TIBCO ActiveMatrix® BPM SOA Concepts

Software Release 4.2

August 2017

Two-Second Advantage®

Important Information

Contents

Figures 5

TIBCO Documentation and Support Services 6

Introduction 8

Challenges of Service Delivery 9

Design 10

Design Activities 10

Runtime 11

Runtime Overview 11

Hosts 12

Nodes 13

Features 14

Resource Templates 14

Resource Instances 15

Administration 16

Overview of ActiveMatrix Administrator 16

TIBCO ActiveMatrix 16

Enterprise 17

Environments 18

Messaging Bus 19

Governance 19

Service Registry 20

Identity Management 20

Monitoring Service 20

Policy Management 20

Service Component Architecture 21

Composites 21

Components 22

Component Implementation 22

Services and References 23

Interfaces 23

Bindings 23

Properties 24

Message Exchange Patterns 25

Java Components 25

Composite Components 26

Mediation Components 26

WebApp Components 28

Spring Components 29

Figures

TIBCO ActiveMatrix Runtime 11
Runtime Configuration 12
Multiple Node Scenario 14
TIBCO Business Studio Resource Instance 15
Administrator Resource Instance 15
TIBCO ActiveMatrix Administration 16
TIBCO ActiveMatrix Administrator 17
Cross Environment Wires 19
Composite 21
Composite Editor 22
Bindings 24
Mediation 27
Mediation Flow 28

TIBCO Documentation and Support Services

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, visit:

https://docs.tibco.com

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site.

The following documents form the documentation set:

- *Concepts*: Read this manual before reading any other manual in the documentation set. This manual describes terminology and concepts of the platform. The other manuals in the documentation set assume you are familiar with the information in this manual.
- *Development Tutorials*: Read this manual for a step-by-step introduction to the process of creating, packaging, and running composites in TIBCO Business Studio.
- *Composite Development*: Read this manual to learn how to develop and package composites.
- *Java Component Development*: Read this manual to learn how to configure and implement Java components.
- *Mediation Component Development* : Read this manual to learn how to configure and implement Mediation components.
- *Mediation API Reference* : Read this manual to learn how to develop custom Mediation tasks.
- *Spring Component Development* : Read this manual to learn how to configure and implement Spring components.
- *WebApp Component Development* : Read this manual to learn how to configure and implement Web Application components.
- *Administration Tutorial*: Read this manual for a step-by-step introduction to the process of creating and starting the runtime version of the product, starting TIBCO ActiveMatrix servers, and deploying applications to the runtime.
- *Administration*: Read this manual to learn how to manage the runtime and deploy and manage applications.
- *Hawk ActiveMatrix Plug-in User's Guide*: Read this manual to learn about the Hawk plug-in and its optional configurations.
- *Installation and Configuration*: Read this manual to learn how to install and configure the software.
- *Release Notes*: Read this manual for a list of new and changed features, steps for migrating from a previous release, and lists of known issues and closed issues for the release.

The documentation for the following features is installed separately:

- TIBCO ActiveMatrix Implementation Type for C++
- TIBCO ActiveMatrix Binding Type for EJB
- TIBCO ActiveMatrix Binding Type for Adapters
- TIBCO ActiveMatrix Implementation Type for TIBCO Adapters
- TIBCO ActiveMatrix Implementation Type for Microsoft CLR
- TIBCO ActiveMatrix Binding Type for REST

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

How to Join TIBCO Community

TIBCO Community is an online destination for TIBCO customers, partners, and resident experts. It is a place to share and access the collective experience of the TIBCO community. TIBCO Community offers forums, blogs, and access to a variety of resources. To register, go to the following web address:

https://community.tibco.com

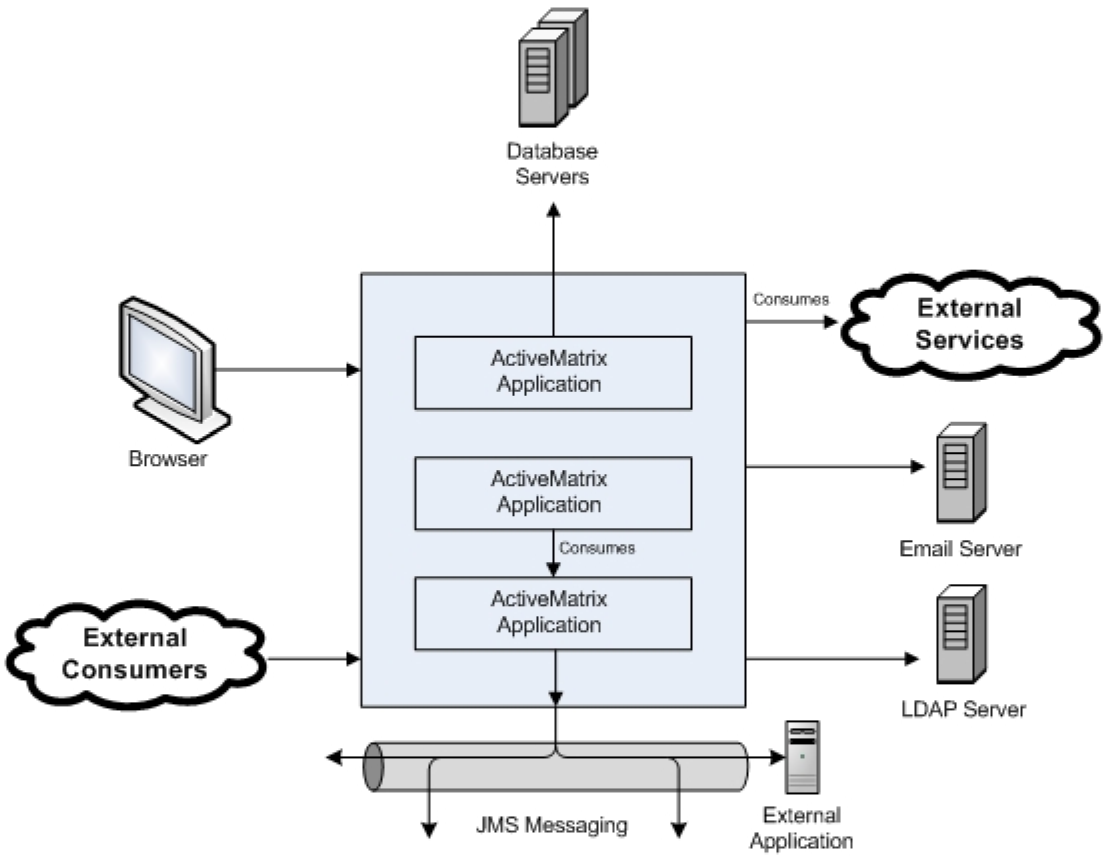
Introduction

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 Bus, 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 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 heterogenous 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 sofware 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 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).

See Service Component Architecture section for more information.

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 *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.

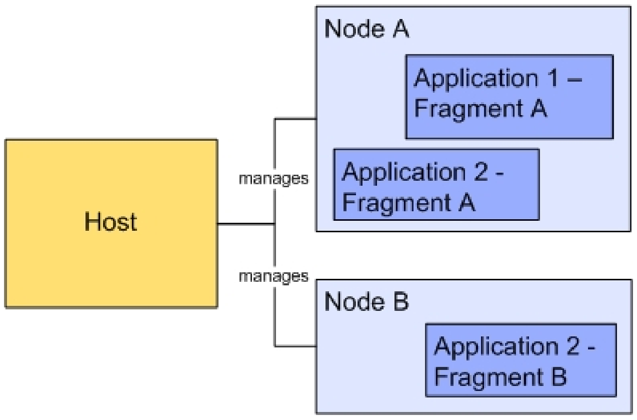
The TIBCO ActiveMatrix Administrator provides the deployment and run-time management for ActiveMatrix environments. In reality, the administrator too is an ActiveMatrix component running on a dedicated node.

Runtime Overview

TIBCO ActiveMatrix employs a three-level runtime environment consisting of hosts, nodes, and application fragments.

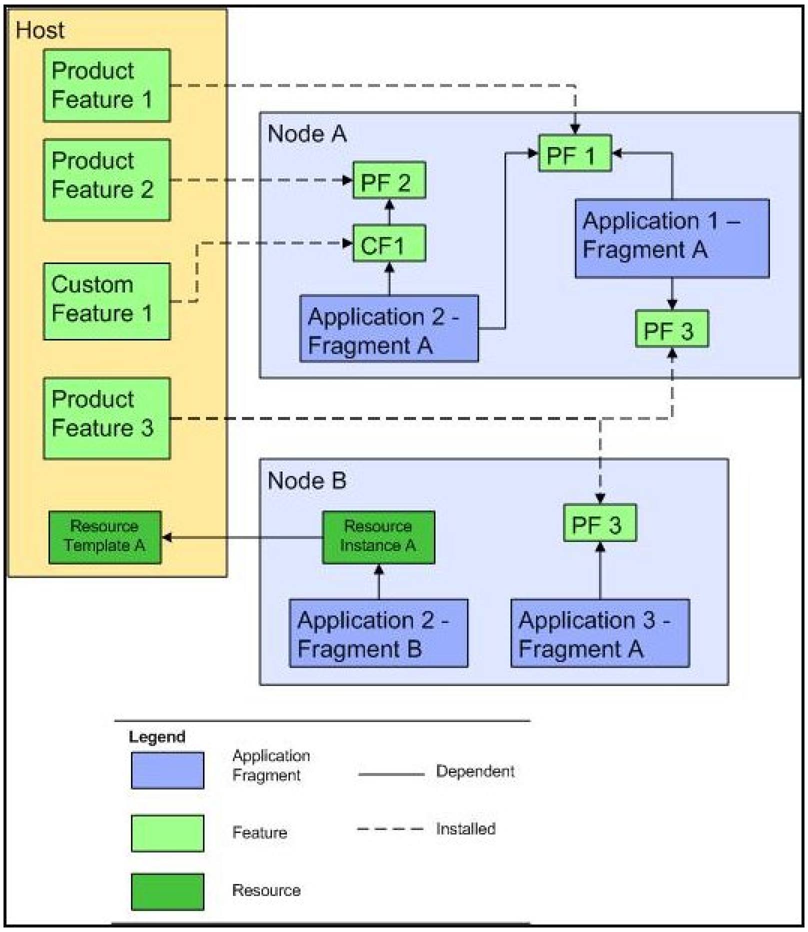
The following figure illustrates a possible configuration of hosts, nodes, and application fragments. The host manages node A and node B. Application 1- Fragment A runs on node A. Application 2's fragments are distributed over nodes A and B.

TIBCO ActiveMatrix Runtime


Host
manages
manages
Node A
Application 1 – Fragment A
Application 2 - Fragment A
Node B
Application 2 - Fragment B

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 Administrator server at a time. Hosts can contain nodes from multiple environments within one 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.

The host that manages the SystemNode node on which the Administrator server runs is named SystemHost.

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 in 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.

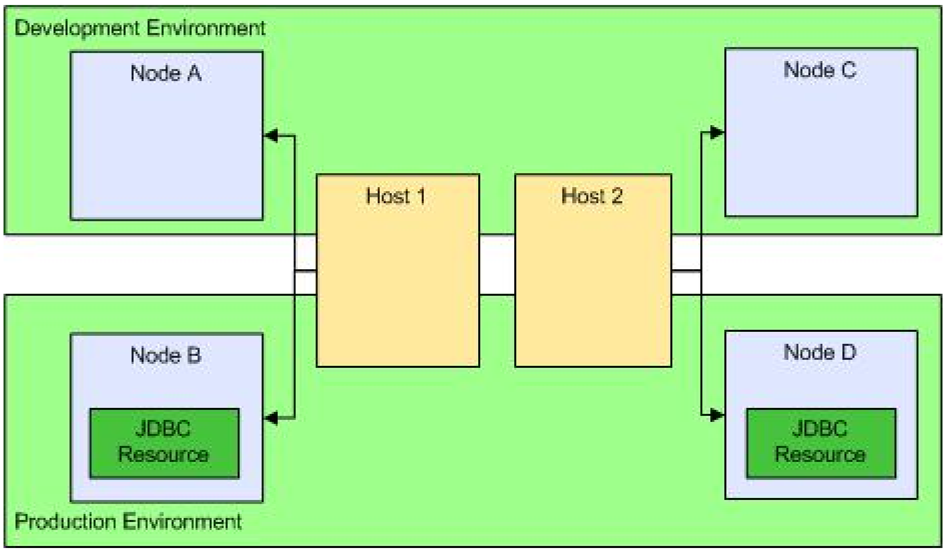
Application fragments are components or bindings of an application that are distributed and deployed to nodes. A fragment can be distributed to many nodes, and a single node can run many fragments. To increase throughput for a component or binding you can deploy multiple copies of the fragment to multiple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

The node on which the Administrator server runs is named SystemNode.

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. Shared library features contain component implementations and libraries. When you create a distributed application archive containing a composite, you can package the composite's required features in the application archive or you can package the features as a standalone distributed application archive.

When you upload a distributed application archive containing a composite in Administrator you can optionally import the features contained in the archive into the Administrator software repository. When you deploy an application, 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.

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.

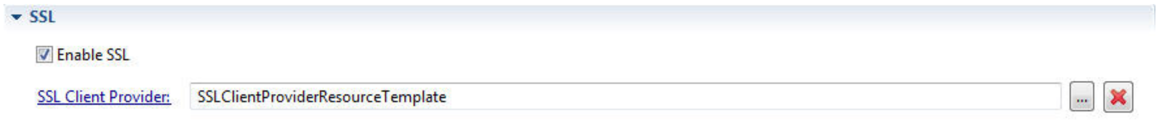
Resource Instances

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 on page 15 and Administrator Resource Instance on page 15 show how to set a resource property in TIBCO Business Studio and Administrator respectively.

TIBCO Business Studio Resource Instance

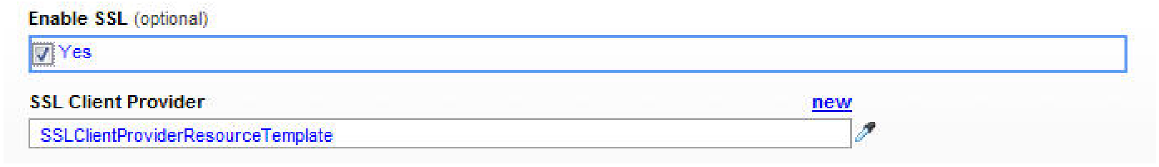


SSL

☑ Enable SSL

SSL Client Provider: SSLClientProviderResourceTemplate

Administrator Resource Instance



Enable SSL (optional)

☑ Yes

SSL Client Provider

SSLClientProviderResourceTemplate

Administration

TIBCO ActiveMatrix Administrator is a central administration server that let you create, deploy, and manage applications in TIBCO 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.

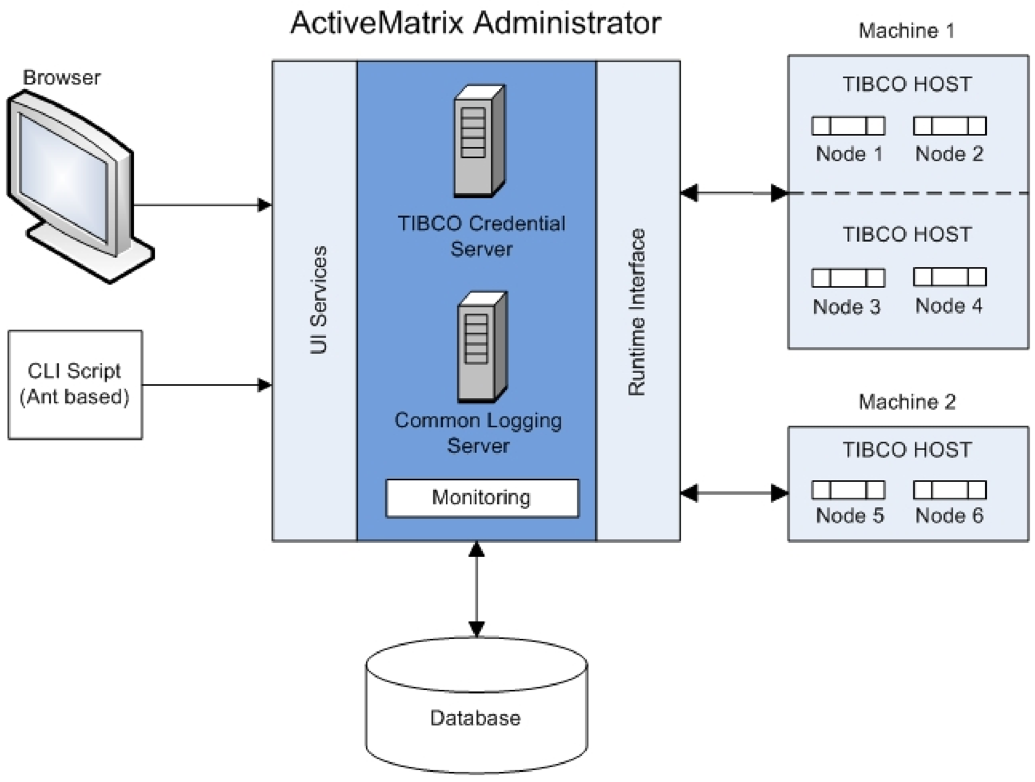
TIBCO ActiveMatrix administration is supported by TIBCO ActiveMatrix Administrator and TIBCO Business Studio. System administrators can do the following:

- configure environments and messaging buses
- register hosts and associate them with environments
- provision nodes with features and resources
- 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 it manages. This section provides an overview of the environment and Messaging Bus. Runtime on page 11 discusses hosts, nodes, applications, features, and resources.

TIBCO ActiveMatrix Administration



TIBCO ActiveMatrix

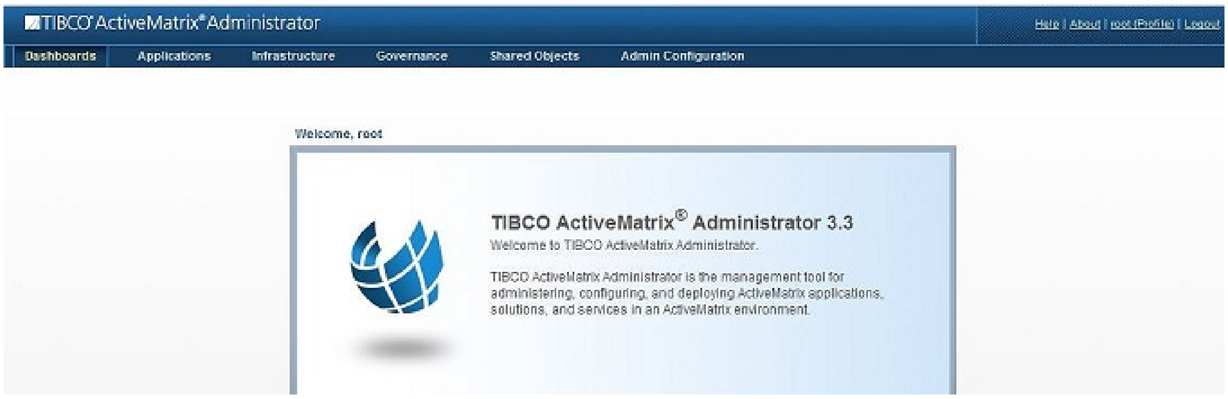
TIBCO ActiveMatrix Administrator consists of the following components:

- Administrator server
- Administrator server clients

- – Administrator web interface
- – Administrator command-line interface
- – TIBCO Business Studio

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

TIBCO ActiveMatrix Administrator



Welcome, root

TIBCO ActiveMatrix® Administrator 3.3

Welcome to TIBCO ActiveMatrix Administrator.

TIBCO ActiveMatrix Administrator is the management tool for administering, configuring, and deploying ActiveMatrix applications, solutions, and services in an ActiveMatrix environment.

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

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

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

- Log - Aggregates log data from nodes and saves to persistent store.
- Payload - Stores and retrieves large payloads for log entries.
- Monitoring -Aggregates performance data from nodes and saves to 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
- UDDI server - (optional) maintains published service data

The communication channels between Administrator servers and other servers can be secured with SSL. For information on SSL support, see the installation manual for your product.

For information on TIBCO ActiveMatrix Administrator, see *TIBCO ActiveMatrix 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 Shared Objects that are available across the enterprise. The objects are available in the 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 Administrator server.

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

Enterprise Objects

Enterprise Object	Location
Application Template	**Infrastructure** > **Software Management** > **Application Templates**
Environment	**Infrastructure** > **Environments**
Feature	**Infrastructure** > **Software Management** > **Features**
Host	**Infrastructure** > **Hosts**
Logging appender	**Shared Objects** > **Logging Appenders**
Resource template	**Shared Objects** > **Resource Templates**
Substitution variable	**Shared Objects** > **Substitution Variables**
User, superuser, and group	**Governance** > **Users and Groups**
Enterprise permission	**Governance** > **Enterprise Permissions**

Environments

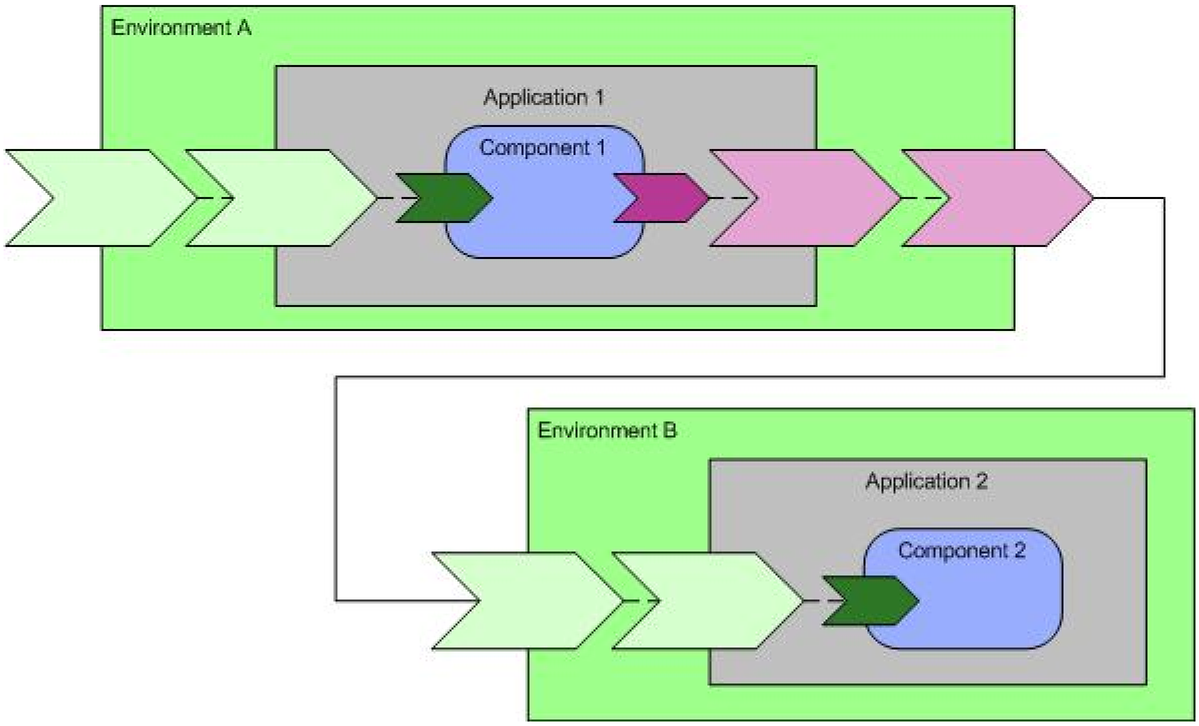
An *environment* is a logical grouping of applications and nodes. An Administrator server can have multiple environments. For example, you can define environments distinguished by product life cycle function such as development and production, by geographical area, or by business unit.

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 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** Nodes are 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 Administrator server runs, SystemNode, is named SystemEnvironment.

Development Environment

When you create an 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 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 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 metrics collection and monitoring and credentials that support identity management.

Service Registry

TIBCO ActiveMatrix can be configured to use any Universal Description, Discovery, and Integration (UDDI) registry and metadata repository that provides a standards-based system of record for discovering and publishing reusable business and IT services based on business classification and usage models.

When developing applications in TIBCO Business Studio, you can search a standards-based UDDI registry for services. You can configure a UDDI server in Administrator so that when you deploy an application in Administrator, the service is automatically registered in configured UDDI Server.

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.

Monitoring Service

The monitoring service and dashboards in TIBCO ActiveMatrix Administrator provide summary and detailed views into the operational health and performance of your TIBCO ActiveMatrix infrastructure, applications, and services.

The monitoring service is created when you create an Administrator server. For details, see the installation manual for your product. The monitoring service is configured with a database for storing performance data and the notification server that conveys the performance data from runtime objects to the monitoring service.

Basic monitoring provides one hour and since started time periods and a select number of metrics such as runtime state, requests, faults, response time. These features support advanced performance troubleshooting, but impose an additional load on the machine on which Administrator runs.

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 at the object.

TIBCO ActiveMatrix support for intents, policy sets, and policies conforms to the SCA Policy specification.

Service Component Architecture

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 ActiveMatrix support for service-oriented applications. Business function is provided as a set of components assembled into a structure called a composite.

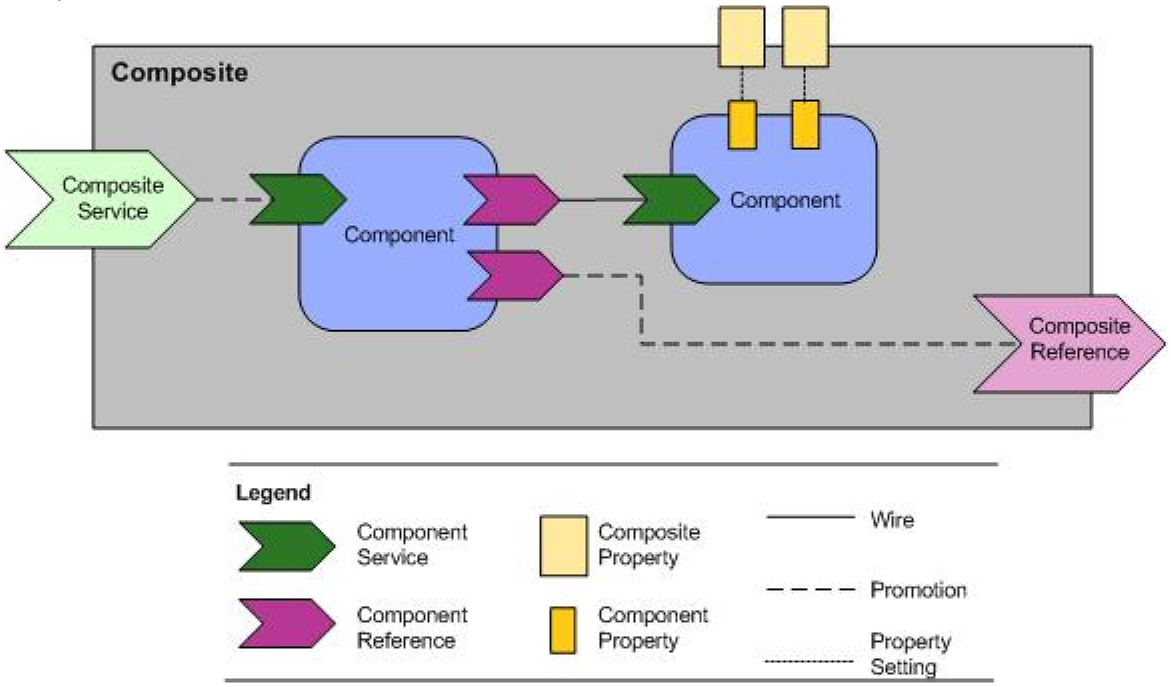
Composites

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.

See the SCA Assembly specification for information on service-oriented architectures.

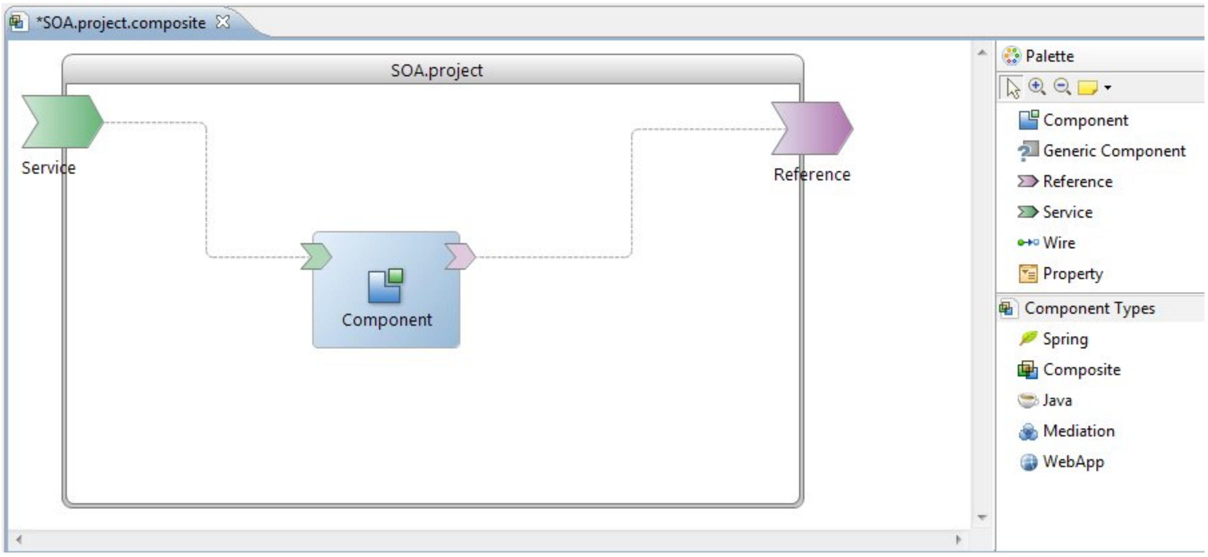
The constituents of a composite can run in a single process on a single computer or be distributed across multiple processes on multiple computers. A complete application might be constructed from just one composite or it could combine several different composites. The components making up each composite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

Composite



You edit composites in the TIBCO Business Studio Composite Editor shown in the following screen shot.

Composite Editor



Components

A *component* is the basic element of 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 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 the following implementation types in different products:

- Java
- Mediation
- Composite
- WebApp
- Spring
- C++
- Adapters

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 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 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:

- Virtualization
- REST
- SOAP
- JMS
- EJB

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.

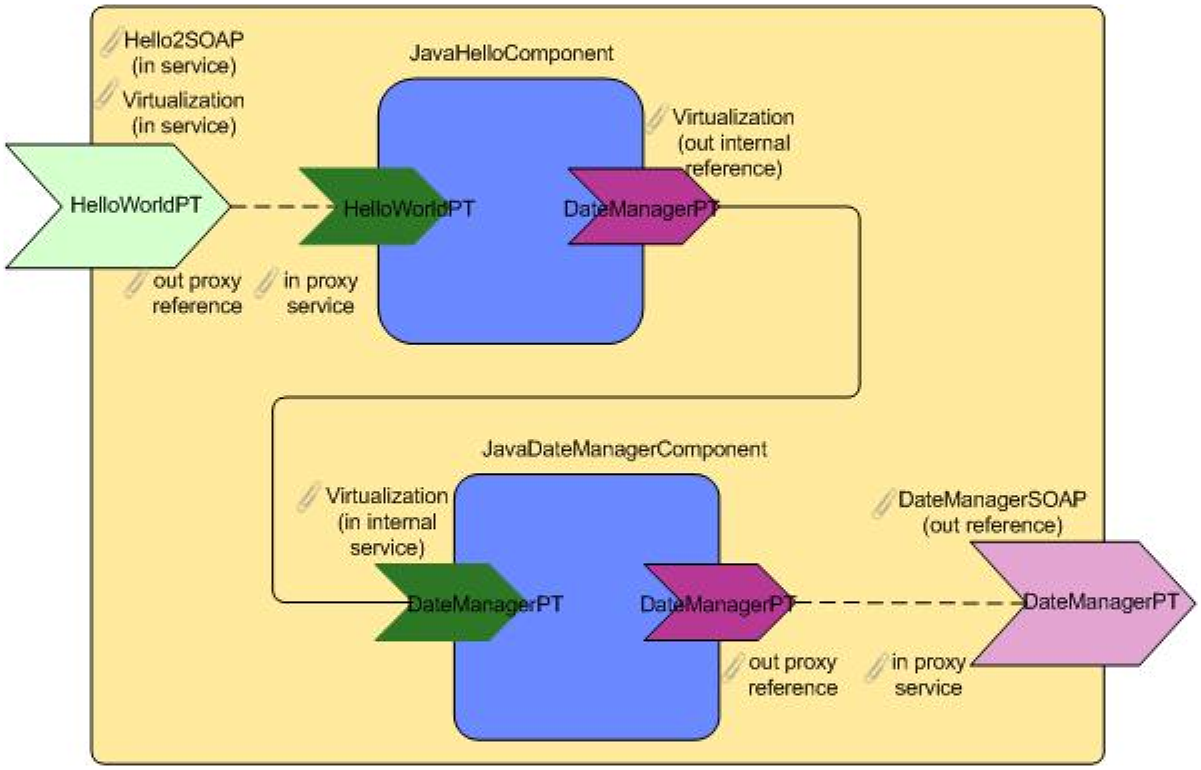
The following bindings are explicitly created by architects and developers only on promoted services and references:

- SOAP
- Adapters
- JMS Bindings

TIBCO Business Studio and TIBCO ActiveMatrix Administrator provide the option to choose between TIBCO's SOAP/JMS and W3C SOAP/JMS on SOAP binding type and a target service while adding a binding to a service.

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, any time 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.

Bindings



Properties

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

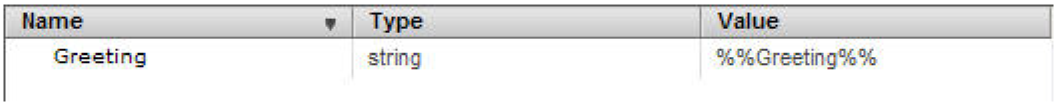
A property has a type, which may be either simple or complex. Implementations, components, composites, bindings, logging configurations and appenders, and resource templates can have properties. Implementation, component, and composite properties are defined in TIBCO Business Studio. Binding, logging 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 (component 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 following figure shows a property named Greeting bound to a substitution variable named Greeting.



Name	Type	Value
Greeting	string	%%Greeting%%

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

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 *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 upon 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 provide support for 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 so on.

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 provides 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

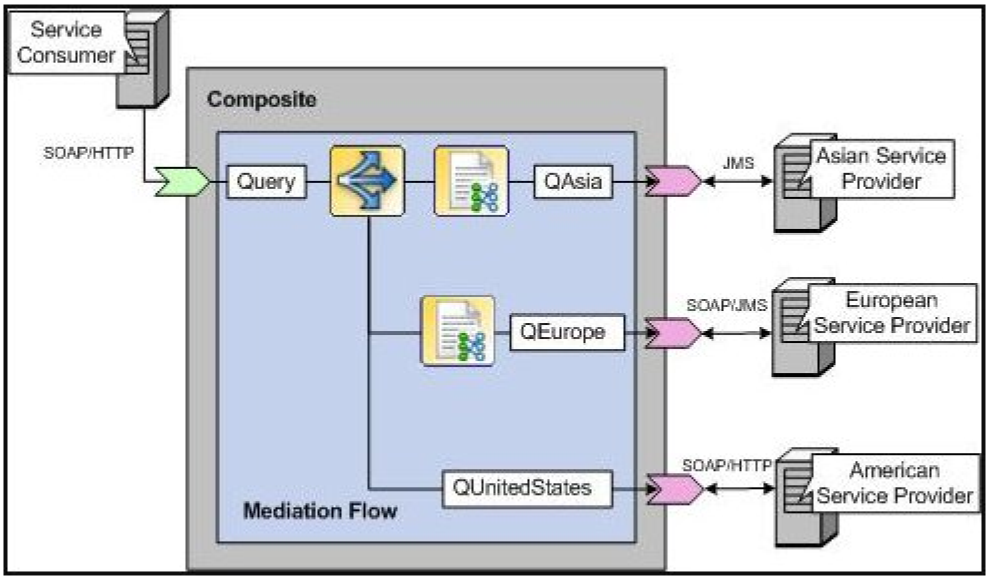
- Data enrichment
- Data validation
- Message filtering
- Log message and context data

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:

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:

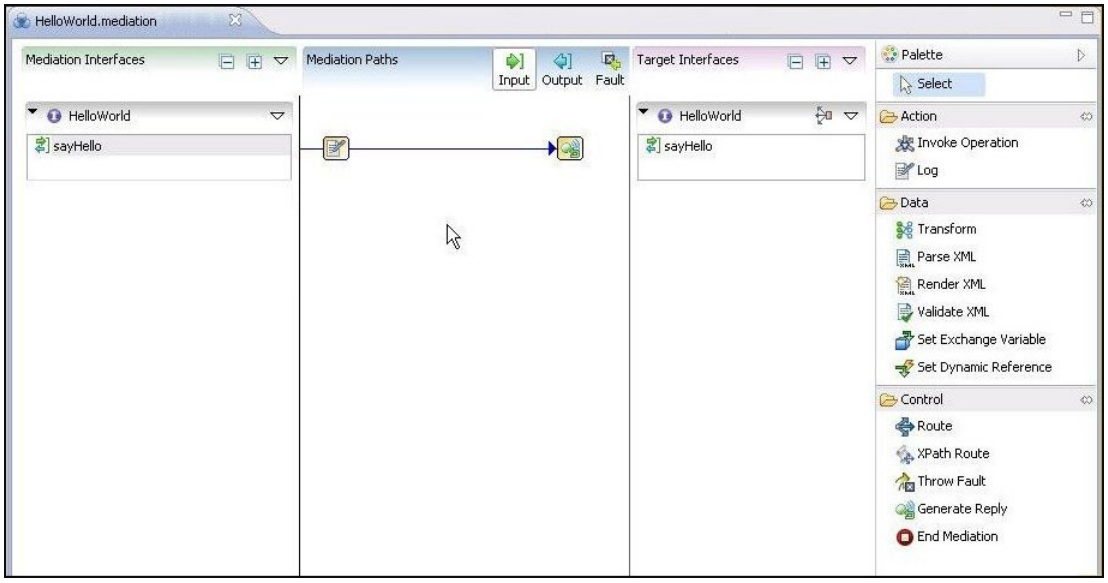
- **Service virtualization** A mediation service hides the location of service providers and details of how the services are provided (for example, the transport protocol, message format, and schema) from service consumers. Virtualization enables:
  – **Location transparency** The location of the service 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 flows 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 vice versa. 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 *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 *WebApp Component Development* .

Spring Components

Spring components integrate Spring Beans into the TIBCO ActiveMatrix platform.

A Spring component is very similar to 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 is associated with a Bean (as opposed to all being associated with the same Java class in the case of Java components).