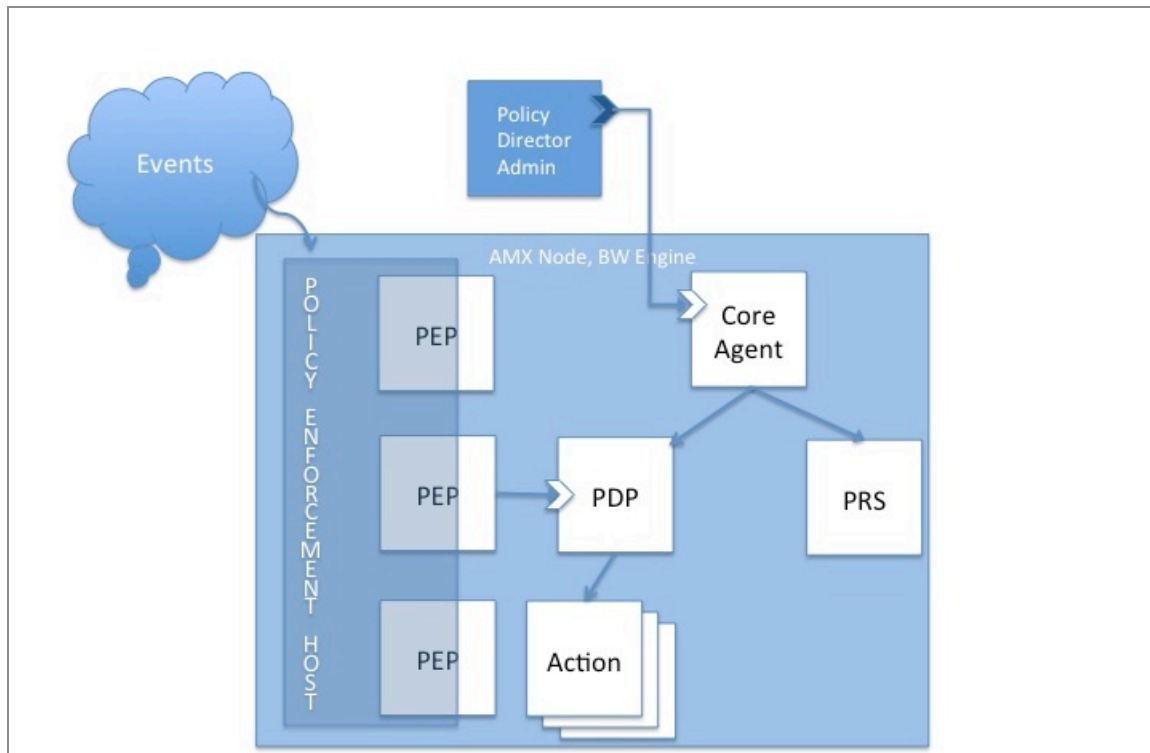
After a policy or resource is distributed to the required governance agents, the distribution engine waits for status reports from each governance agent and processes them as they arrive. The status is aggregated at the governance control level and displayed to the user in the TIBCO ActiveMatrix Policy Director Governance Administrator UI or reported in the command-line interface.

## Policy Enforcement

TIBCO ActiveMatrix Policy Director Governance uses embedded agents and intermediary proxy applications to enforce policies by inspecting, rejecting, recording, transforming, and redirecting messages.

The Governance Agent enforces policies even if the TIBCO ActiveMatrix Policy Director Governance Administrator services are not online. This is possible by maintaining a compiled version of the processed policies using its Policy Runtime Store component.

## Governance Agent



When the Governance Agent receives an instruction from TIBCO ActiveMatrix Policy Director Governance Administrator, it can perform any of the following functions:

- Process a new Governance Control
- Update an existing Governance control
- Delete an existing Governance Control

After processing the instructions, the Governance Agent updates its local cache to reflect the new state of policies to enforce a given PEP and Governed Object combination.

The Governance Agent completes all of these tasks without interrupting service availability. The actions to run are dynamically updated when the Governance Agent successfully processes the Governance Control instructions from TIBCO ActiveMatrix Policy Director Governance Administrator. If there are any policy conflicts or other errors while processing Governance Controls, the Governance Agent leaves the existing cache of policies to enforce undisturbed policies and reports the status to TIBCO ActiveMatrix Policy Director Governance Distribution Engine.

Using the Governance Control templates that TIBCO ActiveMatrix Policy Director Governance provides, you can only specify a few required parameters to define a policy.



However, behind the scenes, the compiled Governance Controls contain all of the information the Governance Agent needs to successfully process and enforce policies.

## Policy Enforcement Point Identifiers

Policy enforcement points are specific points such as event listeners or interceptors where policy enforcement is required. Each policy enforcement point has an identifier in TIBCO ActiveMatrix Policy Director Governance.

A policy enforcement host is an environment or domain that requires policy enforcement. For example, the SOAP Pipeline in TIBCO ActiveMatrix Service Grid is a policy enforcement host. Each policy enforcement host supports one or more specific points where policy enforcement is required.

The compiled governance control specifies one or more policy enforcement points and the policy actions that must be ran for a governed object in that policy enforcement point.

## Order of Action

Policy actions are not necessarily run in the same order that governance controls are defined and deployed.

For example, you may deploy an authorization policy for a policy first and then later decide to deploy a logging policy also. However, the governance agent logs the request before authorizing it.

Actions for a single policy enforcement point are run in a well-defined and consistent order regardless of the number of policies from which they are derived. The governance control lines up action by dividing the policy enforcement points into segments, which are called stages. Each stage is further divided into intervals for finer control of the action.

For example, consider the `serviceINFlow` policy enforcement point in the default section of the SPLINE host. It is identified by a QName as follows:

```
{http://tns.tibco.com/governance/policy/host}pipeline/default/serviceInFlow
```

The `service-InFlow` policy enforcement point has the following stages:

1. Receive
2. Crypto

3. Authentication
4. Authorization
5. Forward

Each of the above stages has three intervals:

1. Begin
2. Middle
3. End

**i Note:** The names of stages and intervals are relevant. Crypto operations such as decryption and encryption occur before authentication and authorization operations.

## Governance Agent's Merge Behavior

The governance agent ensures that new policy actions are successfully merged with the existing policy actions.

The policy decision point maintains a list of actions and the order in which to execute them for a particular governed object in a policy execution point. The list of actions comes from one or more policies applied to the governance agent.

Whenever a policy is applied to a governed object and policy execution point, the governance agent performs the following actions:

- ensures that new actions do not clash with existing actions
- positions the new action
- updates the list of actions

Each action can declare two attributes to specify how to merge instances with existing actions.

The first attribute defines the **placement** of the action in a given interval. The available options are:

Option	Description
First In Interval	<p>This means you must place the action before all other actions across types in the same stage and interval. If another action is already placed first <i>due to specifying this attribute value</i>, an error is reported because there can be only one action with this placement value in a given interval for a given policy enforcement point and GO combination.</p> <p>An example of an action that wants to specify a placement value is a Timer Start Action that begins the measurement of a service response time.</p>
Last In Interval	<p>This means you must place the action after all other actions across types in the same stage and interval. If another action is already placed last <i>due to specifying this attribute value</i>, an error is reported because there can only be one action with this placement value in a given interval for a given policy enforcement point and GO combination.</p> <p>An example of an action that wants to specify a placement value is a Timer End Action that finishes the measurement of a service response time.</p>
Sequential In Interval	<p>This is the default value if the action does not explicitly specify a placement value. You must place the action instance after all other actions across types in the same stage and interval, but before an action that specified Last In Interval placement value.</p>



**Note:** Note that the placement of attributes takes effect across all action types in a given interval.

The second attribute defines the **cardinality** of a single action type. The available options are:

Option	Description
Singleton In Interval	<p>This means that there can only be one instance of an action type in a given interval for a policy enforcement point and governed object. To add an action if another action of the same type is already present in that interval is invalid. If this occurs, an error is reported.</p>

Option	Description
Singleton In Stage	This means that there can only be one instance of an action type in a given stage across its intervals for a policy enforcement point and governed object. To add an action if another action of the same type is already present in any of the intervals of that stage is invalid. If this occurs, an error is reported.
Singleton In PEP	This means that only one instance of an action type can be present in a given policy enforcement point across stages and intervals for a given governed object. To add an action if another action of the same type is already present in any of the stages and intervals of that policy enforcement point is invalid. If this occurs, an error is reported.
Unbounded	This is the default value if the action does not specify a cardinality value. There is no restriction on the number of instances of an action type in the policy enforcement point, stage, and interval for a given action.

## Governance Definitions and Concepts

Many of the definitions and concepts used in TIBCO ActiveMatrix Policy Director Governance contain definitions and concepts from the Service Component Architecture policy specification and common terminology for policy-based management.

## Governance Controls

*Governance control* is responsible for the monitoring and enforcement aspects of runtime governance.

TIBCO ActiveMatrix Policy Director Governance provides governance control with runtime policies, policy templates, a distribution engine, and governed object groups.

## Governance Control Templates

A TIBCO ActiveMatrix Policy Director Governance *governance control template* encapsulates a common behavioral pattern, such as authenticating user names and passwords against LDAP. A template is used in the process of defining a governance control policy.

You can apply templates to a variety of management situations to achieve runtime goals such as authentication, authorization, encryption, and logging.

TIBCO ActiveMatrix Policy Director Governance provides the following governance control template types for run time:

**Authentication policies**

Require that requesting users authenticate their identities, using one of the several available.

**Credential-Mapping policies**

Maps user credentials between two realms, for example, between departments of an enterprise, or between enterprises.

**Authorization policies**

Require that governance agents verify a user is authorized to request an operation.

**Logging policies**

Record messages as they enter and leave an endpoint, so you can examine the messages later.

**Messaging policies**

Define how messages are sent and received.

## Governed Object

A Governed Object is the target of governance controls.

Each Governed Object belongs to an object domain such as ActiveMatrix Service Grid and has an object type.

ActiveMatrix Service Grid object domain supports Governed Objects such as service endpoints or reference (partner invocation) endpoints that can further be narrowed down by transport (HTTP or JMS) and by protocol (REST or SOAP). ActiveMatrix Service Grid also supports governed objects such as virtualization services and references as well as WebApp components.

Policies that are relevant to service-side governed objects differ from policies that apply to client-side governed objects. For example, the Authentication policy applies to service type

of Governed Objects and the Credential-Mapping policy applies to reference type of Governed Objects.

## Object Groups

An *object group* is a user-defined set of governed objects.

You can assign governed objects to a group of similar governed objects to manage and use them as a unit during run time.

### Object Groups and Object Group Types

A governed object can be a logical object, such as an ActiveMatrix application, or a physical object, such as a component instance. An object group always contains the same type of governed objects, and can consist of hosts, services, references, applications, and machines. For example, an object group can consist of all machines that run Linux, or can consist of web applications that run in the DMZ, or of services that a Claims system uses.

TIBCO ActiveMatrix Policy Director Governance supports several of the following object group types:

- Service Bindings
- Reference Bindings
- Components
- Instances (for reference bindings, service bindings, and components)

### Defining Object Groups

You can define the following object groups in two ways:

- Fixed, with governed objects that are explicitly added and do not change.
- Dynamic, or defined by criteria, with governed objects that move in and out of the group as they meet the standards set for membership.

When an object group is dynamic, you can apply the appropriate governance policies to any governed object that the system discovers in the future.

## Ways to Use Object Groups

Use an *object group* to combine governed objects that have the same governance requirements and to apply the same policies to that group.

Examples of using an object group to apply policies to related objects include:

- Apply an encryption policy to all finance services.
- Apply a message logging policy to all proxy services.

After you create an object group by combining governed objects with the same governance requirements, you can apply the same policies to the group.

For example, you can apply:

- An encryption policy to all finance services
- A message logging policy to all proxy services

## Shared Resources

Shared resources contain connection details for physical resources. Shared resources provide a way to use an identifier to reference the configuration of a resource. Using the identifier, multiple governing objects that require similar resource configuration can refer to an already configured resource without having to configure the resource multiple times.

When using shared resources you do not need to provide connection details in services, component implementations, and references. Instead, specify a property of the type of required physical resource in the service, component, or reference. When you configure the application for deployment, map the property to a resource instance available on the node on which the service, component, or reference is deployed.

With TIBCO ActiveMatrix Policy Director Governance, you can manage the configuration of resources such as LDAP, Keystores, Identity Providers, and HTTP connections separately from your policy configurations and reuse them in different policy configurations. For example, you could use a single Keystore resource in the WSS Provider policy and Credential Mapping policy.

Separating resources from policies offer many benefits. A security architect could define Identity Providers, Trust Providers, and WSS Authentication mechanisms to suit the interaction agreements. This can be done with partners independent of the application topologies and policy administration. The Administrator can manage applications, policies, and monitor systems.

## Policy Enforcement Host

A Policy Enforcement Host is an environment that generates the events that are candidates for policy enforcement.

A single Policy Enforcement Host supports one or more Policy Enforcement Points. TIBCO ActiveMatrix Service Grid is an example of a Policy Enforcement Host.

## Policy Enforcement Point

A Policy Enforcement Point is an interceptor or event listener that resides in a Policy Enforcement Host.

It interacts with the Governance Agent's [Policy Decision Point](#) by requesting an enforcing policy decision. Policy decisions are actions to be ran for a given Policy Enforcement Point and Governed Object.

## Policy Decision Point

Policy Decision Point is the component of the Governance Agent that makes policy decisions when requested by a Policy Enforcement Point.

It uses information from Policy Enforcement Point and Governed Object to decide which policy actions to execute.

## Policy Action

A policy action defines an action to perform to enforce a policy rule when the conditions of the rule are met.

When a policy is enforced, it may result in one or more actions executed to fulfill the requirements of the policy. For example, when you define and deploy a WS-Security policy, the Governance Agent may run decryption, authentication, and signature verification actions.



## Policy Runtime Store

Policy Runtime Store is the Governance Agent component that contains policy rules, policy conditions, actions, and policy data for use by Policy Decision Point.

Runtime optimization is a key requirement, so the Policy Runtime Store contains proprietary artifacts that are efficiently interpreted by Policy Decision Point.

## Policy Error

Policy errors occur when an attempt to enforce policy action fails.

Policy actions may fail due to a temporary state or a permanent mismatch between the policy action and the device enforcement capability. However, this is not the same as [Policy Conflict](#).

## Policy Conflict

A policy conflict occurs when the actions of two rules that are both satisfied simultaneously contradict each other.

Policy conflict may happen in TIBCO ActiveMatrix Policy Director Governance if you attempt to deploy two conflicting policies on the same governed object.

The governance agent uses *MergeCardinality* and *MergePlacement* metadata in the Action definitions to determine and avoid a policy conflict. When you deploy a governance control in TIBCO ActiveMatrix Policy Director Governance, each governance agent receiving the policy configurations reports all policy conflict errors. The deployment fails and you can review and resolve the errors.

For more information, see [Policy Enforcement](#).

# Glossary

---

Product	Term	Definition
Active Matrix	<b>authentication realm</b>	Defines the method of storing and retrieving information about ActiveMatrix users and groups. ActiveMatrix Administrator supports the following authentication realms: Local XML File, Database, TIBCO Administrator, and LDAP.
Active Matrix	<b>binding (WSDL)</b>	A concrete protocol and data format specification for the operations contained in a port or interface.
Active Matrix	<b>binding definition set</b>	A container for binding definitions. The extension of a binding definition set file is .bdf.
Active Matrix	<b>cluster</b>	A group of ActiveMatrix Administrator servers. Servers in a cluster are maintained as clones, that is, they are automatically kept in an identical state for fail-over purposes. All servers in a cluster share the same database tables and authentication realm.
Active Matrix	<b>component</b>	Represents an implementation of one or more port types. The component attributes are: Name Implementation type Port types, provided and consumed by the component Links to services that export the provided port types Links to partner references or topics that export the consumed port types Links to artifacts that implement the provided port types.
Active Matrix	<b>component reference</b>	Component references describe the binding information required to invoke an external service. When the component implements a process that has a reference, then the process reference is exposed as a component reference.

Active Matrix	<b>component service</b>	Represents a service provided by a <a href="#">component</a> .
Active Matrix	<b>composite</b>	One of the outputs of the ActiveMatrix design phase. A container for composite elements. The suffix of a composite file is .composite.
Active Matrix	<b>composite element</b>	A child element of a composite. Services, components, references, and topics, wires, and the properties that components are configured with.
Active Matrix	<b>composite reference</b>	See <a href="#">reference</a> .
Active Matrix	<b>composite service</b>	See <a href="#">service</a> .
Active Matrix	<b>endpoint</b>	A combination of a binding and a network address. The URL at which a consumer can access a service. An <i>internal endpoint</i> is accessible only to consumers within an ActiveMatrix environment. An <i>external endpoint</i> has a binding that provides access to consumers outside the ActiveMatrix environment. See <a href="#">port (WSDL 1.1)</a> .
Active Matrix	<b>environment</b>	A logical grouping of ActiveMatrix nodes administered as one entity.
Active Matrix	<b>HTTP</b>	Hypertext Transfer Protocol. The Internet protocol used to retrieve hypertext objects from remote hosts. HTTP messages consist of requests from client to server and responses from server to client. HTTP/S is a secure version of HTTP.
Active Matrix	<b>interface</b>	A collection of operations. See also <a href="#">port type (WSDL 1.1)</a> .
Active Matrix	<b>JDBC</b>	A Java API that allows applications to invoke SQL commands to create database tables, access the data stored in a table, and create and manage distributed

		transactions.
Active Matrix	<b>JMS</b>	A Java API that allows applications to create, send, receive, and read messages. The JMS API enables communication that is decoupled, asynchronous, and reliable. JMS can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them. JMS can ensure that a message is delivered once and only once. Lower levels of reliability are available for applications that can afford to miss messages or to receive duplicate messages.
Active Matrix	<b>JNDI</b>	A Java API that allows applications to store and retrieve named Java objects of any type. In addition, JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.
Active Matrix	<b>keystore</b>	A database of private keys and their associated X.509 certificate chains authenticating the corresponding public keys.
Active Matrix	<b>keystore entry</b>	An entry in a keystore. An entry can be a trusted certificate entry, which contains a single public key certificate belonging to a trusted party or a key-certificate entry, which contains a private key and a corresponding public key certificate. Each entry in a keystore is identified by an alias.
Active Matrix	<b>location transparency</b>	Network locations that provide services are hidden from the service consumers (and replaced with some other network location). The actual locations are transparent, that is, not visible to the service consumer.
Active Matrix	<b>mediation</b>	Performing programmatic tasks on messages in between service providers and the service consumer, to enhance the services provided.
Active	<b>mediation flow</b>	Software program that performs activities between

Matrix		consumers and providers of services, such as routing and transforming messages.
Active Matrix	<b>mediation service</b>	Service provided by a mediation component. The service provided combines one or more target services, which provide business functions, with mediation functions such as routing and data transformation.
Active Matrix	<b>mediation task</b>	Set of TIBCO Business Studio plug-ins that performs a mediation function. Mediation tasks are placed on mediation paths, and are processed in sequence at runtime to provide mediation logic (some coherent set of mediation tasks designed to support mediation goals, in the same way that business logic supports business goals).
Active Matrix	<b>message exchange pattern (MEP)</b>	The sequence and cardinality of messages sent between the consumer and the provider. The messages include both normal and fault messages.
Active Matrix	<b>Messaging bus</b>	The message exchange layer responsible for routing messages between service providers and consumers. Messaging Bus supports performing such message delivery with varying qualities of service depending on application needs and the nature of the messages being delivered. It also performs transaction propagation, policy enforcement, and so on.
Active Matrix	<b>node</b>	A Java Virtual Machine running the ActiveMatrix Platform and the Messaging Bus.
Active Matrix	<b>normalized message</b>	The messaging unit of interoperability between ActiveMatrix containers. It uses abstract WSDL message type definitions and consists of two parts: Content An XML document that conforms to an abstract WSDL message type, without any protocol encoding or formatting. (This is not a canonical form for the message.) Properties Data associated with a message gained during the processing of the message. Such properties can include security

		information (security principal for digital signers of received messages), transaction context information, and container-specific information.
Active Matrix	<b>point-to-point messaging</b>	A messaging system involving a single sender and receiver. It is built on the concept of message queues. Senders send messages addressed to a specific queue; receivers extract messages from the queues established to hold their messages.
Active Matrix	<b>port (WSDL 1.1)</b>	A combination of a binding (WSDL) and a network address.
Active Matrix	<b>port type (WSDL 1.1)</b>	A collection of operations.
Active Matrix	<b>publish/subscribe messaging system</b>	A messaging system in which clients address messages to a specific node in a content hierarchy. Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a node's multiple publishers to its multiple subscribers.
Active Matrix	<b>QName</b>	A qualified name, a data type specified by XML Schema. A qualified name consists of a namespace URI and a local part.
Active Matrix	<b>queue</b>	A queue.
Active Matrix	<b>reference</b>	Represents an endpoint consumed by an ActiveMatrix composite element. A composite reference represents the endpoints consumed by the composite. A composite reference enables ActiveMatrix components to consume port types implemented outside ActiveMatrix. The reference attributes are: Name Port types, provided by the reference Bindings that enable composite elements to access the provided port types Resource profiles

---

Active Matrix	<b>resource template</b>	<p>An object that allows services, components, and references to share access to communication, data, naming, and security resources. The shared resource types supported by ActiveMatrix are:</p> <ul style="list-style-type: none"><li>• Hibernate</li><li>• HTTP Client</li><li>• HTTP Connector</li><li>• Identity Provider</li><li>• JDBC</li><li>• JMS Connection Factory</li><li>• JMS Connection Factory Configuration</li><li>• JMS Destination</li><li>• JMS Destination Configuration</li><li>• JNDE Connection Configuration</li><li>• Kerberos Authentication</li><li>• Kerberos Identity Provider</li><li>• Keystore Provider</li><li>• LDAP Authentication</li><li>• LDAP Connection</li><li>• Mutual Identity Provider</li><li>• OpenID Authentication</li><li>• SAML SSO Web Profile Authentication</li><li>• SiteMinder Authentication</li><li>• SMTP</li><li>• SSL Client Provider</li><li>• SSL Server Provider</li></ul>
---------------	--------------------------	---

---

		<ul style="list-style-type: none"> <li>• Teneo</li> <li>• Thread Pool</li> <li>• Trust Provider</li> <li>• WSS Authentication</li> </ul>
Active Matrix	<b>resource template profile</b>	Represents a reference to a shared resource.
Active Matrix	<b>SOA</b>	Service Oriented Architecture. A software architecture in which applications and data are decomposed into discrete, operationally independent services, which can be executed in a highly distributed manner. SOA is based on three principles: Modularity — Breaking tasks and services into smaller tasks or services; Encapsulation — Clearly defined interfaces that insulate a service's internal workings from outside contact; Reuse — Re-assembling services into composite applications that support new business processes; SOA inherently encompasses a heterogeneous collection of platforms and sources. In particular, enterprise applications may be hosted on Java and .NET platforms as well as third-party packaged systems and legacy applications. SOA infrastructure spans these varied application architectures and provide mediation between them.
Active Matrix	<b>SOAP</b>	Simple Object Access Protocol. A protocol for exchanging XML-based messages over a computer network, normally using HTTP. SOAP also defines a way to perform remote procedure calls (RPCs) using HTTP as the underlying communication protocol. SOAP forms the foundation layer of the web services stack. SOAP 1.1 and SOAP 1.2 are supported.
Active Matrix	<b>REST</b>	Representational State Transfer. This software architectural style defines the constraints to create web services. The web services that follows the REST architectural style are called RESTful Web Services.



Active Matrix	<b>TIBCO ActiveMatrix Administrator</b>	Product component for administering TIBCO ActiveMatrix infrastructure and services.
Active Matrix	<b>TIBCO Business Studio</b>	Product component for developing TIBCO ActiveMatrix SOA projects. You can execute composites in run mode or in debug mode. Running or debugging in a local node is also referred to as rapid application development (RAD) because you do not have to package and deploy the composite to test it.
Active Matrix	<b>TIBCO Enterprise Message Service</b>	TIBCO product that implements JMS. ActiveMatrix Messaging Bus uses Enterprise Message Service as the messaging backbone.
Active Matrix	<b>topic</b>	Represents a publish-subscribe messaging channel between service consumers and providers. Topics can have multiple consumers and can consume multiple providers. Topic attributes include: Name Port type provided by the topic Links to components or references that export the provided port type See also publish/subscribe messaging system.
Active Matrix	<b>web service</b>	A software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that is described in a machine-processable format such as WSDL. Other systems interact with the web service in a manner prescribed by its interface using messages, which may be enclosed in a SOAP envelope.
Active Matrix	<b>wire</b>	Represents a connection between composite elements.
Active Matrix	<b>WSDL</b>	Web Services Definition Language. An XML-based language for describing web services. The services are described in WSDL documents or files. A client program connecting to a web service can read the WSDL document to determine what operations are available on the server. Data types referenced in the document are embedded in the WSDL

		file in the form of XML Schema. The client uses SOAP to invoke the operations listed in the WSDL.
Active Matrix	<b>XML</b>	Extensible Markup Language. A text-based markup language in which tags (markup) identify the content, data, and text in the documents. Although tags can be defined as needed in the generation of an XML document, a document type definition (DTD) or XML Schema is usually used to define the elements allowed in a particular type of document. An XML document can be compared by using the rules in the DTD or XML Schema to determine its validity and to locate particular elements in the document.
Active Matrix	<b>XML schema</b>	An XML language for describing the structure of XML documents. The suffix of an XSD document is .xsd.
Active Matrix	<b>JSON</b>	JavaScript Object Notation. JSON is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays.
Active Matrix	<b>operation</b>	An exchange of messages, between a service consumer and provider, according to a message exchange pattern that is understood by both participants. A WSDL operation, or its name. The part of a SOAP message that specifies the WSDL operation that a consumer is requesting from a service.
Active Matrix	<b>Policy Directory agent</b>	Agents are the Policy Directory components that enforce policies. Agents are embedded in ActiveMatrix nodes; external proxy agents can be deployed outside of ActiveMatrix nodes.
Active Matrix	<b>Policy Directory criteria</b>	A set of target services for a policy, or a query expression that determines a set of target services.
Active Matrix	<b>Policy Directory endpoint</b>	An HTTP address for interacting with a service. A combination of a binding and a network address. (WSDL

		2.0)
Active Matrix	<b>Policy Directory external service</b>	A service deployed outside of any ActiveMatrix node.
Active Matrix Policy Directory	<b>inflow</b>	The part of a message exchange that flows from a consumer toward a provider.
Active Matrix Policy Directory	<b>message exchange</b>	A pairing of a request message (inflow) and either a response message or a fault message (outflow).
Active Matrix Policy Directory	<b>outflow</b>	The part of a message exchange that flows from a provider back toward the consumer.
Active Matrix Policy Directory	<b>policy endpoints</b>	The subset of endpoints that become visible to Policy Directory when you deploy an ActiveMatrix service assembly. ActiveMatrix instructs Policy Directory to register and manage these endpoints.
Active Matrix Policy Directory	<b>proxy endpoint</b>	A managed endpoint for an external service, managed by a proxy agent. An additional managed endpoint for a service deployed in an ActiveMatrix node.
Active Matrix Policy Directory	<b>SOAP</b>	A web services protocol standard for making web services available remotely. Consumers and providers of web services exchange XML-based messages over a computer network according to the SOAP protocol. See also WSDL (Web Services Description Language).
Active Matrix Policy	<b>SOAP binding type</b>	When an ActiveMatrix service communicates using ActiveMatrix normalized messages, and an external service communicates using SOAP messages, the SOAP binding

Directory		engine translates their messages between these two protocols, so the services can interoperate.
Active Matrix Policy Directory	<b>WSDL</b>	An XML-based language for describing web services. The services are described in WSDL documents or files. A client program connecting to a web service can read the WSDL document to determine the operations that are available on the server. Data types referenced in the document are embedded in the WSDL file in the form of XML Schema. The client uses SOAP to invoke the operations listed in the WSDL file. See also SOAP, web service, WSDL file.
Active Matrix Policy Directory	<b>alert</b>	A notification to an end-user, for example, scheduled alerts deliver portal headlines to a chosen device.
Active Matrix Policy Directory	<b>domain</b>	In TIBCO Administrator, two kinds of domains are used: administration domain and application domain.
Active Matrix, Active Matrix Policy Directory	<b>policy</b>	A rule or property that dynamically affects the behavior of a service.
Active Matrix, Active Matrix Policy Directory	<b>WSDL file</b>	An XML document that describes the services offered by a service provider. A WSDL file can take two forms: abstract and concrete. An abstract WSDL file defines operations, which define specific message request and response formats, and data types, which describe the types of the objects passed in the messages. A concrete WSDL file contains the abstract WSDL plus the communication protocols and data formats by which the operations defined in the abstract WSDL can be invoked. A binding

---

		connects a port type or interface (a collection of operations) to a protocol and data format. The combination of a binding and a network address is a port (WSDL 1.1) or an endpoint (WSDL 2.0). A service is a collection of ports (WSDL 1.1) or endpoints (WSDL 2.0).
Active Matrix	<b>consumer</b>	A software service that sends and receives messages defined by a WSDL document to a provider.
Active Matrix	<b>operation</b>	An exchange of messages between a service consumer and provider according to a message exchange pattern that is understood by both participants.
Active Matrix	<b>perspective</b>	In TIBCO Business Studio, a set and layout of views in the Workbench window. In ActiveMatrix Administrator, a set of screens and controls used to carry out a category of administration tasks.
Active Matrix	<b>provider</b>	A software service that conforms to a published WSDL document. The WSDL document describes the services it offers and the messages required to access the services. The provider generates and responds to messages sent by a consumer.

---

# TIBCO Documentation and Support Services

---

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [Product Documentation website](#), mainly in HTML and PDF formats.

The [Product Documentation website](#) is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

The documentation for this product is available on the [TIBCO ActiveMatrix® Service Grid Product Documentation](#) page.

## How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our [product Support website](#).
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the [product Support website](#). If you do not have a username, 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 [TIBCO Community](#).

# Legal and Third-Party Notices

---

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document 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 Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, ActiveMatrix, Business Studio, Enterprise Message Service, and Hawk are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or 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. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG 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 a specific version of Cloud SG software 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. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

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 "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at <https://www.tibco.com/patents>.

Copyright © 2010-2025. Cloud Software Group, Inc. All Rights Reserved.