TIBCO ActiveMatrix® Service Gateway

User's Guide

Software Release 1.1
May 2012

two-second advantage™

TIBCO®
The Power of Now®

Important Information

Contents

Preface

TIBCO ActiveMatrix® Service Gateway is a high-performance event-based service-request routing engine. It treats requests and responses as events and implements the logic of a gateway layer using simple event-condition-action rules. TIBCO ActiveMatrix Service Gateway provides the functionality of service abstraction, back-end integration, routing, request authentication and authorization, orchestration and throttling management with the implementation of policies at each layer of the gateway. To support these core functions, TIBCO ActiveMatrix Service Gateway offers the centralized logging and Partner & Service Management.

TIBCO ActiveMatrix Service Gateway architecture has been designed so that when multiple TIBCO ActiveMatrix Service Gateway servers are deployed, key management functions including throttle management, cache management and logging are co-ordinated across servers.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveMatrix Service Gateway Documentation

The following documents form the TIBCO ActiveMatrix Service Gateway documentation set:

- TIBCO ActiveMatrix Service Gateway *Installation* Read this manual for instructions on site preparation and installation.
- TIBCO ActiveMatrix Service Gateway *User's Guide* Read this manual for instructions on how to configure and use this product.
- TIBCO ActiveMatrix Service Gateway *Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

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

- TIBCO BusinessEvents®
- TIBCO ActiveSpaces®
- TIBCO Rendezvous®
- TIBCO Enterprise Message Service™
- TIBCO ActiveMatrix BusinessWorks™
- TIBCO ActiveMatrix® Spotfire®

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
ENV_NAME *TIBCO_HOME* ASG_*HOME*	TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments.
ASG_CONFIG_HOME	An installation environment consists of the following properties: • **Name** Identifies the installation environment. This name is referenced in documentation as *ENV_NAME*. On Microsoft Windows, the name is appended to the name of Windows services created by the installer and is a component of the path to the product shortcut in the Windows Start > All Programs menu. • **Path** The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*. TIBCO ActiveMatrix Service Gateway installs into a directory within a *TIBCO_HOME*. This directory is referenced in documentation as *ASG_HOME*. The default value of <ProductAcronym>*_HOME* depends on the operating system. For example on linux platform, the value of *ASG_HOME* is /home/asg/tibcoasg/asg/1.1. TIBCO ActiveMatrix Service Gateway stores the configuration files in a directory which is seperate from the installation directory. This directory is referenced in documentation as *ASG_CONFIG_HOME.* For example on linux platform, the value of *ASG_CONFIG_HOME* is: /home/asg/tibcoasgconfig/tibco/cfgmgmt
`code font`	Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example: Use `MyCommand` to start the foo process.

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
`bold code font`	Bold code font is used in the following ways: • In procedures, to indicate what a user types. For example: Type **`admin`**. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, `MyCommand` is enabled: `MyCommand [enable | disable]`
italic font	Italic font is used in the following ways: • To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*. • To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName*
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
`[ ]`	An optional item in a command or code syntax. For example: `MyCommand [optional_parameter] required_parameter`
`|`	A logical `OR` that separates multiple items of which only one may be chosen. For example, you can select only one of the following parameters: `MyCommand para1 | param2 | param3`

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

Connecting with TIBCO Resources

How to Join TIBCOmmunity

TIBCOmmunity 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. TIBCOmmunity offers forums, blogs, and access to a variety of resources. To register, go to http://www.tibcommunity.com.

How to Access TIBCO Documentation

You can access TIBCO documentation here:

http://docs.tibco.com

How to Contact TIBCO Support

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

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

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

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

  https://support.tibco.com

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

Chapter 1 Introduction to TIBCO ActiveMatrix Service Gateway

This chapter provides an overview of TIBCO ActiveMatrix Service Gateway software, and a brief description of the major functional components.

Topics

Product Overview

TIBCO ActiveMatrix Service Gateway provides the routing between partners and the service providers operating at the back-end. TIBCO ActiveMatrix Service Gateway is an event-based routing engine which processes requests and responses at a high performance level. TIBCO ActiveMatrix Service Gateway exposes these services, wherever they may be located, through a single logical point of ingress so it can not only route service requests, but also protect itself and services from unexpected or undesirable demand.

TIBCO ActiveMatrix Service Gateway provides the following key features:

- Receives, routes and forwards requests at high speed.
- Routes requests between any requestor and any service endpoint.
- Protects from over-usage or access from unauthorized partners.
- Protects service endpoints from over-usage or unauthorized usage.
- Reports operation activity such as logging, performance and fault monitoring.
- Assures service level agreements are met.

TIBCO ActiveMatrix Service Gateway provides an event-driven webservices platform responsible for routing, co-ordinating and managing the partner and customer API requests to the various services exposed by an organizations internal services layer.

TIBCO ActiveMatrix Service Gateway supports the following policy models:

- Authorization rules

  You can configure authorization policies for partners that determine whose requests are handled.

- Throttling models

  You can configure various types of throttles that determine when the requests are handled.

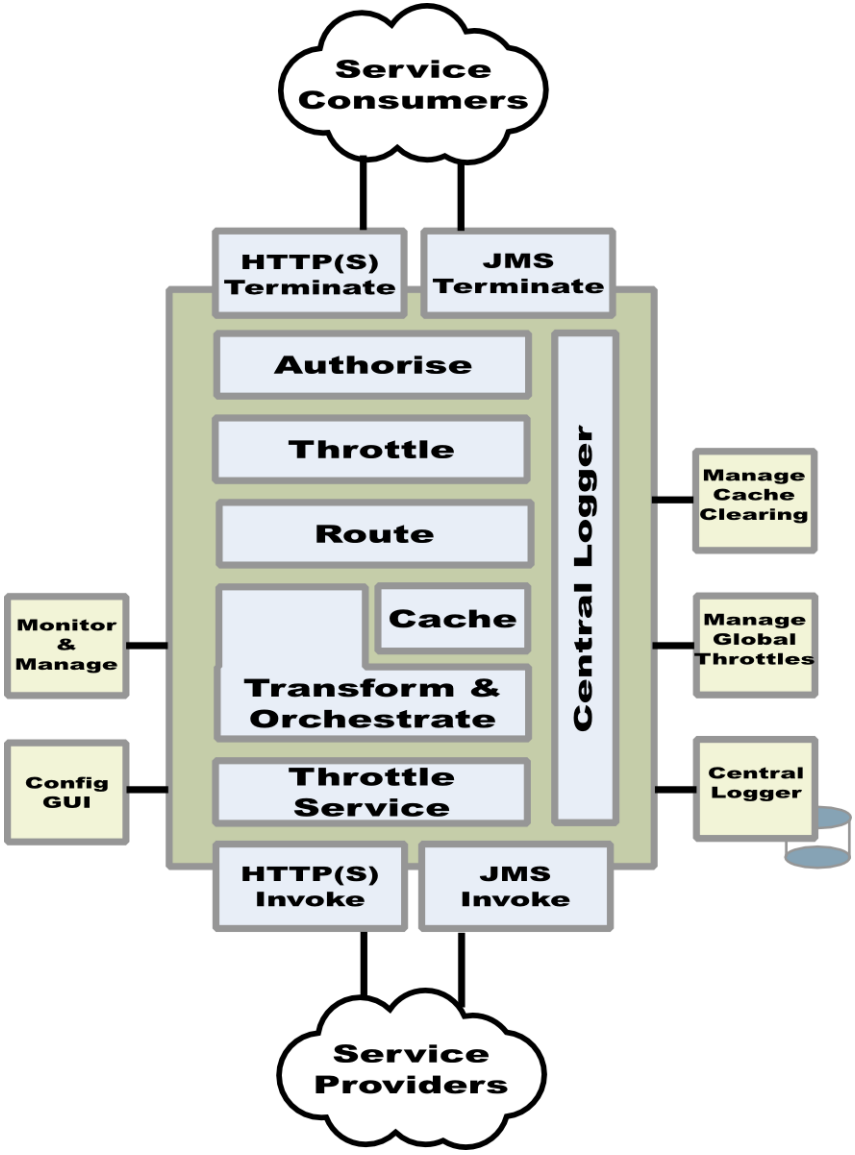
- Routing rules

  You can configure the routing rules that determine where the requests are handled.

- Light-weight orchestration models determine how requests are handled.

  The following diagram shows the overview of the product functionality:

Figure 1 Functional Overview



Functional Components

This section provides a general overview of each product component used by TIBCO ActiveMatrix Service Gateway software.

Design time components

TIBCO ActiveMatrix Service Gateway provides the following design time components:

TIBCO ActiveMatrix Service Gateway Configuration GUI

Partner data, partner operations, partner groups, services, operations, mappings, throttles, errormaps, schemas and routing information required for the various functionality of the software are configured via a web-based GUI.

TIBCO ActiveMatrix Service Gateway Gateway Studio

The Gateway Studio is a design time environment which allows you to design and develop the custom extensions. The custom extensions can be integrated with the default implementation to customize the default behavior of the gateway core engine.

Run time components

TIBCO ActiveMatrix Service Gateway provides the following runtime components:

Gateway Operational Layer

The Gateway Operational layer consists of following components:

Module for Apache HTTP Server

The Apache layer is used to terminate the incoming HTTP(s) transport and communicate with Facade component to forward the requests for further processing. Optionally, a JMS Server can be deployed to use the JMS transport.

Core Engine

The core engine is a high-performance event-based service-request routing engine that receives requests as events and uses the rules engine to determine where requests are handled.

The core engine contains the following main sub-components:

- Facade: The Facade provides a public northbound interface for the gateway to receive requests for a given API with a given binding (for example, SOAP over HTTP or SOAP over JMS).
- Router: The Router receives the requests from the Facade and routes it to the appropriate service handler.

Cache Agent

The cache agent stores the cache data for all objects of the cluster.

Gateway Management Layer

The Gateway Management layer provides the activity logging and request tracking. It has following sub-components.

Central Logger

Central Logger provides the centralized logging of messages in a database or file.

Global Throttle Manager

The Global Throttle Manager manages the Façade Throttle Manager and Service Throttle Manager. This component maintains the state of all global throttles in both Façades (Façade Throttles) and Routers (Service Throttles).

Cache Clearing Manager

The Cache Clearing Manager component clears the cache based on the size and age of the cached values.

Monitoring and Management Server

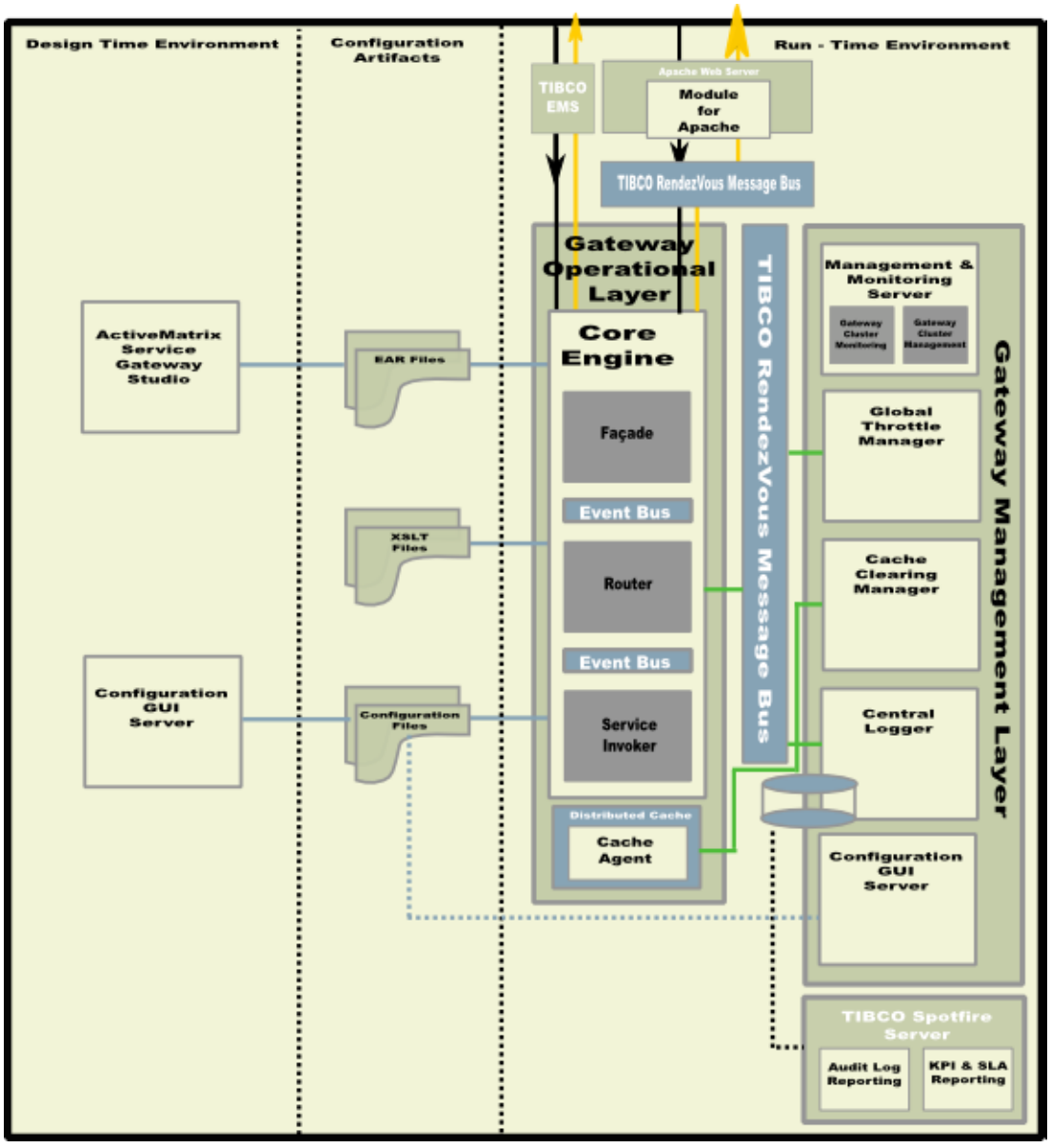
The Monitoring and Management Server is the central management component that allows you to monitor the status and manage the operational tasks of all components in the Gateway cluster.

Gateway Reporting (Optional)

The Gateway Reporting component generates the various type of reports based on the data logged in by the Central Logger component. This component integrates with the TIBCO Spotfire product to display the metrics.

The primary software components are shown in the following diagram:

Figure 2 Functional Components



Deployment Architecture

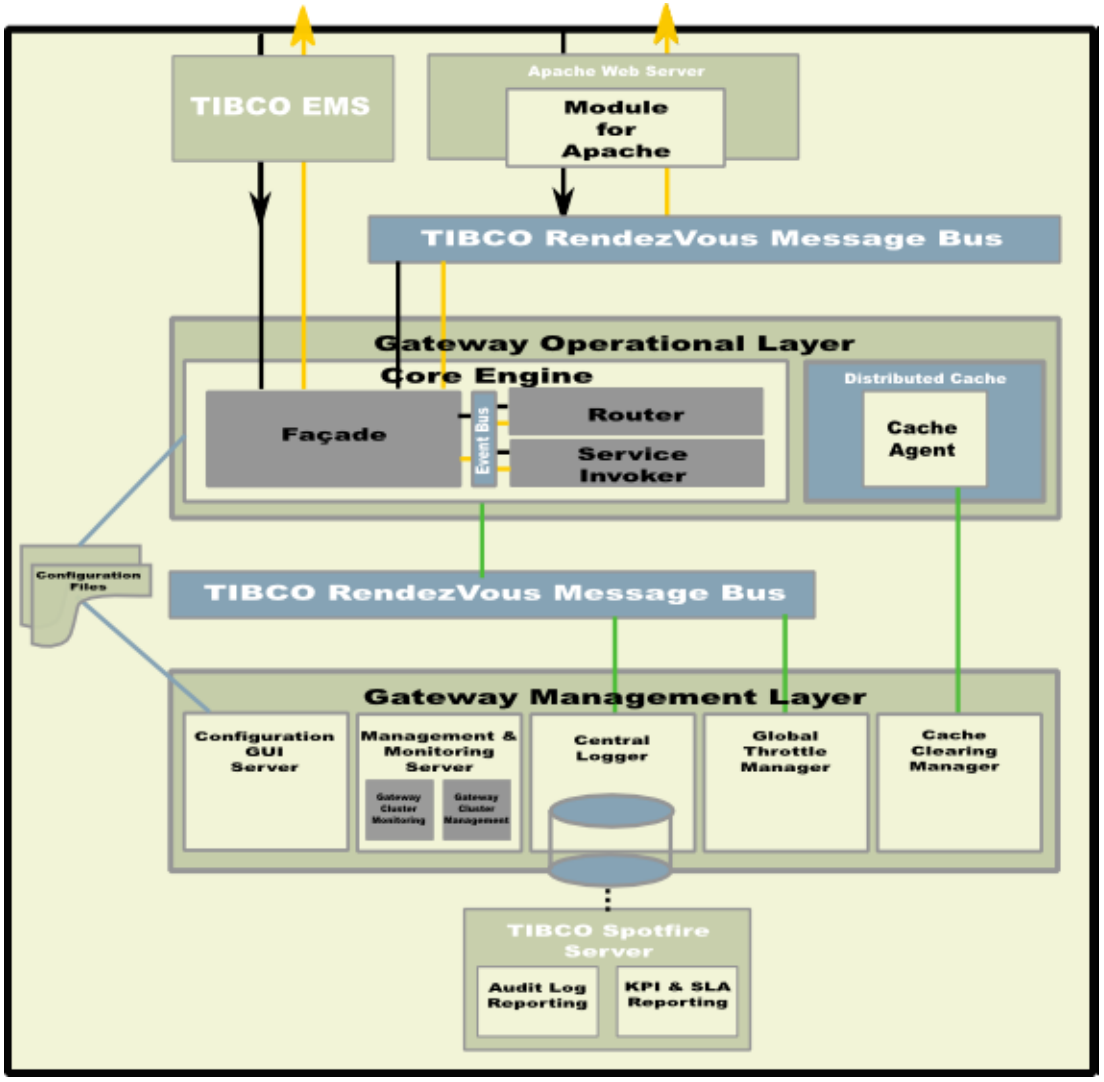
TIBCO ActiveMatrix Service Gateway is deployed as a cluster of engines that together act as a single logical gateway. The engines in the cluster can run on a single server or in a distributed environment across multiple physical or virtual servers, providing fine grained control over the cluster deployment topology.

TIBCO Rendezvous is used for the communication between most of the run-time components, both in the gateway operational layer itself and between the components from the gateway management layer and the gateway operational layer and all components share a single set of configuration files which need to be stored on shared storage device that's accessible for each of the run-time components. In case the multiple instances of core engine are deployed in a cluster, also multiple cache-agents are instantiated providing a single distributed cache that is shared across all core engines to support association and response cache functionality.

Single Server Deployment Architecture

In its simplest non-high available form, TIBCO ActiveMatrix Service Gateway can run as a single server. The deployment of the software components as a single server instance are shown below:

Figure 3 Single Server Deployment



This configuration provides full gateway functionality, including the optional operational reporting & analytics provided by the TIBCO Spotfire Server components. The TIBCO Spotfire Professional client software running on a Windows workstation and the TIBCO Spotfire WebPlayer Server software running on a Windows server are not shown in the diagram.

The protocol termination components, Apache HTTP server for HTTP and HTTPS transports and optionally the TIBCO Enterprise Messaging Service for JMS transport, need to be deployed and managed with their standard operations management tools.

The Module for Apache HTTP Server that is part of the TIBCO ActiveMatrix Service Gateway is deployed as a normal module for the Apache HTTP server. This module turns the Apache http requests into TIBCO Rendezvous messages for the communications with the gateway core engine. In case the Apache HTTP Server is deployed within a DMZ zone, the TIBCO Rendezvous Routing Deamon can be configured to forward the TIBCO Rendezvous messages from the DMZ network through the firewall to the internal network in which the TIBCO ActiveMatrix Service Gateway components are deployed.

Run Time Components

The run-time components of the TIBCO ActiveMatrix Service Gateway are deployed as a single application that can span multiple host servers. The run-time components are as follows:

- Gateway Core Engine
- Central Logger
- Global Throttle Manager
- Cache Clearing Manager

The management layer components, Central Logger and Global Throttle Manager, communicate at run-time with the gateway core engine using Rendezvous messages.

The Central Logger component receives logging events as the messages from the Rendezvous bus and stores them in the central logger database at appropriate intervals; this reduces the disk load during high transaction rates. These logging events are published by the gateway core engines during their operation. The Global Throttle Manager reports the throttle usage data to the Central Logger via the same mechanism. Network logging reduces the operating system load of the gateway core engines, as they are not logging to a local disk. It provides the mechanism to centralize the logging, as it is very important to provide a single, consistent view of activities. Finally, logging ensures that all components are not limited in their performance by the file system.

The Global Throttle Manager controls the throttle allocation for each gateway core engine. The Global Throttle Manager receives throttle reports from the gateway core engines over the Rendezvous bus. It, also, sends the throttle grants back to the gateway core engines over the Rendezvous bus.

The Global Throttle Manager treats the throttle usage events as the heartbeat interval of a core engine. In absence of, a configurable, number of consecutive heartbeats, the Global Throttle Manager treats the gateway core engine as dead and distributes the throttle limits of the dead instance equally to all the alive core engines.

The Cache Clearing Manager does not use the Rendezvous bus to interact with the gateway core engine. It connects directly to one of the Cache agents to clear the cache so that it does not grow to an excess size. This cleanup of the cache is called cache flushing.

To deploy and start the cluster of run-time components, you can use any of the following methods:

- Command Line:

  At the command-line, you specify the component unit to start and optionally a custom CDD file to use.

- TIBCO ActiveMatrix Service Gateway Monitoring and Management Server:

  The run time components can be deployed and started using the Management and Monitoring Server. This is the recommended method.

It's recommended to use only one method for the entire gateway cluster you are deploying. Using the Management and Monitoring Server to deploy the runtime components is recommended method.

Both of the deployment methods use two default resources: an EAR file and a cluster deployment descriptor (CDD), which is an XML file.

When the gateway core engine (with or without caching agent), global throttle manager or cache clearing manager are started, they use the `asg_core.ear` and `asg_core.cdd` files in the *ASG_HOME*`/bin` directory.

When the central logger component is started, it uses `asg_cl.ear` and `asg_cl.cdd` files in the *ASG_HOME*`/bin` directory.

The Monitoring and Management Server and the GUI Configuration server can be started at the command line.

Any configuration updates that are made through the GUI Configuration server is persisted in the configuration files on the shared storage device. These configuration files needs to be reloaded by the run-time components in order to be effectuated.

The optional operational reporting & analytics provided by the TIBCO Spotfire server components interacts with the Central Logger through the central logger database using a standard JDBC connection.

Distributed Deployment Architecture

TIBCO ActiveMatrix Service Gateway supports a distributed deployment environment, in which multiple instances of the gateway engines can be deployed. This architecture meets the requirements of high availability and scalability of the gateway components, which is recommended in a production environment.

Scaling and High Availability

TIBCO ActiveMatrix Service Gateway provides a default site topology file which is configured for a deployment with single instances for each engine of the gateway cluster, all being deployed on a single server host. This configuration allows you to quickly deploy the TIBCO ActiveMatrix Service Gateway in a development environment, though it typically does not meet availability and scalability requirements for a production deployment.

TIBCO ActiveMatrix Service Gateway Studio can be used to create production site topology configurations for your production environment including load balanced and fault-tolerant setups.

Load Balancing

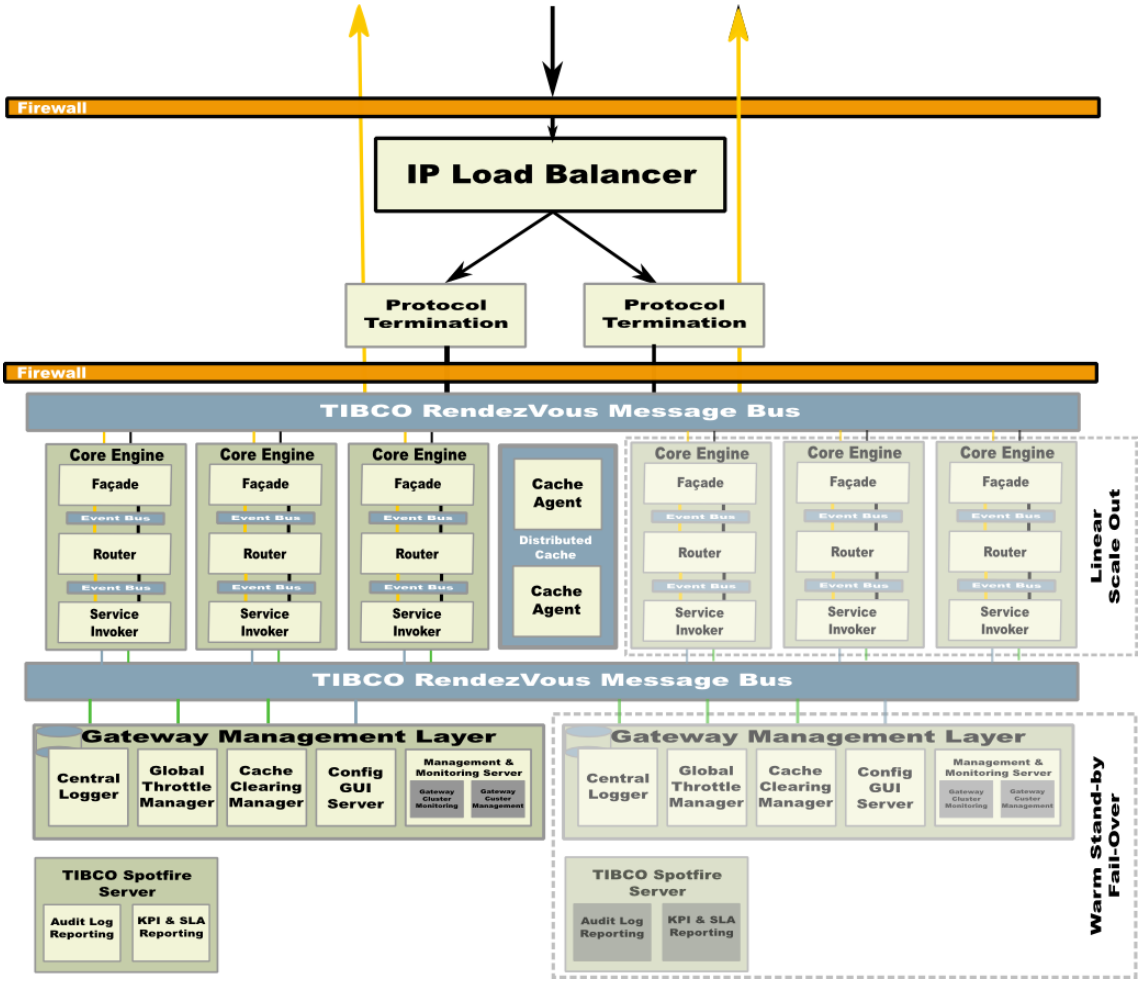
TIBCO ActiveMatrix Service Gateway can be rapidly scaled up and down through the addition or removal of additional instances of the core engines to the gateway cluster.

The TIBCO ActiveMatrix Service Gateway architecture has been designed so that when multiple core engine instances are deployed in a gateway cluster, the key management functions including throttle management, cache management, cache clearing management and central logging are co-ordinated across all core engine instances. As these management functions are performed out-of-band, the components that provide these functions do not need to be scaled in support of higher transaction volumes.

However, as transaction levels increase, it is likely that this will be accompanied by a corresponding increase in management activity. To avoid the possible impact of management activity upon the core engines of the TIBCO ActiveMatrix Service Gateway, these management components and the TIBCO Spotfire Servers should be moved onto separate servers.

Following diagram illustrates a simplified view of the scaled solution depicting the deployment of various components in a distributed environment:

Figure 4 Deployment of Multiple Components in Distributed Environment



Increasing the number of the core engines in TIBCO ActiveMatrix Service Gateway deployment provides a near linear increase in the maximum number of transactions that can be managed and reduces the impact of the failure of an individual core engine. TIBCO ActiveMatrix Service Gateway uses a `shared nothing` model between the active core engines to ensure that there is no shared state.

To support a load-balanced setup, the transport protocol termination components have to be configured appropriately.

For the JMS endpoints, load balancing requests across multiple core engines is achieved by setting up non-exclusive queues in the JMS server which automatically balances the load of incoming messages across the JMS receivers of the core engine instances. For HTTP and HTTPS endpoints, load balancing the requests across multiple core engines is handled by the TIBCO ActiveMatrix Service Gateway Module for Apache HTTP Server. If a comma-separated list of TIBCO Rendezvous subjects is configured for an Apache server location, the module for Apache HTTP server load balances the incoming requests across the list of Rendezvous subjects. For each deployed core engine, a different Rendezvous subject from the list needs to be configured to ensure that requests are only handled once by a single core engine.

When the protocol termination components themselves reach the limits of the scale they can provide, an IP load-balancer can be added to the deployment in front of multiple Apache HTTP servers or JMS servers. The load-balancer should be configured to make these all available on a single IP address.

High Availability of TIBCO ActiveMatrix Service Gateway

For a high available setup of the TIBCO ActiveMatrix Service Gateway deployment, a different approach is taken for the components in the Gateway Operational Layer and the components in the Gateway Management Layer.

Gateway Operational Layer

As the core engine and Apache HTTP server maintain no state, fault tolerance is provided by having multiple instances running across sites and the host servers with the same configuration supporting a load balanced configuration. See Load Balancing, page 13.

A fault tolerant setup for JMS endpoints of the TIBCO ActiveMatrix Service Gateway is leveraging the fault-tolerant setup capabilities of the TIBCO Enterprise Message Service. See *TIBCO Enterprise Message Service™ User's Guide* for details.

Fault tolerance of cache agents is handled transparently by the object management layer. For fault tolerance of cache data, the only configuration task is to define the number of backups you want to keep, and to provide sufficient storage capacity. Use of a backing store is not needed as the cache agents are only used to implement the association cache, which is automatically rebuilt after complete failure as new transactions are handled by the TIBCO ActiveMatrix Service Gateway.

Gateway Management Layer

The components of the Gateway Management Layer should be deployed once in a primary-secondary group configuration. The central logger and the global throttle manager need to have a single running instance at all times to ensure that the gateway core engine operates without loss of functionality.

Therefore the central logger and global throttle manager need to be deployed in fault tolerant configuration with one active engine and one or more standby agents on a separate host servers. Such fault-tolerant engine setup can be configured in the cluster deployment descriptor (CDD) file by specifying the maximum number of one active agent for either of the agent classes and by creating multiple processing unit configurations for both the global throttle manager and the central logger agent. Deployed standby agents maintain a passive Rete network. They do not listen to events from channels and they do not update working memory. They will take over from the active instance in case it fails.

The other components of the Gateway Management Layer have no direct impact on the functionality of an operating ActiveMatrix Service Gateway instance and hence they can be deployed with a cold standy configuration. This applies to the following components:

- Cache Clearing Manager
- Configuration GUI Server
- Monitoring and Management Server

It is suggested that multiple versions of these components are deployed across host servers with one running. If the running instance goes down start one of the other instances to regain full gateway functionality.

Chapter 2 Getting Started

This chapter provides an overview of the examples provided with the product.

Topics

Overview

TIBCO ActiveMatrix Service Gateway provides the following examples:

- GetLocation
- BookQuery
- Caching

Prerequisites

This section describes the list of tasks to be completed before you can run the examples.

Installation

Finish the installation of TIBCO ActiveMatrix Service Gateway software and post installation tasks, as described, in the *TIBCO ActiveMatrix Service Gateway Installation Guide*.

Verify the TIBCO ActiveMatrix Service Gateway server status

Verify the installation and health of TIBCO ActiveMatrix Service Gateway server by following the steps listed in **`Checking TIBCO ActiveMatrix Service Gateway server status`** section of the *TIBCO ActiveMatrix Service Gateway Installation Guide.*

Examples

This section describes the examples shipped with the TIBCO ActiveMatrix Service Gateway software.

GetLocation

The GetLocation service demonstrates the routing capability of TIBCO ActiveMatrix Service Gateway. The gateway interacts with a mock-up of a backend service which returns the coordinates of a device associated with a phone number. The gateway routes the request to a different backend service depending on the input phone number in the request.

This TIBCO ActiveMatrix Service Gateway example is shipped with a **`GetLocation`** project which contains the client operations and back end services to be executed.

Refer to the `ASG_HOME/examples/GetLocation/readme.html` for the instructions on how to run the example.

BookQuery

The BookQuery service queries all the books in a book store by different criteria such as query by author, isbn, publisher, and title. TIBCO ActiveMatrix Service Gateway example project implements the following four policies:

- Rate Throttles
- Quota Throttles
- High Water Mark Throttles
- Error Throttles

Refer to the `ASG_HOME/examples/BookQuery/readme.html` for the instructions on how to run the example.

Caching

The Caching example demonstrates the caching functionality provided by TIBCO ActiveMatrix Service Gateway.

It has following main components:

- ASG_CBA is a TIBCO BusinessWorks project simulates a east side service, which takes the `firstIdentity` field as a cross-reference key and translates its value. This project does a simple translation on the key using an XML file. Alternatively, the service can be an adapter call, a lookup in a database, and other webservice.
- A SOAP service which is hosted on the TIBCO ActiveMatrix Service Gateway. This service type is NOOP which means that it just returns an XML document with a received key and a translated value.

After the translated value is received from the service implemented in ASG_CBA BusinessWorks project, this is plugged into a SOAP payload. This payload is used to call a SOAP service running on the TIBCO ActiveMatrix Service Gateway.

When the **`CustomStage`** feature is applied to an operation configuration, it enables a set of rules in the **`ASG_DefaultImplementation`** project. When the operation request is invoked, it looks into the cache for a cross-reference before the routing step. It uses the value of **`firstIdentity`** as a key for the cross-reference. If a value for that key is present in the cache, it is used in the cross reference. If a value is not found, then it sends a TIBCO Rendezvous message to retrieve that value from the service in the ASG_CBA BusinessWorks project.

Refer to the `ASG_HOME/examples/Caching/readme.html` for the instructions on how to run the example.

Configuring an endpoint operation for ActiveMatrix Service Gateway

This section explains the steps to configure an service operation on the TIBCO ActiveMatrix Service Gateway platform. This includes:

Create a new configuration

1. Launch the GUI. See Starting the GUI on page 27 for details.
2. Click **`Add New Configuration`** icon.
3. Configuration Name: Type a configuration name (For example, **`ASG_Get_Start`**)
4. Click **`Save`** button.

An error is displayed indicating that there are errors in the configuration. This is an empty configuration with no data defined, so the error. Adding the data to the configuration is explained in the following sections. Click OK to save this.

Configure the Partner Group

1. Select the **`ASG_Get_Start`** configuration.
2. Click on **`Partner Groups`** tab
3. Click **`Add property (+)`** icon.
4. Type the values for the fields as shown in the following table:

Table 3 Partner Group Configuration:.

Parameter	Value
Group Name	supportASG

Parameter	Value
Email	support@tibco.com
Phone	0019202331999

5. Click the Apply button.

Configure the Partner Data

1. Select the **ASG_Get_Start** configuration, if not already selected.
2. Click on **Partner Data** tab.
3. Click **Add property (+)** icon.
4. Type the values for the fields as shown in the following table

Table 4 Partner Data Configuration:.

Parameter	Value
Partner Name	tibcoASG
Partner Email	support@tibco.com
Partner Phone	0019202331999
Partner Group	supportASG
Partner Serial Number	anonymous
Partner Issuer CA	O=TIBCO;CN=ASG;CN=RV;CN=Anonymous

5. Click the Apply button.

Configure a gateway endpoint operation (Operation)

1. Select the **ASG_Get_Start** configuration, if not already selected.
2. Click on **Operations** tab.
3. Click **Add property (+)** icon.
4. Type the values for the fields as shown in the following table:

Table 5 Operation Configuration:

Parameter	Value
Operation Name	queryBookByAuthorBW
SOAP Action	"/GetBooksByAuthorEndpoint"
Operation URI	/ServerProcesses/GetBooksByAuthorEndpoint
Operation Service Name	MWC

5. Click the Apply button.

Configure a gateway reference (Service)

1. Ensure that the **`ASG_Get_Start`** configuration is selected.
2. Click on **`Services`** tab.
3. Click **`Add property (+)`** icon.
4. Type the values for the fields as shown in the following table:

Table 6 Service Configuration:

Parameter	Value
Service Name	http.GetBooksByAuthor
Type	HTTP (select from the drop down list box)
Timeout	30000
SOAP Action	"/GetBooksByAuthorEndpoint"
URI	/ServerProcesses/GetBooksByAuthorEndpoint
Host	127.0.0.1
Port	9696

5. Click the Apply button.

Configure an authorization configuration

1. Ensure that the **ASG_Get_Start** configuration is selected.
2. Click on **PartnerOperations** tab.
3. Click **Add property (+)** icon.
4. Type in the following values:

Table 7 Partner Authorization Configuration::

Parameter	Value
Partner	tibcoASG
Partner Operation	queryBookByAuthorBW
Partner Timeout	5000

5. Click the Apply button.

Save the gateway configuration

On the menu bar, click the **Save Configuration** icon to save the **ASG_Get_Start** configuration.

Test the gateway configuration

Run Apache HTTP Server, if not running

See Setup and Run Apache HTTP Server, page 119 for details.

Run TIBCO ActiveMatrix Service Gateway Core Server

1. Navigate to the TIBCO ActiveMatrix Service Gateway installation as below:

   `cd` *ASG_HOME*`/bin`
2. Start the gateway engine components as below:

   ```
   ./asg-engine -u asg-caching-core -a ASG_Get_Start
   ```
3. Verify that the gateway engines started successfully and there are no errors.
4. Verify that the configuration for operation, services and partner are loaded correctly in the **asg_engine** engine logs.

Test the configured operation and service

1. Launch TIBCO Designer and open the project:
   ASG_HOME/examples/BookQuery/BookQuery
2. Run the following server processes:
   — BooksInterface-service1
3. Run the following client process:
   — QueryByAuthorClient
4. Verify that the process runs successfully without any errors.

Chapter 3 Gateway Configuration User Interface

This chapter describes the graphical user interface that allows you to add a new gateway configuration with partners data, operations, services, mapping between request and response messages, throttles, schemas and other parameters required for various functionality of TIBCO ActiveMatrix Service Gateway.

Topics

Overview

The configuration GUI allows you to manage a configuration for the TIBCO ActiveMatrix Service Gateway. A gateway configuration contains all the information related to the partners, partner groups, operations, services, mappings, schemas, throttles, routing which is required by the runtime gateway engine.

Starting the GUI

This section explains the steps to start the graphical user interface in a test environment.

1. Open a terminal window.
2. Navigate to *ASG_HOME*/bin directory.
3. Type the following command:

   ```
   ./asg-configui
   ```

4. Open a browser window and enter the following URL:

   ```
   http://localhost:9200/ASGConfig/
   ```

5. Enter the login information as below:

   username:asgadmin
   password:asgadmin

6. Verify that the user logs on successfully on user interface.
7. Verify that the following default configurations are displayed:
   - default
   - GetLocation
   - BookQuery
   - BookQueryWSS
   - Caching

Changing Login, Host and Port Information

The values for username, password, host and port are configured in the *ASG_HOME*/**bin/asg-configui.tra** file. By default, the values are shown as below:

```
tibco.env.ASG_HOST=localhost
tibco.env.ASG_PORT=9200
tibco.env.ASG_ADMIN_USERNAME=asgadmin
tibco.env.ASG_ADMIN_PASSWORD=asgadmin
```

To change the values for username, password, host and port, follow the steps:

1. Open a terminal window.
2. Navigate to *ASG_HOME*/bin directory.
3. Make a copy of the **asg-configui.tra** file.

4. Edit the **asg-configui.tra** file using a text editor to set the following properties, as required:

   tibco.env.ASG_HOST=type your hostname

   tibco.env.ASG_PORT=type your port

   tibco.env.ASG_ADMIN_USERNAME=type your username

   tibco.env.ASG_ADMIN_PASSWORD=type your password

5. Stop the GUI launcher (if running already).
6. Type the following command to start the GUI launcher:

   ```
   ./asg-configui
   ```

For production systems, it is recommended to deploy the **ASGConfig.war** file on a secure application server running in the production environment. The war file is located under *ASG_HOME*/**webapp** directory.

Manage Configuration

The GUI allows you to add, update, delete and duplicate a configuration. A configuration is a folder that contains all the information related to the partner data, partner operations, operations, services, mappings, routing etc.

Table 8 Manage Configuration:

Action	Description
Add a new configuration	Allows you to add a new configuration and add the configuration data.
Update an existing configuration	Allows you to update an existing configuration data.
Delete an existing configuration	Allows you to delete an existing configuration.
Duplicate an existing configuration	Allows you to duplicate an existing configuration.

Operations

An Operation is defined as a single type of request sent to the gateway engine.

The Operations tab allows you to add the list of operations supported by theTIBCO ActiveMatrix Service Gateway.

Add a new operation

To add a new operation, follow the steps:

1. Click on **`Operations`** tab.
2. Click + (Add property) icon to create a new operation.
3. Enter the details for operation, as defined below:

Table 9 Operation Configuration

Parameter	Description
Operation Name	Logical operation name.
SOAP Action	SOAP Action for this operation. This field is used to identify which operation an incoming request is applied to. Must be unique.
Operation URI	URI for this operation.
Operation Service Name	Logical service name (used for routing).
New ProcessBody XSLT	New XSLT transformation sheet file containing the rules to parse and validate the message. Optional.
Existing ProcessBody XSLT	Existing XSLT transformation sheet file containing the rules to parse and validate the message. Optional.
New FaultReport XSLT	New XSLT transformation used to produce fault message for provided fault data.

Table 9 Operation Configuration

Parameter	Description
Existing FaultReport XSLT	Existing XSLT transformation used to produce fault message for provided fault data. Optional.
Forward Mapping	Reference to forward northbound mapping. This mapping transforms from requestor API to canonical request format. If no mapping reference needed, then enter **id**. Required.
Reverse Mapping	Reference to reverse northbound mapping. This mapping transforms from canonical request format to requestor API. If no mapping reference needed, then enter **id**. Required.
Operation Method	HTTP method used to separate REST requests which are made on same URI but with different operations. Optional.
Operation Features	List of keywords identifying features required by the operation. Supported features are: — Validation – XSD validate northbound request and response. Optional.

4. Click **`Apply`** button.
5. Click **`Save Configuration`** button.

Delete an existing operation

To delete an existing operation, follow the steps:

1. Click the red cross icon located at the top left corner of **`Operation Name`** field.
2. Click **`Apply`** button.
3. Click **`Save Configuration`** button.

Services

A Service is defined as a single type of request that the gateway instance sends to backend systems. The backed-API defines the request structure, and the expected reply structure returned.

The Services tab allows you to configure the southbound services and related details like transport information supported by the backend systems and applications.

TIBCO ActiveMatrix Service Gateway supports the following types of services:

- ESB
- HTTP
- SOAPJMS

Add a new service

To add a new service, follow the steps:

1. Click on **`Services`** tab.
2. Click + (Add property) icon to create a new service.
3. Enter the details for service, as defined below:

Table 10 Services Configuration

Parameter	Description
Type: No Operation Empty backend. Does accept any messages. Used for operations that make use of the information retrieved during customer validation.	
Service Name	Name of the backend service.
Type	The type of the transport to use when invoking the backend service. For example: No Operation.
Service Group	Name of the service group.
Timeout	Timeout in milliseconds to use when invoking the backend service.

Table 10 Services Configuration

Parameter	Description
Forward Mapping	Mapping from request canonical form to backend service API. Mapping details are defined in **Mapping** tab.
Reverse Mapping	Mapping from backend service API to response canonical form. Mapping details are defined in **Mapping** tab.
Throttle Chain	Throttle chain to be applied when invoking the backend service. You can add one or more throttle names. The details of the throttles are defined in the **Throttles** tab.
Type: ESB When the JMS transport is used to invoke the backend service, configure the following parameters specific to JMS transport	
ESB Channel	Number of predefined ESB channels.
ESB Service	Name of the ESB service to call.
Service Instance	Identity of service instance to call
ESB Operation	Called operation. Note that this tuple determines SOAP Action used by ESB as: "/esb/service//operation"
Type: HTTP When the HTTP transport is used to invoke the backend service, configure the following parameters specific to HTTP transport	
SOAP Action	The value of the SOAP Action as defined by the service WSDL.
URI	The URI to use when invoking the backend service.
Host	The IP address or hostname of the service implementation when invoked over HTTP.
Port	The TCP port of the service implementation when invoked over HTTP.

Table 10 Services Configuration

Parameter	Description
Username	Username with BASIC authentication.
Password	Password with BASIC authentication.
Headers To Forward	This field allows the users to copy headers information from the northbound incoming request and forward it to the southbound side. If the value of Headers To Forward is specified as "*", then all the headers are copied. If the Headers To Forward contains "*,-SoapAction" then any incoming SOAP Action header is removed from the incoming headers and the value set on the endpoint is ignored.
Type: SOAPJMS When the SOAPJMS transport is used to invoke the backend service, configure the following parameters specific to SOAPJMS transport	
ESB Channel	Number of predefined ESB channels.
SOAP Action	The value of the SOAP Action as defined by the service WSDL.
JMS Priority	
JMS Expiration	
Destination Name	Name of the destination on JMS server.
Destination Type	Type of the destination (TOPIC/QUEUE).
Target Service	Name of the service to call.
Content Type	

Routing

The Routing tab allows you to configure the routing information for the gateway engine.

The routing provides binding between northbound operation and southbound service.

4. Start the GUI, if not already started. See Starting the GUI, page 27 for details.
5. Click the **Routing** tab
6. Enter the following parameters, as defined below

Table 11 Routing Configuration

Parameter	Description
Operation Name	Name of the northbound operation. This is defined in the **Operations** tab.
Service Version	Version number of the back-end service.
Service Name	Name of the southbound service reference. This is defined in the **Services** tab.
Secondary ACL Keys	ACLs to route to this service.
Routing Key	Evaluated routing key for the given operation.

Partner Groups

The Partner Groups tab allows you to configure the information for a partner group and the throttle chain which is applied to any requests sent by a partner who belongs to this group.

Add a throttle for a partner group

To add a new partner group, follow the steps:

1. Click on **`Partner Groups`** tab.
2. Click + (Add property) icon to create a new partner group.
3. Enter the details for partner group and throttle, as defined below:

Table 12 Partner Group Configuration

Parameter	Description
Group Name	Partner Group name.
Email	Contact email for the partner group.
Phone	Contact phone for the partner group.
Partner Throttle Chain	Defines the throttle chain which is applied to the group. You can define a list of throttle names. The throttles are applied in the order given in the chain list. The details of the throttles are defined in the **Throttles** tab.

Partner Data

The Partner Data tab allows you to configure the information for a partner who are authorized to access the gateway and send the operation request.

Add Partner Data

To add the data for a new partner, follow the steps:

1. Click on **`Partner Data`** tab.
2. Click + (Add property) icon to create a new partner.
3. Enter the details for partner data, as defined below:

Table 13 Partner Data Configuration

Parameter	Description
Partner Name	Partner name.
Partner Email	Contact email for the partner.
Partner Phone	Contact phone for the partner.
Partner Group	Name of the group of which the partner is a member. Partner group name is defined in the **Partner Groups** tab.
Partner Serial Number	Serial number of the Partner's SSL certificate
Partner Issuer CA	Issuing Certificate Authority of the Partner's SSL certificate.
Enable Secondary ACL	If checked, then this partner exists in the Secondary ACLs list. If not checked, then this partner does not exist in the Secondary ACLs list.
Partner Throttle Chain	Defines the throttle chain which is applied to the partner. You can define a list of throttle names. The throttles are applied in the order given in the chain list. The details of the throttles are defined in the **Throttles** tab.

Partner Operations

The Partner Operations allows you to authorize a specific partner to invoke a specific operation. This tab also defines any throttles that are applied for the operation from this partner.

Add Partner Operations

To add the data for a new partner, follow the steps:

1. Click on **`Partner Operations`** tab.
2. Click + (Add property) icon to create a new partner operation.
3. Enter the details for partner data, as defined below:
4.

Table 14 Partner Operations Configuration

Parameter	Description
Partner	Partner name.
Partner Operation	Name of the Operation for which the partner is granted the access. This operation is configured in **Operations** tab.
Partner Throttle Chain	A list of the throttle names to be applied to any requests sent by this partner which invoke the Operation. The throttles will be applied in the order given in the chain.
Partner Timeout	Timeout in milliseconds to any incoming request from the Partner to this Operation.
Partner Secondary ACL Check	If checked, then this partner is verified in secondary ACL list when the partner invokes the operation.
Forward Mapping	Partner-specific mapping reference to call on inbound message.
Reverse Mapping	Partner-specific mapping reference to call on outbound message.
Allowed Requestor IDs	List of authorized partner's references when partner is an aggregator

Schemas

The Schemas tab allows you to configure the list of XSD files. The XSD files are used to validate the northbound request and response documents.

To add a new schema, follow the steps as below:

1. Click on **Schemas** tab.
2. Click + (Add property) icon to create a new schema.
3. Enter the details for schema files, as defined below:

Table 15 Schemas Configuration

Parameter	Description
Schema Key	Unique id of XSD schema file.
New XSD File	Location of the XSD file.
Existing XSD File	Allows you to select any existing XSD file.

Throttles

The Throttles tab allows you to define different types of throttles with different throttle metrics.

1. Click on **`Throttles`** tab.
2. Click + (Add property) icon to create a new throttle.
3. Enter the details for throttles, as defined below:

Table 16 Throttles Configuration

Parameter	Description
Throttle Parameters	
Throttle Name	Logical name for the throttle. This is the value used in the configuration to identify the throttle.
Throttle Type	Type of the throttle. It can have one of the following values: Rate, Quota, High Water Mark, Error.
Throttle Interval	The time interval during which the throttle is applied. It has different time units depending on the throttle type. For example, Rate and Error throttle types specifies the interval in seconds, Quota throttle type specifies the interval in hours.
Throttle Max Count	Number of requests allowed during the throttle interval. This number should be a positive integer greater than zero.
Throttle Count	Optional

Parameter	Description
Throttle Time Modifiers	Specifies the time modifiers to apply to the throttle. Mutiple time modifiers are allowed. This has following parameters: — Max Count - Throttle limit to apply if time modifier is active (in range). — Start Date - Specifies the start date when the time modifier can be applied. The format is yyyyMMdd. It can be combined with end date to specify a date range. Optional. — End Date - Specifies the end date when the time modifier can be applied. The format is yyyyMMdd. Optional. — Day of Week - Specifies a list of the days of the week when the time modifier can be applied. The order of the days is not important. Optional. — Time Range - Specifies a list of time ranges when the modifier can be applied. The format is hh:mm:ss-hh:mm:ss. Multiple time ranges can be specified. Optional.
Throttle Metric	Specifies the level at which this throttle is applied. The allowed values are Partner, Group, Partner+Operation, Service.
Throttle Metric: Partner	
Partner Name	Name of the partner to which the throttle is applied.
Throttle Metric: Group	
Group Name	Name of the partner group to which the throttle is applied.
Throttle Metric: Partner+Operation It requires both the partner name and operation name to which the throttle is applied.	
Partner Name	Name of the partner to which the throttle is applied.
Operation Name	Name of the operation to which the throttle is applied.

Parameter	Description
Throttle Metric: Service	
Service Name	Name of the service to which the throttle is applied.

ErrorMaps

The ErrorMaps tab allows you to define all the error messages supported by TIBCO ActiveMatrix Service Gateway.

To add a new errormap, follow the steps:

1. Click on **`ErrorMaps`** tab.
2. Click + (Add property) icon to create a new ErrorMap.
3. Enter the details for Errors, as defined below:

Table 17 ErrorMaps Configuration

Parameter	Description
Error Id	External Id of the ErrorMap concept to be created from this line in the configuration file. Must be globally unique.
Status	Transaction status reported by the gateway error.
Component	Gateway component for this error message.
Error Description	Description of the error.
Category	Category of the error. Allowed values are: — ServiceException(SVC:) — PolicyException(POL:)
Fault Code	The SOAP Fault Code. Usually, its client.
Fault String	The SOAP Fault String.
Fault Actor	The SOAP Fault Actor. Currently unused
Message Id	The message id (or any client specific code)
Text	The error text with locations for variables to be embedded indicated by “%1”, “%2” etc
Variables	A list of strings separated by “,” that the gateway client uses for token replacement in the text.

Mapping

The Mappings tab allows you to register the XSLT with TIBCO ActiveMatrix Service Gateway engine

To add a new mapping, follow the steps:

1. Click on **`Mapping`** tab.
2. Click + (Add property) icon to create a new mapping.

Enter the details for Errors, as defined below.

Table 18 Mapping Configuration

Parameter	Description
Mapping Configuration	
Mapping Name	Name of mapping.
Type	RV, XSLT (select from drop down list box).
Type RV	
Subject	RV subject to send the mapping request to
Transformation Name	Transformation to perform on the message.
Type XSLT	
New File	Location of the transformation file.
Existing Files	Existing transformation file.
Response Type	Payload, Full. See Transformations (XSLT Mapping), page 56

Chapter 4 Transaction Pipeline Processing

This chapter discusses the full cycle of transaction pipeline processing and discusses the mapping and transformation capabilities of TIBCO ActiveMatrix Service Gateway software.

Topics

Transaction Pipeline processing

TIBCO ActiveMatrix Service Gateway uses a staged event-driven architecture. TIBCO ActiveMatrix Service Gateway supports the processing of the transactions into a set of stages for high performance.

The transaction pipeline processing consists of following:

- Request pipeline processing
- Response pipeline processing

Request pipeline processing

The request processing cycle indicates the normal life cycle of the incoming request message and consists of the following phases. This life cycle assumes that there are no errors in any of the processing stage.

1. Apache Level Authentication:

   When the partner sends a request using http transport, the first level authentication happens at the Apache server depending upon the first part of uri.

   The authentication type is configured in the apache server configuration file as **asg_mod.conf.** See **Configure Apache web server for Basic Authentication, page 120** **for details.**

   Following types of authentication are supported:

   — No Authentication - This indicates that the request does not have any user credentials. In this case, the request is processed as an anonymous user.

   — Basic Authentication - This indicates that the request has the user credentials. In this case, the user is authenticated.

   — Digest Authentication

   For example, No Authentication is defined as below:

   <Location / >

   SetHandler asg_rv_inbound_handler

   AsgSubject _LOCAL.asg.north.request

   AsgTimeout 30

   </Location>

   For example, Basic Authentication is defined as below:

<Location /asg/ba>

AuthType Basic

AuthName "ASG"

AuthBasicProvider file

AuthUserFile /home/asg/apache/htpasswd/htpasswords

Require validuser

SetHandler asg_rv_inbound_handler

AsgSubject _LOCAL.asg.north.request

AsgTimeout 30

</Location>

After the request is processed by the Apache server for authentication, it is passed to the gateway core engine over Rendezvous transport.

2. Operation Identification:

   When the gateway engine receives the request from the Apache server, it identifies the operation as below:

   — For SOAP requests, the operation is identified from the SOAP Action header, and/or URI as defined in the HTTP header.

   — For HTTP/XML requests, the operation is identified from the combination of method and URI.

   — For HTTP/S REST requests, the operation is identified from the combination of method, URI and the value of some named HTTP header.

   The operation details are configured in the **Operations** tab of the GUI.

3. Partner Identification:

   After the operation is identified, the partner is identified from the request context. The partner data is matched usually against certificate information as configured in **Partner Data** tab of the GUI.

4. Request Parsing (Optional):

   After the operation and partner are identified from the incoming request, the next step is parse the request in order to derive the routing key. Routing key is a key factor to determine the southbound service endpoint for the incoming request. See Routing, page 67 for details.

Routing key is defined in the **Routing** tab of the GUI. It can have following values:

- default: If no XSLT files are defined for the operation, the routing key is default.
- undefined: If no value is returned from the XPath mapper, the routing key is undefined.
- routingKey: routingKey is the value returned from the parsing of the request document as per defined XSLT files.

5. Authorization:

   The gateway engine checks if the identified partner is authorized to run the requested operation by finding the associated configuration in **PartnerOperations** tab.

6. Request message validation:

   If the flag for request validation is enabled, the request message is validated for syntax against XSD for incoming northbound request message.

7. Facade Throttling:

   Facade throttling allows you to enforce the commercial throttles for Service level agreements for a partner request. Facade throttling is done on the partner, partner group, partner operations.

   After the request reaches this stage in the processing pipeline, a check is made on how often can this partner invoke the operation.

8. Forward northbound Mapping:

   After the request passes the facade throttle check, the request is processed by the northbound mapper for any transformations required from operation request message to canonical request message. Whether the mapping is required for this request operation or not, is configured using the **Forward Mapping** field on the **Operations** tab. The transformation details are defined in **Mappings** tab of the configuration GUI.

   By default, If no mappings are defined, then the request message is just copied as the output request message at this stage.

9. Routing:

   Based on the operation name and routing key defined in **Routing** tab of the configuration GUI, the gateway engine determines the name of the southbound service endpoint for this operation.

10. Service Throttling:

Once the name of the southbound service endpoint is derived for the invoked operation, service throttling policy is applied. Service throttles are technical throttles implemented to protect the over use of the service endpoints.

11. Forward southbound mapping:

    If the service throttle is not violated, the request message is processed by the southbound mapper for any transformations required from the canonical request message to the service request message. Whether the mapping is required for this request operation or not, is configured using the **Forward Mapping** field on the **Services** tab. The transformation details are defined in **Mappings** tab of the configuration GUI.

    By default, If no mappings are defined, then the request message is just copied as the output request message at this stage.

12. Invoke southbound service:

    This is the final stage where the gateway engine invokes the southbound service for the requested operation.

Response pipeline processing

The response processing cycle indicates the life cycle of the response message and consists of the following phases:

1. Reverse southbound mapping:

   After the response is received from the southbound service, the response document is processed for any transformations required from the service response message to canonical response message. Whether the mapping is required for this request operation or not, is configured using the **Reverse Mapping** field on the **Services** tab. The transformation details are defined in **Mappings** tab of the configuration GUI.

   By default, If no mappings are defined, then the response message is just copied as the output response message at this stage.

2. Reverse northbound mapping:

   This stage allows any transformations required from canonical response message to northbound response message. Whether the mapping is required for this request operation or not, is configured using the **Reverse Mapping** field on the **Operations** tab. The transformation details are defined in **Mappings** tab of the configuration GUI.

   This mapping can be used for censor response policy., which allows the users to hide certain fields from the response message so that they are not exposed to the requestor.

3. Termination:

   The response message is finally sent back to the original requestor in this stage.

Generate Transaction Logging

After the request and response messages are processed, the gateway engine generates the events to audit log the transaction details. The Central logger component receives the events and logs the transaction details in a database or file, as configured.

Mappings and Transformations

TIBCO ActiveMatrix Service Gateway provide message transformations using mappings. This allows the users to map the gateway endpoint operation request and response messages to the gateway reference operation request and response messages with another format.

The mapping capabilities of TIBCO ActiveMatrix Service Gateway allows you to decouple the northbound operations interface and southbound services interface by providing a mutual common canonical message format between the operations interface and services interface.

TIBCO ActiveMatrix Service Gateway supports the mappings at various levels and reduces the number of point-to-point mappings between gateway endpoint operations (exposed by the gateway) and gateway reference operations (service clients of the gateway that invoke internal service operations).

TIBCO ActiveMatrix Service Gateway provides the transformation of request and response messages at following four points:

— Facade request handler to Router boundary:

  After the request has been received by Facade request handler and before it is been passed to the Router.

— Router to Service endpoint handler boundary:

  After the request has been routed but before it is passed to the service endpoint handler.

— Service Endpoint handler to Router boundary:

  After the response has been received from the service endpoint handler and before it is passed to the Router.

— Router to Facade request handler

  After the response has been routed from the router to the facade request handler and before the response is sent back to the original requestor.

The mapping transformations allows you to:

— Access the multiple versions and formats of service APIs

— Add semantic content validation rules to the incoming requests

— Access the fields from the request context and payload

— Change the routing of request and response messages based on the error handling in case of message validation failures

— Protect the service end-points using service validation policies

TIBCO ActiveMatrix Service Gateway supports the transformations of both request and response messages using:

- Forward mapping - The transformations are done from request canonical form to backend service API as per defined mapping.
- Reverse mapping - The transformations are done from backend service API to response canonical form as per defined mapping.

Mappings allows you to have six different versions of documents as below:

— Northbound request message

— Canonical request message

— Southbound request message

— Southbound response message

— Canonical response message

— Nouthbound response message

TIBCO ActiveMatrix Service Gateway supports mapping at both northbound and southbound sides.

Northbound Mapping

TIBCO ActiveMatrix Service Gateway allows you to map:

— Map northbound request message to canonical request

— Map canonical response to northbound response message

— Map errors to northbound fault response

Northbound mapping configuration is defined in **Operations** and **Partner Operations** tab of the GUI.

Southbound Mapping

TIBCO ActiveMatrix Service Gateway allows you to map:

— Map canonical request to southbound request message

— Map southbound response message to canonical response

Southbound mapping configuration is defined in **Services** tab of the GUI.

Mapping Types

Following types of mapping are supported:

Identity mapping

Identity mapping is a single mapping doing no transformation, and just copies input message to output message recursively. This is called **`Pass-Through`** Mapping.

Rendezvous mapping

Rendezvous mapping allows you to access an external mapping handler as a service using Rendezvous. Rendezvous starter process that calls service endpoint handler and listens to the matched requested subject must exist.

Rendezvous mapping provides external processes to execute mappings between corresponding but not compatible APIs. This provides as set of services called with Rendezvous messages that transform given input message(s) to another message(s). The gateway core engine calls the mappers services whenever it transforms messages between northbound, canonical and southbound form.

XSLT mapping

XSLT mapping describes embedded mapping transformation. The transformation of the request and response documents are done according to the XSLT files defined in the mappings configuration. This can be defined for both forward and reverse mappings. See Transformations (XSLT Mapping), page 56 for details.

Mapping Configuration

The gateway user interface allows you to add the mapping configuration for northbound and southbound request messages (gateway operation endpoint) as well as southbound response and northbound response messages (gateway reference endpoint).

This section explains the steps to configure mappings for gateway endpoint operation and gateway service endpoint.

Identity mapping type configuration

By default, mapping id **ID** is available for forward and reverse mappings to operation endpoint and reference endpoint. ID mapping id refers the Identity mapping. See Identity mapping, page 53.

Rendezvous mapping type configuration

1. Start the GUI, if not already started. See Starting the GUI, page 27 for details.

2. Click the **Mappings** tab

   Enter the following parameters, as defined below:

Table 19 TIBCO ActiveMatrix Service Gateway mapping type configuration

Parameter	Description
Mapping Name	Name of mapping
Type	RV
Subject	RV subject to send the mapping request to
Transformation Name	Transformation to perform on the message. If it is default, the transformation will be determined by mpSubject. Other values force specific transformation.

XSLT mapping type configuration

1. Start the GUI, if not already started. See Starting the GUI, page 27 for details.
2. Click the **Mappings** tab.
3. Enter the following parameters, as defined below:

Table 20 XSLT Mapping Type Configuration

Mapping Name	Name of mapping
Type	XSLT (Select from the drop down list box)
New File	File name containing the transformations (xsl file)
Existing Files	File name containing the transformations (xsl file)

Assign to the gateway operation endpoint

After the mappings are registered, they are assigned to the gateway operation endpoint for forward and reverse mappings to be performed for request operations. They are defined in the **Operations and Partner Operations** tab of the GUI. See Add a new operation, page 30 and Add Partner Operations, page 38.

Assign to the gateway reference endpoint

After the mappings are registered, they are assigned to the gateway operation endpoint for forward and reverse mappings to be performed for request operations.They are defined in the **Services**.tab of the GUI. See Add a new service, page 32

Transformations (XSLT Mapping)

To define XSLT mapping for northbound or southbound request and response messages, XSLT files must be provided in the mappings configuration. See XSLT mapping type configuration, page 54.

For the XSLT mapping, the responseType indicates the output of the transformations, which can be as follows:

Payload

If the responseType is specified as payload, the output document of transformation contains just the payload and is used as the request payload for the next stage in the request pipeline processing.

The output document of the mapping transformation either becomes a canonical request or a southbound request, just depending upon where you are in the request processing pipeline.

For example, the output document from the northbound forward mapping is a payload XML document. This becomes the canonical request and is populated as the input for the southbound forward mapping.

Full

If the responseType is specified as full, it allows you to access the additional fields from the transaction object. The main field is request context field which contains the HTTP headers information, URI, user name, and other authentication fields.

This provides the mapping and transformations of full transaction (request) object. The transaction context contains the header information and the payload. Once the transformation is done:

- the value of the transaction context is replaced by the output document from the transformation.
- the payload is extracted from the transaction context and is used for the next stage in the request pipeline processing.

The transformation with **full** responseType allows you to:

- map and update the document with additional fields available in the request context (HTTP headers, JMS headers etc).
- Set error code for content validation.
- create documents for enumeration orchestration.

Following functionality is implemented using the full responseType of mapping transformation capability:

Set error codes for content validation

With the mapping transformation with responseType as **full**, it allows you to set the error code when the validation of the contents fails. The output of the transformation is a document which contains the updated transaction object with error code.

For example, in case of extra validation of requests like semantic validation using XPath, it sets the error code if it fails. With full mapping type, the error code is updated in the transaction object. The request message processing follows the error handler path in the request processing pipeline, then sent back to the original requestor with the error message. See Implement Request Validation, page 57.

Validation

This functionality allows you to do the schema validation against XSD for an incoming request message. Request validation is performed during the request parsing step and after the Authorization step in the request processing pipeline.

Enabling Validation

To enable the request validation, perform the following:

- Set Operation Features for the operation

  In the **Operations** tab of configuration GUI, set the **Operation Features** parameter as below:

  Operation Features Validation

- Configure Schema

  In the **Schemas** tab of configuration GUI, set the XSD file reference as below:

  New XSD File *Select the location of the XSD file*

Implement Request Validation

Validation of requests is supported for the following stages:

- Request Parsing

Request message can be validated at the request parsing stage. Following elements exist in the schema for the output message to be used by the Parse XSLT:

<errorCode>

<errorMessage>

If errorCode is set, then the request is handled by the error processing path of the processing pipeline. An error XSLT, as defined, for the operation generates an error response message by executing the FaultReportXSLT that is defined for the operation. The request is rejected with the error message sent back as the response to the request.

- Mapping and Transformation Stage

Extended request or response validation can be implemented at any of the four mapping steps. If the message does not pass the validation rules, an error message is constructed and returned in the message.

To set an Error Code in the transformation step, do the following things:

— Include the following name spaces in the XSLT

xmlns:map="http://www.tibco.com/asg/mapping"

xmlns:err="http://www.tibco.com/schemas/asg/error"

— With these namespaces, use the following elements in the output message of the mapper:

<map:mapping-result>

<map:error>

— Within "**http://www.tibco.com/schemas/asg/error**" namespace, use the following error elements to set error code and message details:

<errorCode>

<errorMessage>

<errorBody>

<errorDetails>

<nestedError>

</errorDetails>

With the mapping registered as **Full** mode, if an error element within the registered namespaces is found in the output message, the error attributes for the request object are set with the corresponding values in the elements from the **/schemas/asg/error** schema from the output message.

This instructs the gateway engine to follow the error handling path of the processing pipeline. An error XSLT, as defined, for the operation generates an error message. The request is rejected and the error message is sent back as the request response.

In Parsing step, you can only set **errorCode** and **errorMessage**.

In any of the four mapping steps, you can return four error elements: **errorCode, errorMessage**, and additional elements as **errorBody** and **errorDetails**. In case, if **errorCode** and **errorMessage** are only set, then the request processing is handled in same way as in the parse step. The additional elements, **errorBody** and **errorDetails** elements allows you to construct the actual error response message for the request, which can override the creation of error message by the FaultReportXSLT. This helps to speed up processing as no second XSLT action have to be executed.

Map the protocol headers in request context

This functionality allows the users to do the transformations of the protocol headers with in the request and response messages which includes the transport related information.

For any mapper using the responseType as **full** allows you to access the request context field. The request context field contains the transport level information and is available for any transformations. This allows you to:

- Map the transport header properties and pass it to the next stage in request processing pipeline
- Set the transport header properties. For example, when the JMS transport is used as the channel for incoming request, the JMS priority property can be set based on the XPATH value received in a request.

Enumeration Orchestration

Enumeration Orchestration is implemented using the mapping capability of TIBCO ActiveMatrix Service Gateway.

With parallel orchestration, a single inbound request is split into a set of multiple outbound sub-requests. Each sub-request may be routed differently to various service endpoints. After processing and receiving the responses for each sub-requests, all responses are recombined into a single response message for the original inbound request.

You can define the transformations for the northbound request forward mapping in such a way that the output document contains the sequence that defines the enumeration orchestration. This output document from this transformation serves as an input for the southbound forward request mapping. This contains the `<asg_map:repeat>` tag for enumeration and instructs the gateway engine to split the requests.

Enumeration orchestration is done at the southbound forward mapping meaning that one payload is split into multiple sub-requests independently. It also allows you to override the routing key. For each sub-request, the routing key can be defined so that each sub-request is forwarded to a separate service endpoint. This allows you to modify the routing of sub-requests to a different service at the southbound forward mapping boundary.

Following shows the example snippet of forward southbound mapping transformation for enumeration orchestration:

```
<asg_map:repeat>
 <asg_map:payload-xml>
  <asg_map:mapping-result>

<asg_map:routingKey>getbookbyAuthor_AAA.HTTP</asg_map:routingKey>
  <asg_map:payload>
<SOAP-ENV:Envelope>
 <SOAP-ENV:Body>
  <...>
    <..>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
  </asg_map:payload>
  </asg_map:mapping-result>
 </asg_map:payload-xml>
 <asg_map:payload-xml>
  <asg_map:mapping-result>
  <asg_map:routingKey>getbookbyTitle_BBB.JMS</asg_map:routingKey>
  <asg_map:payload>
<SOAP-ENV:Envelope>
 <SOAP-ENV:Body>
  <...>
    <...>
</SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
   </asg_map:payload
   </asg_map:mapping-result>
 </asg_map:payload-xml>
</asg_map:repeat>
```

For example, GetLocation is a service which finds the location for a single service. This service can be exposed as a service to find location for a group of services. This means that the gateway has to map one request to multiple requests per service. For this case, the payload in the operation request contains a sequence of multiple requests which has to be spilt and sent individually to each service endpoint.

Re-combination of response documents

Transformations are defined at the southbound reverse response mapper boundary to re-join the multiple response documents for each sub-request into a single response document.

Mapping Schemas

Mapping Container

The input document to the XSLT file for XSLT transformations is described by the following schema(XSD):

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified"
attributeFormDefault="unqualified">
<xs:element name="transformation" type="transformationType"/>
<xs:complexType name="transformationType">
<xs:sequence>
<xs:element name="nbRequest" type="stageType" minOccurs="0"/>
<xs:element name="cnRequest" type="stageType" minOccurs="0"/>
<xs:element name="sbRequest" type="stageType" minOccurs="0"
maxOccurs="unbounded"/>
<xs:element name="sbResponse" type="stageType" minOccurs="0"
maxOccurs="unbounded"/>
<xs:element name="cnResponse" type="stageType" minOccurs="0"/>
<xs:element name="nbResponse" type="stageType" minOccurs="0"/>
<xs:element name="context" type="stageType" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="stageType">
<xs:attribute name="href" type="xs:string"/>
</xs:complexType>
</xs:schema>
```

The request payloads and the request context is passed to the XSLT as a map, keyed using the values of the “href” attribute for each element. In order to access the actual payload, it is necessary to load it using the document() function.

The schemas contains the following elements:

- nbRequest – This element represents the northbound request payload.
- cnRequest – This element represents the request payload after the northbound forward mapping has been applied.
- sbRequest – This one element is present for each of the array of request payloads after the southbound forward mapping has been applied
- sbResponse – This one element for each request sent to the southbound service endpoint. It contains either the received response or an error document.
- cnResponse – This element represents the response payload generated by the southbound reverse mapping.
- nbResponse – This element represents the response payload after the northbound reverse mapping has been applied.
- context – This element represents additional context document related to the request. It contains the headers from request and response and transport related information.

At any stage during request processing, only the versions of the request payload created up to that point are available. Though the mapping container has the corresponding element present, an attempt to load the payload via document() causes the XSLT processor to throw an error.

For example to map a value when the received request is SOAP use the following XSLT snippet:

```
<xsl:variable name="nbRequest">
<xsl:value-of select="/transformation/nbRequest/@href"/>
</xsl:variable>
<xsl:choose xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<xsl:when test="count(document($nbRequest)/soap:Envelope/soap:Body)=1">
This is a SOAP request
</xsl:when>
<xsl:otherwise>This is not a SOAP request</xsl:otherwise>
</xsl:choose>
```

Context Document

The contents of the context document are described by the following XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:c="http://www.tibco.com/schemas/asg/context"
targetNamespace="http://www.tibco.com/schemas/asg/context"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="context">
<xs:complexType>
<xs:sequence>
<xs:element ref="c:entry" minOccurs="0"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="entry">
<xs:complexType>
<xs:sequence>
<xs:any namespace="##any" processContents="lax"/>
</xs:sequence>
<xs:attribute name="key" type="xs:string"/>
</xs:complexType>
</xs:element>
<xs:simpleType name="key">
<xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:schema>
```

If the incoming request is received from Apache HTTP server, the context document contains an entry with key set to **asg:httpRequest**, which conforms to the following XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.tibco.com/asg/protocols/http"
targetNamespace="http://www.tibco.com/asg/protocols/http"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="request">
<xs:complexType>
<xs:sequence>
<xs:element name="server-ip" type="xs:string" minOccurs="0"/>
<xs:element name="server-port" type="xs:string"
minOccurs="0"/>
<xs:element name="client-ip" type="xs:string" minOccurs="0"/>
<xs:element name="client-port" type="xs:string"
minOccurs="0"/>
<xs:element name="scheme" type="xs:string" minOccurs="0"/>
<xs:element name="method" type="xs:string" minOccurs="0"/>
<xs:element name="request-uri" type="xs:string"
minOccurs="0"/>
<xs:element name="protocol-version" type="xs:string"
minOccurs="0"/>
<xs:element name="header" minOccurs="0"
maxOccurs="unbounded">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="name"
type="xs:string"/>
</xs:extension>
```

```
</xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element name="body" type="xs:string" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Routing

TIBCO ActiveMatrix Service Gateway provides the routing capabilities to determine the reference endpoint for an operation request. Routing allows you to map a single northbound request document to multiple southbound service interfaces.

The router component of the TIBCO ActiveMatrix Service Gateway receives the request as events and uses the rules engine to determine where requests are handled.

The Router typically works with the canonical version of the request and response documents.

The routing rules are driven by the routing policies, which operate on parameters like:

- Operation [Service Type]: Routing is done based on the requested service, such as a SOAP Action or URL string.
- Requestor [Client]: Routing is done based on the name, type or class of the requesting-application.
- Version: Routing is done based on the version of service APIs. TIBCO ActiveMatrix Service Gateway supports multiple concurrent versions of a service or a service implementation.
- Time-of-Day: Routing is done to different services (explicitly or through orchestrations) depending on time-of-day
- Authenticated Identity: Routing is done to different services based on the Identity of individual partners and consumers. (also supports attribute filtering of responses)
- Message content
- Message and Transport context
- Cross-References with external systems

Configuration

The router component routes the operation request to the service handler based on the configuration settings done on the **Routing** tab of the configuration GUI. The Routing tab allows you to configure the Operation name, Service name and Routing key.

For an Operation configuration, if the Processing Body XLST is defined, then the incoming request is parsed as per the defined XSLT and the routing key is returned. The routing key is the key factor to determine the service handler for an operation. The routing key is extracted from the parsing of the request content as well as context done in the parsing step of the request processing pipeline.

Routing key is defined in the **Routing** tab of the GUI. It can have following values:

— default: If no XSLT files are defined for the operation, the routing key is default.

— undefined: If no value is returned from the XPath mapper, the routing key is undefined.

— routingKey: routingKey is the value returned from the parsing of the request document as per defined XSLT files.

Routing tab

The Routing tab of the configuration allows you to configure the settings required for routing. See Routing, page 35 for details.

Schema document

In the request processing pipeline, the input document to the XSLT for parsing the northbound request is defined by the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified"
attributeFormDefault="unqualified">
<xs:element name="transformation" type="transformationType"/>
<xs:complexType name="transformationType">
<xs:sequence>
<xs:element name="nbRequest" type="stageType" minOccurs="0"/>
<xs:element name="cnRequest" type="stageType" minOccurs="0"/>
<xs:element name="sbRequest" type="stageType" minOccurs="0"
maxOccurs="unbounded"/>
<xs:element name="sbResponse" type="stageType" minOccurs="0"
maxOccurs="unbounded"/>
```

```
<xs:element name="cnResponse" type="stageType" minOccurs="0"/>
<xs:element name="nbResponse" type="stageType" minOccurs="0"/>
<xs:element name="context" type="stageType" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="stageType">
<xs:attribute name="href" type="xs:string"/>
</xs:complexType>
</xs:schema>
```

The following elements are available as input for the transformations and can be used to generate the routing key:

- Northbound request content (as defined by **nbRequest** element)
- Northbound request context (as defined by **context** element)

XLST Transformation for Routing

Following example section illustrates how you can use variables for request content and context as an input to the transformations.

Input:

nbRequestHref is a reference to the **nbRequest** element which indicates the body of the northbound request. Below is an example of **nbRequest:**

the is an example of the XLST transformation used to extract routing key:

```
<xsl:variable name="nbRequestHref">
  <xsl:value-of select="/transformation/nbRequest/@href"/>
 </xsl:variable>
 <xsl:variable name="nbRequest">
  <xsl:copy-of
select="document($nbRequestHref)/soap:Envelope/soap:Body/*"/>
 </xsl:variable>
```

If you intend to use request context, it can be done as below:

```
<xsl:variable name="contextHref">
```

```
  <xsl:value-of select="/transformation/context/@href"/>
 </xsl:variable>
 <xsl:variable name="context">
  <xsl:copy-of select="document($contextHref)/soap:Envelope/soap:Body/*"/>
 </xsl:variable>
```

Output

The output in a transformation is a predefined interface expected from the transformation output. The fields from the output are available to set the values and populate it for routingKey generation using XPath.

<output>

```
 <xsl:variable name="address">
  <xsl:value-of select="$nbRequest/loc:getLocation/loc:address"/>
 </xsl:variable>
 <xsl:variable name="partner">
  <xsl:value-of select="$nbRequest/loc:getLocation/loc:requester"/>
 </xsl:variable>
 <xsl:variable name="opCoId">
  <xsl:value-of select="substring($address,6,2)"/>
 </xsl:variable>
<requester><xsl:value-of select="$partner"/></requester>
<serviceInterfaceVersion></serviceInterfaceVersion>
<referenceId></referenceId>
<serviceId></serviceId>
<timestamp></timestamp>
<correlationId></correlationId>
<identityId></identityId>
<opCoId><xsl:value-of select="$opCoId"/></opCoId>
<partnerId><xsl:value-of select="$partner"/></partnerId>
 <routingKey>
  <xsl:choose>
```

```
    <xsl:when test="$opCoId != ""><xsl:value-of select="$opCoId"/></xsl:when>
    <xsl:otherwise>undefined</xsl:otherwise>
   </xsl:choose>
  </routingKey>
<address><xsl:value-of select="$address"/></address>
</output>
```

Once the routing key is populated, it searches for the right operation defined for this routing key and determines the southbound service endpoint where this operation request is handled. If no routing key is defined in the transformation, it uses the configuration with default routing key.

Overriding Routing Key

In a typical request processing pipeline, routing key is derived based on the northbound request content and context.

TIBCO ActiveMatrix Service Gateway has the capability to overwrite the routing key at the southbound request transformation stage. This functionality is supported using the Enumeration Orchestration. The routingKey is defined in the transformation document (XLST) using the **<asg_map:routingKey>getbookbyAuthor_AAA.HTTP</asg_map:routingKey>** element within the **<asg_map:repeat>** tag.

Chapter 5 Throttles

This chapter explains the throttle functionality of TIBCO ActiveMatrix Service Gateway product.

Topics

Overview

Throttles determine whether a request is allowed to pass-on or whether it should be rejected. Throttles usually provide this determination at the time of a request through state. Throttles have a type and a metric against which some condition is checked.

TIBCO ActiveMatrix Service Gateway uses throttles for following purposes:

- To enforce service level agreements at a partner level.
- To protect itself from over-usage or access from unauthorized partners.
- To protect service endpoints from over-usage or access from unauthorized partners.

You can construct the useful throttles using the state and timers. For example, a simple throttle can be implemented to maintain a counter of the number of requests. This allows the requests to pass-on and count until a specified number is reached, after that the requests are rejected. There is more than one way to implement the rule logic for a throttle. For example, in the above case, the logic can be refined that the requests are allowed to pass-on in an interval of time and count until a specified number is reached, after that the requests are rejected.

If a throttle is accessed by multiple partners, then its state must be synchronized between the partners.

TIBCO ActiveMatrix Service Gateway throttles are implemented as abstract counters and may be shared across multiple policy enforcement points.

Throttle Types

This section explains the throttle types supported by TIBCO ActiveMatrix Service Gateway.

- Rate Throttle:

  This is a simple throttle that allows the requests to pass-through until a limit is reached for a time interval. The rate throttle is always increased on the request.

  Rate throttles are technical throttles which are designed to protect particular service implementations.

- High-Water Mark Throttle:

  This throttle is similar to the Rate Throttle, but it also decrements the count as passed-on requests are completed and the response is ready to return to the

requestor. The High-Water Mark throttle are increased on the request and decreased on the response.

This can be implemented as a Rate throttle, but has to include a flag to indicate that the count decrements on the response.

- Quota:

  This throttle is similar to the Rate Throttle, but it uses a much larger count over much longer intervals (days). The quota throttles are increased on the request.

  Quota throttles are commercial throttles designed to prevent commercial over-use of services, for example wholesale usage.

- Error Throttle:

  Error Throttles acts as a Rate Throttle in logic, but this throttle counts the error responses, as opposed to the requests. The error throttle is increased on the responses.

  Error throttles are technical throttles designed to minimize the impact of external parties.

Throttle metrics

This section explains the throttle metrics associated with supported throttle types:

— Partner

This throttle metric is used for the throttle-rate by a partner.

— Partner + Operation

This throttle metric is used for the throttle-rate by the combination of the partner and the operation.

— Operation

This throttle metric is used in the throttle quota by an operation.

— Service

This throttle metric is used in the throttle rate by the service.

— Time-of-day

This throttle metric allows you to design the throttles that work on the time modifiers, which specify the time ranges when the time modifier can be active. You can specify the multiple day and time ranges. For example, a throttle can relax or tighten its limit based on the time of day, such as between 9am and 11am.

In each throttle metric, a key is used for correct instance identification. For example, **`partner1`** is used to identify that this throttle is configured to operate against the requests from **`partner1`**.

Partner+Operation, Partner and Partner group are defined as Façade Throttles and they are applied once the partner operation is identified.

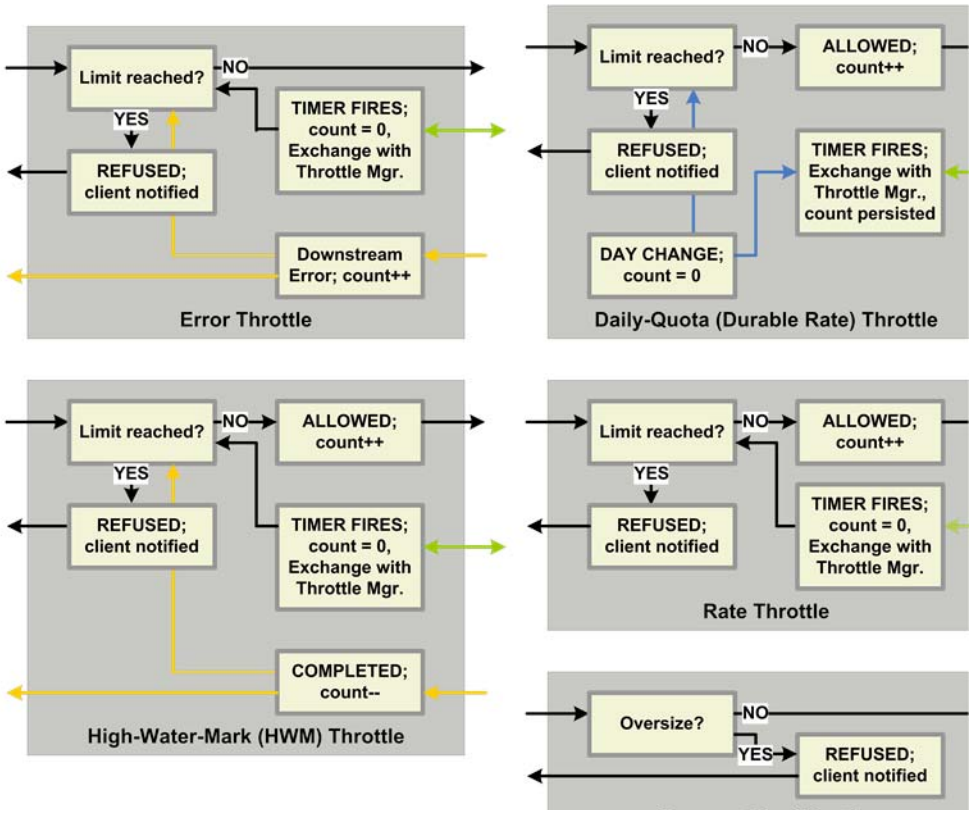
Service throttles are applied after the service has been identified.

For both Facade and Service throttles, if an error occurs before the throttle has been identified, then the throttle is not applied.

The range of throttle metrics is constrained by their availability.

Following diagram shows how the throttles are implemented for a throttle type:

Figure 5 Throttle Types



Throttle Chaining

TIBCO ActiveMatrix Service Gateway allows you to assemble throttles into a throttle chain. Multiple throttles can be configured for a partner request and service endpoint for each enforcement point. This allows a partner request to be passed through a chain of throttles to meet the complex throttling needs. For example, following throttle chains can be configured:

- Rate & High-Water-Mark: This throttle chain addresses interface load.
- Quota: This throttle chain addresses interface agreements.

The configuration of the throttles determines whether the request is passed on to be processed or whether it is applied to any other throttle. When the request is passed on to another throttles, its called throttle chaining.

Order of Throttles in Throttle Chain

Its important to take care of the order of throttles in the chain. For example, if the chain is required to throttle on the quota, then it applies the quota throttle only after rate or high water Mark throttles have been applied. This is important to note the order so that they can be applied in the correct way.

The ordering for the existing types of throttles is recommended as below:

Payload Size-> Error -> Rate -> High Water Mark -> Quota

Configuring Throttles

The configuration GUI allows you to configure all types of throttles. This section explains how to create a throttle policy definition and assign a throttle policy to a partner, partner group, partner operation, or service.

Create a throttle policy definition

This section explains how to configure a quota throttle T1 with a limit of 5 requests within 24 hour for service throttle metric.

Prerequisites

This example assumes that you have completed Configuring an endpoint operation for ActiveMatrix Service Gateway, page 20.

Modify the existing Configuration to add throttles

This section explains to open the existing configuration.

1. Start the GUI, if not already started. See Starting the GUI, page 27 for details.
2. In the left hand panel, select the **`ASG_Get_Start`** configuration.

Define A Quota Throttle

Add a new throttle as follows:

1. Click on **`Throttles`** tab.
2. Click + (Add property) icon to add a new throttle.
3. Enter the throttles details, as defined below:

Table 21 Quota Throttle Configuration

Throttle Name	T1
Throttle Type	Quota
Throttle Interval	24

Table 21 Quota Throttle Configuration

Throttle Max Count	5
Throttle Metric	Service
Service Name	queryBookByAuthorBW

Assign a throttle policy to service

To assign a **Quota** throttle to the service, follow the steps:

1. Click on **`Services`** tab.
2. Edit the **http.GetBooksByAuthor** service.
3. Enter the details, as defined below

Table 22 Edit Service Configuration

Service Name	http.GetBooksByAuthor
Throttle Chain	T1 (select from the drop down list box)

Test the service

1. Launch TIBCO Designer and open the project: *ASG_HOME*/examples/BookQuery/BookQuery
2. Run the following server processes:
   - BooksInterface-service1
3. Run the following client process:
   - QueryByAuthorClient
4. Verify that the process runs successfully without any errors.
5. Send few more requests.
6. Notice that after 5 successful requests, the error appears for Quota Throttle violation.

Chapter 6 Authentication and Authorization

This chapter explains the user authentication and authorization functionality supported by TIBCO ActiveMatrix Service Gateway .

Topics

User Authentication Overview

TIBCO ActiveMatrix Service Gateway supports authentication at two levels:

— Transport Layer

This layer supports the authentication mechanism supported by the transport.

— Gateway Layer

This layer supports Web Services Security Authentication.

Transport and Protocol Level Authentication

When the partner sends a request, the first level authentication happens at the transport layer. TIBCO ActiveMatrix Service Gateway supports authentication for following transports:

— HTTP(s)

TIBCO ActiveMatrix Service Gateway uses the Apache HTTP Server.

— JMS

TIBCO ActiveMatrix Service Gateway uses the TIBCO Enterprise Message Service.

Authentication at Apache HTTP server

Following types of authentication are supported at this layer:

— No Authentication - This indicates that the request does not have any user credentials. In this case, the request is processed as an anonymous user.

— Basic Authentication - This indicates that the request has the user credentials. In this case, the user is authenticated.

— Digest Authentication

— Mutual Authentication using SSL certificates

Refer to the Apache HTTP server documentation to configure the authentication type for Apache server.

Authentication at TIBCO Enterprise Message Service

TIBCO ActiveMatrix Service Gateway provides the authentication mechanism supported by the TIBCO Enterprise Message Service for the JMS transport.

For each of the authentication mechanism, the request is created with header fields for HTTP transport and JMS Application Header fields for JMS transport which are part of the request context message. This request context message becomes part of the RV message using the protocol termination functionality when forwarded to the gateway.

WS Security Services Authentication

TIBCO ActiveMatrix Service Gateway supports the web services security authentication services for the northbound messages.

TIBCO ActiveMatrix Service Gateway supports following security token profiles:

— User name token

TIBCO ActiveMatrix Service Gateway provides the user authentication for the northbound requests with the LDAP system.

— SAML

TIBCO ActiveMatrix Service Gateway provides SAML based sign-in authentication of the northbound requests.

— X.509

TIBCO ActiveMatrix Service Gateway uses X.509 protocol to process the request messages for signatures.

TIBCO ActiveMatrix Service Gateway provides the processing of northbound messages as follows:

- Northbound Request Messages

  The gateway engine processes the signatures and encrypted elements of the incoming request messages, if specified.

- Northbound Response Messages

  The gateway engine supports the signing and encryption of the elements of northbound response messages, as specified in the configuration.

Configuring Web Services Security Authentication

This section explains the procedure to configure the web services security for the gateway engine.

Define the WSS Configuration Properties file

This section explains how to define the properties files required for the WSS shared resources configuration.

Sample Files

TIBCO ActiveMatrix Service Gateway provides the sample configuration file for the shared resources for each of the security type profile. It is recommended to use the sample files as templates and edit the properties as per your requirement. The sample files are located in the *ASG_CONFIG_HOME*/**asg/default/wss** directory.

The property files are defined depending on the type of WSS configuration selected. Following section explains the WSS type and a sample property file which can be used for that type:

- User name token

TIBCO ActiveMatrix Service Gateway authenticates the user with the LDAP system and requires to create the configuration file for LDAP configuration as follows:

LDAP configuration for bind mode

This configuration type provides the authentication based on the user name token with a LDAP system for bind mode.

The sample file **req_usernametoken_ldapbind.properties** for LDAP shared resource configuration is located in the following directory: *ASG_CONFIG_HOME*/**asg/default/wss**

You can use this file as a template and edit the LDAP server properties as per your environment.

LDAP configuration in bind mode with SSL Enabled

This configuration type provides the authentication based on the user name token with a LDAP system with SSL enabled in bind mode.

The sample file **req_usernametoken_ldapbindssl.properties** for LDAP shared resource configuration is located in the following directory: *ASG_CONFIG_HOME*/**asg/default/wss**

You can use this file as a template and edit the LDAP server with SSL properties as per your environment.

LDAP configuration for search mode

This configuration type provides the authentication based on the user name token with a LDAP system for search mode.

The sample file **req_usernametoken_ldapsearch.properties** for LDAP shared resource configuration is located in the following directory: *ASG_CONFIG_HOME*/**asg/default/wss**

You can use this file as a template and edit the LDAP server properties for search mode as per your environment.

- Subject Identity

The configured keystore along with a valid key from keystore can be used to provide an identity of the interested subject. Identity provider takes as an input the password of the Key alias, and it is used to access the private key of that particular alias. This is used for signing.

TIBCO ActiveMatrix Service Gateway requires certain properties to be defined for this type. These properties are defined in a file, which, can be imported in the configuration GUI. See Define the WSS Configuration Properties file, page 83

This configuration type provides the properties for the keystore configuration (private key) to sign the message or decrypt the message.

The sample file **resp_sign.properties** describes the keystore properties required to sign the message. This file is located in the following directory: *ASG_CONFIG_HOME*/**asg/default/wss**

You can use this file as a template and edit the keystore configuration as per your environment.

- Trust Identity

The trust store consumes a keystore provider and it is used for accessing public keys of the keys for signature verification or for encryption.

TIBCO ActiveMatrix Service Gateway requires certain properties to be defined for this type. These properties are defined in a file, which, can be imported in the configuration GUI. See Define the WSS Configuration Properties file, page 83

This configuration type provides the properties for the keystore configuration to verify the signatures or encrypt the message.

The sample file **resp_encrypt.properties** describes the certificate keystore properties required to encrypt the message. This file is located in the following directory: *ASG_CONFIG_HOME*/**asg/default/wss**

You can use this file as a template and edit the keystore configuration as per your environment.

Register WSS resources with TIBCO ActiveMatrix Service Gateway

The **WSS** tab on the configuration allows you to register the WSS resources with TIBCO ActiveMatrix Service Gateway gateway.

To setup the WSS configuration, perform the following steps:

1. Start the GUI, if not already started.
2. Click **WSS** tab.

3. Enter the details for WSS, as defined below:

Table 23 WSS Configuration

Parameter	Description
WSS Name	Unique name which identifies a WSS configuration.
Type	Type of the WSS configuration. WSS Subject Identity Trust Identity
New Property File	A new property file which defines the WSS resources configuration. See Define the WSS Configuration Properties file, page 83.
Existing Property Files	Select a configuration file of WSS resources configuration.

Define the WSS security operations

This section explains the steps to define a WSS enabled security operation. An operation is WSS enabled using the **Operations** tab of the GUI.

To setup the WSS configuration, perform the following steps:

1. On the configuration GUI, Click **Operations** tab.
2. Add a new operation. Enter the details of the Operation. See Add a new operation, page 30
3. Check the **Enable WSS** checkbox.
4. Enter the details for WSS enabled operation as defined below:

Table 24 WSS Enabled Operation Configuration

Parameter	Description
WSS Request	This is the name of the WSS configuration from **WSS** tab. The property file from this configuration is used for northbound request processing.
WSS Response	This is the name of the WSS configuration from **WSS** tab. The property file from this configuration is used for northbound request processing.
Encrypt Response	This checkbox flag indicates whether to encrypt the response message.

Table 24 WSS Enabled Operation Configuration

Parameter	Description
Sign Response	This checkbox flag indicates whether to sign the response message.
Encryption Algorithm	This list box allows to select the algorithm to use for encryption. Supported values are: TRIPLE_DES, AES_128, AES_256, AES_192
Key Algorithm	This list box allows to select the algorithm to use for key. Supported values are: RSA15, RSAOEP, AES128, AES192, AES256, TRIPLEDES
Signing Algorithm	This list box allows to select the algorithm to use for signing. Supported values are: HMAC_MD5, DSA_SHA1, HMAC_SHA1, RSA_SHA1, RSA_MD5, RSA_RIPEMD160, RSA_SHA256, RSA_SHA384, RSA_SHA512, ECDSA_SHA1, ECDSA_SHA224, ECDSA_SHA256, ECDSA_SHA384, ECDSA_SHA512, HMAC_RIPEMD160, HMAC_SHA256, HMAC_SHA384, HMAC_SHA512
Key Type	Supported values are: BST_DIRECT_REFERENCE, ISSUER_SERIAL, X509_KEY_IDENTIFIER, SKI_KEY_IDENTIFIER, EMBEDDED_KEYNAME, EMBED_SECURITY_TOKEN_REF, UT_SIGNING, THUMBPRINT_IDENTIFIER, CUSTOM_SYMM_SIGNING, ENCRYPTED_KEY_SHA1_IDENTIFIER, CUSTOM_SYMM_SIGNING_DIRECT, CUSTOM_KEY_IDENTIFIER, KEY_VALUE
Keystore Alias	A keystore file alias as configured in the Keystores tab of the UI.

Partner Authorization Overview

TIBCO ActiveMatrix Service Gateway supports the authorization based on following actions:

Operation Identification

TIBCO ActiveMatrix Service Gateway identifies the operation based as follows:

— For SOAP requests, the operation is identified from the SOAP Action header, and/or URI as defined in the HTTP header.

— For HTTP/XML requests, the operation is identified from the combination of method and URI.

— For HTTP/S REST requests, the operation is identified from the combination of method, URI and the value of some named HTTP header.

Partner Identification

TIBCO ActiveMatrix Service Gateway uses the Partner Serial number and Partner Issuer CA from the header fields to uniquely identify the partner. The gateway maps the authenticated users from the transport headers to validate the identified partner in the gateway configuration repository.

The Partner Serial Number and Partner Issuer CA are configured on the **Partner Data** tab of the configuration GUI for a partner.

Make sure that the Partner Issuer CA, as configured on the GUI, matches with the issuer CA defined in the *ASG_CONFIG_HOME*/**asg/asg.properties** file by the following properties:

- **For HTTP transport**

  — For basic authentication

  ```
  tibco.clientVar.ASG/Northbound/CertificateAuthority/Basic=O=ST;CN=ASG;CN=HTTP;CN=Basic Domain
  ```

  — For anonymous user,

  ```
  tibco.clientVar.ASG/Northbound/CertificateAuthority/Anonymous=O=TIBCO;CN=ASG;CN=RV;CN=Anonymous
  ```

- **For JMS transport**

  ```
  tibco.clientVar.ASG/Northbound/CertificateAuthority/ESB=O=ST;CN=ASG;CN=JMS;CN=ESB
  ```

  ```
  tibco.clientVar.ASG/Northbound/SerialNumber/ESB=12345
  ```

For example, for HTTP/(s) transport:

1. If no user is specified in the incoming request:

   For this case, the gateway engine considers this request as a request from anonymous user. The gateway engine looks for the Partner Serial Number as **Anonymous** on the configuration GUI and matches the Partner Issuer CA from this configuration with the issuer CA defined in the *ASG_CONFIG_HOME*/**asg/asg.properties** file by the following property:

   ```
   tibco.clientVar.ASG/Northbound/CertificateAuthority/Anonymou
   s=O=TIBCO;CN=ASG;CN=RV;CN=Anonymous
   ```

   If there is a mismatch, then the gateway engine rejects the partner with Authorization error.

2. If the gateway engine receives the request using basic authentication mechanism:

   For this case, the gateway engine retrieves the user name from the request headers. The gateway engine looks for the **user name** as specified in the request headers with the Partner Serial Number on the configuration GUI and matches the Partner Issuer CA from this configuration with the issuer CA defined in the *ASG_CONFIG_HOME*/**asg/asg.properties** file by the following property:

   ```
   tibco.clientVar.ASG/Northbound/CertificateAuthority/Basic=O=
   ST;CN=ASG;CN=HTTP;CN=Basic Domain:
   ```

3. If the gateway engine receives the request using SSL authentication mechanism:

   For this case, the gateway engine retrieves the user name and issuer CA from the request headers. The gateway engine matches the **user name** and **issuer CA** as specified in the request header with the Partner Serial Number and the Partner Issuer CA on the configuration GUI.

Partner Authorization

After the operation and partner is identified, TIBCO ActiveMatrix Service Gateway validates that the identified partner is authorized to invoke the operation.

Chapter 7 Gateway Management Features

This chapter explains the functionality of TIBCO ActiveMatrix Service Gateway management component.

Topics

Central Logger

Overview

The TIBCO ActiveMatrix Service Gateway provides centralized logging via the Central Logger component. Each Gateway server instance publishes logging events during its operation. The Central Logger component receives the logging events from the management bus as messages. The Central Logger stores the messages in the database or in a file. The Gateway server instance aggregates logging events in memory and publishes the messages at configured intervals in order to reduce the disk load during high transaction rates.

Pre-requisites

The Central Logger component audit logs the transactions in a database or a file. In production systems, its recommended to use a database server. Make sure to configure and setup a database server for the functionality of the central logger component. See Database Setup and Configuration for Central logger, page 92

Database Setup and Configuration for Central logger

This section guides you through the steps required to configure the database and set up the database drivers for TIBCO ActiveMatrix Service Gateway Central Logger component.

Database Location

Instructions in this section assume you are working with a local database for testing purposes. Adapt the instructions if you are working with a remote database. For example, in production environments, you might have to ask a database administrator to create a database and a database user for you.

Make sure that you have access to a running database server instance required for Central Logger component.

Task A Setup Database Driver

- Copy the appropriate JDBC driver jar file to *ASG_HOME*`/lib/ext/tpcl` directory. See the product readme file for the supported versions of database drivers.

For example, database jar file for **`mysql`** database is:

```
mysql-connector-java-5.1.18-bin.jar
```

For example, database jar file for **`oracle`** database is:

```
ojdbc6.jar
```

Task B Create a database

Depending upon your environment, you might have to ask your database administrator to create a database or use an existing database. For example, for **`mysql`**, you can create a local database for your testing purposes. For production environments using **`oracle`** database server, ask your database administrator to create a database for TIBCO ActiveMatrix Service Gateway central logger component.

This section lists the steps to create a database for testing purposes using **`mysql`** database server.

1. Verify that the **`mysql`** database server is running.
2. log on to the database server using the command:
   `mysql -uroot -p`
3. Enter the password when prompted.
   `Enter password:`
4. Type the following command at the mysql command prompt to create a new database:
   `create database asgstat;`
5. Verify that the `asgstat` database is created.

Refer to **`http://dev.mysql.com/doc/refman/5.5/en/creating-database.html`** link for more information.

Task C Create a database user

For testing purposes, this section explains the steps required to create the database user with appropriate privileges in a database, using the **`mysql`** database server.

For production systems using **`oracle`** database server, work with your database administrator.

1. Verify that you are connected to the mysql database server as a root user. If not, type the command:
   `mysql -uroot -p`
2. Type the following command at the mysql command prompt to create a new user:
   `create user 'asguser' identified by 'asgpass';`

3. To grant the appropriate privileges to the database user, type the following command at the mysql command prompt:
```
grant create,select,insert,update on asgstat.* to asguser@'%'
identified by 'asgpass';
```
4. Type the following command to reload the privileges from the grant tables in the database:
```
flush privileges;
```

For Oracle Database

Ask your database administrator to create a database user (for example **asguser)** and grant the **connect,resource** privileges to this user.

Task D Setup the database schema

This section explains the steps to create the database tables in the database required for the TIBCO ActiveMatrix Service Gateway Common Logger component.

For mysql Database

1. Navigate to the following directory:

   ASG_HOME/**config/database/mysql**
2. Type the following command at the command prompt:
```
mysql -D asgstat -uasguser -pasgpass < createAsgTransactions.sql
mysql -D asgstat -uasguser -pasgpass < createAsgKpis.sql
```

For Oracle Database

1. Navigate to the following directory:

   ASG_HOME/**config/database/oracle**
2. Type the following command at the command prompt: (Replace SID with the actual oracle database SID name)
```
sqlplus asguser/asgpass@SID @createAsgTransactions.sql
sqlplus asguser/asgpass@SID @createAsgKpis.sql
```

Task E Setup the database connection parameters

This section explains the steps required to setup the parameters to connect to the database.

1. Open the *ASG_CONFIG_HOME*/config/asg_cl.properties file for editing.
2. Edit the following parameters to provide the values to connect to the appropriate database:

```
tibco.clientVar.CL/Database/Driver=database driver type
tibco.clientVar.CL/Database/Url=database url
tibco.clientVar.CL/Database/Username=database user name
tibco.clientVar.CL/Database/Password=database password
```

For example, values are shown below for mysql database:

```
tibco.clientVar.CL/Database/Driver=com.mysql.jdbc.Driver
tibco.clientVar.CL/Database/Url=jdbc:mysql://localhost:3306/asg
stat
tibco.clientVar.CL/Database/Username=asguser
tibco.clientVar.CL/Database/Password=asgpass
```

For example, values are shown below for oracle database:

```
tibco.clientVar.CL/Database/Driver=com.oracle.jdbc.Driver
tibco.clientVar.CL/Database/Url=jdbc:oracle:thin:@localhost:
1521:ORCL
tibco.clientVar.CL/Database/Username=asguser
tibco.clientVar.CL/Database/Password=asgpass
```

For the database Url field, replace localhost with the host name where database server runs, if needed.

Enable Reporting to Central Logger

By default, the reporting to central logger component is not enabled. You can enable the reporting as follows:

1. Open the *ASG_CONFIG_HOME*/config/asg.properties file for editing.
2. Look in the file for the following section:

   # Turn on or off reporting to CL

   tibco.clientVar.ASG/Logging/reportingEnabled=false
3. To enable the reporting, set the value as:
   tibco.clientVar.ASG/Logging/reportingEnabled=true
4. Save the file and close the editor.

Run Central Logger

This section explains the steps to run the central logger.

1. Open a terminal window.
2. Navigate to *ASG_HOME*/bin directory.
3. Type the following command to start the Central logger:

   ```
   ./asg-engine -u asg-cl
   ```

Global Throttle Manager

The Global Throttle Manager manages the Façade Throttle Manager and Service Throttle Manager. The Global Throttle Manager reports throttle usage to the Central Logger.

The Global Throttle Manager component maintains the state of all the global throttles in both Façades (Façade Throttles) and Routers (Service Throttles). The Global Throttle Manager exchanges the states of global throttles with active Façade Throttle Managers and Service Throttle Managers.

The Global Throttle Manager component provides the mechanism to evenly distribute the global throttles among TIBCO ActiveMatrix Service Gateway server instances.

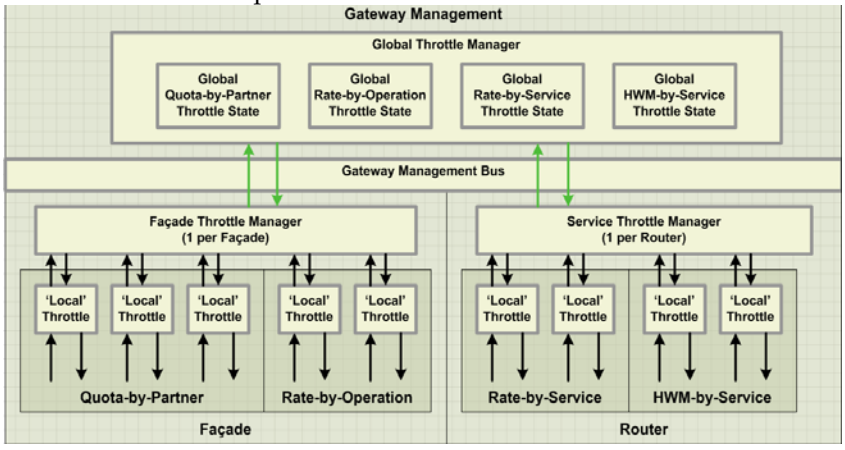
The Global Throttle Manager component allows you to implement simple group throttles. Below are few examples of throttles types that are used in the Façade or Router as global throttles:

— Commercial throttles implement gross usage agreements.

— Partner throttles act to limit a partner's impact on internal services.

— Operation throttles implement fine-grained usage agreements.

— Technical throttles protect service interfaces.



Throttle Calculation

Global Throttle Manager calculates the throttles as follows:

For Quota and High-Water-Mark throttles, it uses the following formula to get the max count per engine:

```
MaxCountPerEngine = ThrottleMaxCount/maxActiveEngines
```

For Rate and Error throttles, the calculation is done as follows:

```
MaxCountPerEngine =
(ThrottleMaxCount/maxActiveEngines)*(UpdateIntervalSec/Throttle
Interval)
```

- ThrottleMaxCount and Throttle Interval are defined in the throttle configuration.
- UpdateIntervalSec is the time interval in seconds for sending throttle updates to Global Throttle Manager. The default value is 10 seconds and can be edited in the *ASG_HOME*/asg/1.1/templates/asg/**asg.properties** file. This is defined by the following property:

  tibco.clientVar.ASG/Throttle/UpdateIntervalSec=10

The value of MaxCountPerEngine is always rounded up. For example, 1.1 will be 2 and 1.9 will also be 2.

For non zero throttles, this rounding means that the MaxCount Per Engine will always be at least 1. For example, if there are 20 active engines and the calculated value of MaxCountPerEngine is 0.4, then the effective throttle limit is calculated as:

```
Ceiling(0.4)*active engines = 1*20 = 20
```

For this case, the value of MaxCountPerEngine is not 8. (0.4 * 20)

This indicates that if there are many active engines and the throttles are defined in this setup, then the MaxCount can never be less than the number of active engines.

Run Global Throttle Manager

This section explains the steps to run the Global throttle manager.

1. Open a terminal window.
2. Navigate to *ASG_HOME*/bin directory.
3. Type the following command:

   ```
   ./asg-engine -u asg-gtm
   ```

Make sure that the cache agent and the gateway core engine are running before you start the global throttle manager.

Cache Clearing Manager

The Cache Clearing Manager provides mechanism to clear the cache. The Cache Clearing Manager manages the eviction of entries in the associative cache, especially for the entries that are not often referenced.

TIBCO ActiveMatrix Service Gateway server stores the responses from side-bound service requests and is used for future look-ups. Cached objects have a time to live, which is evaluated every time an entry is retrieved from the associative cache and the entry gets evicted on lookup if the time to live(TTL) has expired. Cache entries with a relatively short time to live(TTL) that are not often read, might pollute the associative cache using system resources at no benefit. For that reason the Cache Clearing Manager evaluates all cache entries on a scheduled time basis and evicts these entries who's time-to-live has expired.

How To Run Cache Clearing Agent

This section explains the steps to run the cache clearing manager:

1. Open a terminal window.
2. Navigate to *ASG_HOME*/bin directory.
3. Type the following command:

```
./asg-engine -u asg-cache-cleanup
```

Its recommended to start the cache clearing manager after the cache agent is running successfully. If the cache clearing agent starts before the cache agent, there could be a potential possibility of additional cache agents not being able to connect to the cache provider using the discovery listener and function properly.

Monitoring and Management Server

The Monitoring and Management component allows you to deploy the cache-based TIBCO ActiveMatrix Service Gateway engines, and then monitor and manage a deployed cluster. It works with the following sub-components:

- Site topology editor

  The site topology editor is a canvas-based editor, which allows users to set the deployment configuration of the cluster by specifying its hosts and deployment mappings.

- MM Console

  MM Console is a web-based dashboard that enables you to monitor the deployment and perform various operations on the cluster.

The Monitoring and Management components enable you to easily identify bottlenecks and other problems in the system. MM also has a profiler and can generate other helpful reports. The MM component maintains a live history of statistics, continuously queries the deployed engines for their status, and invokes operations on engines. Various overview panels and panes provide graphical views and alerts about the health of the cluster. You can configure the health thresholds as desired.

Pre-Requisites for Deployment

Monitoring and Management server requires that you specify the path to the Site topology file in the MM.cdd before you can start the MM server and initiate deployment.This file is created using the site topology editor in TIBCO ActiveMatrix Service Gateway Studio.

Sample Site topology file

You can find a sample of the site topology file shipped with the product installation. This file is located as:

ASG_HOME/mm/bin/**asg_default.st**

Deploying and Managing Gateway Engines with MM

This section describes the steps required to deploy gateway engines using MM.

Configuration Settings

This section explains the configuration steps required before you can deploy gateway engines.

Edit JMX properties in Engine TRA File

1. Open the following file for editing:

   ASG_HOME/bin/**asg-engine.tra**

2. Uncomment the following properties:

   ```
   java.property.be.engine.jmx.connector.port=%jmx_port%
   java.property.be.engine.jmx.connector.authenticate=false
   ```

Edit MM.cdd File

The CDD file used by the MM server is the `MM.cdd`. In this file, you must specify the path to the site topology file. The `MM.cdd` file is a XML file and can be edited using any XML or text editor.

Before you make any changes to the `MM.cdd` file, make sure to take a back up of this file.

1. Navigate to the *ASG_HOME*/mm/bin directory.

   Edit the MM.cdd and change the value of **be.mm.topology.file** property, if required. This property must be set to the fully qualified path of the site topology file for the cluster to be monitored. Specify the location of the site topology file you want to be loaded by the MM server.

   By default, the property is set to the sample of a site topology file, as shown below:

   <property name="be.mm.topology.file"
   value="*ASG_HOME*/mm/bin/asg_default.st"/>

   The sample site topology file is located as:

   ASG_HOME/mm/bin/**asg_default.st**

Whenever you make changes to the `MM.cdd` file you must restart the MM server so that it uses the updated values.

Edit Site Topology File in a Text Editor

The site topology file contains deploy time information such as what processing units to deploy to specific computer/hosts in your environment. You need to know information about the computers that will host the agents you intend to deploy, for example the information about the machines' operating system and IP address.

An annotated site topology file that can be used as template is located at:

ASG_HOME/mm/bin/asg_default.st

You only need to edit the topology file if the defaults don't serve your purposes. This happens typically when the gateway engines are to be deployed on a different host than the one where the Monitoring and Management server is running.

See Site Topology Reference, page 132 for details on the XML elements defined in the topology file.

Make sure that you take a backup copy of the topology file before you make any changes.

Deployment

This section explains the steps to deploy the gateway engines after the configuration settings are set.

Run MM Server

This section explains how to run the Monitoring and Management Server.

1. Using the command line terminal, navigate to ASG_HOME/mm/bin.
2. Type the following command to run the MM server:

```
./asg-mm -u default -c MM.cdd MM.ear
./asg-mm -help to view usage information.
```

Log On to MM Console

After the Monitoring and Management Server has started, users can log on to MM Console.

1. In a web browser, enter the URL for the console. By default the URL is:
   `http://IP_COMPUTER_HOSTING_MM_SERVER:9000/index.html`

The hostname and port are configured in the `MM.cdd` file. See the following properties in this file:

<property name="tibco.clientVar.HTTPHost" value="localhost"/>

<property name="tibco.clientVar.HTTPPort" value="9000"/>

2. Log on using the user credentials that were configured in the passwords (users.pwd) file. As shipped with the installation, the default credentials are `admin/admin`.

The user credentials are specified in the `users.pwd` file. By default, this file is located in *ASG_HOME*/mm/config directory. The file location is specified by the following property in the `MM.cdd` file:

<property name="be.mm.auth.file.location"
value="*ASG_HOME*/mm/config/users.pwd"/>

3. After you login, you should see the Cluster Explorer in the left panel, and the Cluster Overview on the right.

See Working with Cluster Explorer on page 137 for an introduction to the MM console user interface.

Deploying the Gateway Engines with MM Console

You can deploy gateway engines using the MM Console. You can only deploy gateway engines that are predefined in the site topology file. To deploy, proceed as follows:

1. Log on to MM Console. SeeLog On to MM Console, page 101.
2. From the Cluster Explorer, select the machine node you want to deploy.
3. Select the icon of the host machine where you want to deploy and click the Deploy button.

   The deployment unit that you configured to deploy on that machine in the site topology file deploys.
4. Verify the login details or provide them. Default values are the credentials you specified in the site topology file for this host. If you want to encrypt the password, you must use the Site Topology Editor in TIBCO ActiveMatrix Service Gateway Studio.
5. Click **OK**. The gateway engines configured to deploy to that machine deploy.

To see if an engine or agent is deployed Hover the mouse over its name in the explorer panel. A tooltip shows if it is deployed or undeployed.

Deployment Time Deployment time information is saved to a file located under ASG_HOME/`mm/deployed`, and the last deployment time is displayed in the UI.

Deploying And Starting the Gateway Engine for a configuration

By default, the gateway engines are started using the **`default`** configuration folder. To start the gateway engines for any other configuration (for example, BookQuery), perform the following steps:

1. Navigate to the *ASG_CONFIG_HOME*/`asg` directory.
2. Edit the `asg.properties` file to change the value of the following property as shown below:

   `tibco.clientVar.ASG/ConfigRoot=`*ASG_CONFIG_HOME*`/asg/BookQuery`

Deploying the Gateway Engines on a Remote Machine Node

To deploy the gateway engines to a remote machine, you must have:

— SSH server running on the remote machine.

SSH software enables MM to deploy TIBCO ActiveMatrix Service Gateway engines to the predefined hosts, that is, those configured in the monitored cluster's topology file. SSH can also be used to start and stop remote engines. The SSH utility is available on UNIX machines by default. No action is required.

— Edit the topology file for the deployment unit, processing unit and host settings. See Site Topology Reference, page 132 for details on the properties defined in this file.

Reporting

The Central Logger component stores reporting information to its database and optionally to a file. The Apache HTTP server also generates access logs.

The data from the central logger database can be used for reporting. It contains the following data:

- High level Transaction Auditing
  - One entry for every received request
  - Timing information
  - Service, operation and partner identities
  - Status of overall transaction
- Detailed level Transaction Auditing
  - Request payload before and after every transformation
  - Multiple identities associated with each transaction
  - The list of stages through which the transaction passed
  - Ad-hoc messages generated during transaction processing
- Key performance indicators
  - Count of transactions received in a given time interval
  - Per partner, service and operation
  - Default time intervals one minute, five minutes and one hour

TIBCO Spotfire Integration

TIBCO ActiveMatrix Service Gateway provides a sample TIBCO Spotfire analysis based on data captured by the Central Logger component. The sample shows the following metrics:

- Transaction rate per partner and service
- Throttle violation, Service quality and Service timeouts failures
- Load Visualization for operations and management
  - Volume
  - Latency
  - Peaks

- Defect detection
- System Usage

Spotfire Configuration

This section describes the steps for TIBCO Spotfire Professional configuration. The Spotfire client will retrieve the data stored in TIBCO ActiveMatrix Service Gateway database to display the reports.

Task A Configure TIBCO Spotfire Server and Client

This section assumes that you have a running instance of Spotfire server. Ask the Spotfire administrator to create a username and password to allow access on the Spotfire server instance.

Spotfire User Permissions

Make sure that the Spotfire Administrator creates a user to be part of the Library Administrator group on the Spotfire server instance. Ensure that the Administrator grants the library administration permissions to this user. This allows the users to:

- Create library contents (such as data sources, information links).
- Import and Export the library content.
- Store the analysis files.

1. Install TIBCO Spotfire Professional software. Refer to the product readme for the supported version.
2. Launch the TIBCO Spotfire Professional.
3. Connect to the appropriate TIBCO Spotfire server instance by providing the correct URL in the Server field.
4. Download the software package updates from the server, if prompted.
5. Verify you have successfully connected to the TIBCO Spotfire server instance for TIBCO ActiveMatrix Service Gateway and updated your TIBCO Spotfire client to the version that is deployed on the TIBCO Spotfire server.

Task B Setup a Spotfire Data source

1. Within TIBCO Spotfire Professional, open the Information Designer from the menu bar as: Tools -> Information Designer.
2. Click the Setup Data Source link.
3. Input the **Data Source** fields, as appropriate.

For mysql database, example values are shown as below:

Name: **asgstat**

Type: **MySQL5**

Connection URL: jdbc:mysql://database host:port/database

(where database host is the machine database server runs, by default the port is 3306, and database is the database name such as asgstat).

No. of connections:

— Min: 1

— Max: 10

Username: **asguser**

Password: **asgpass**

For oracle database, example values are shown as below:

Name: **asgstat**

Type: **Oracle**

Connection URL: jdbc:oracle:thin:@database host:port:service name

(where database host is the machine database server runs, by default the port is 1521, and the service name is the Oracle service name)

No. of connections:

— Min: 1

— Max: 10

Username: **asguser**

Password: **asgpass**

4. Click **Save** button.
5. Click **Save** button again, if pop window appears.
6. Verify that this data source appears in the left panel of the Information Designer dialog with a + sign next to it.

Task C Create a Spotfire Information Model for TIBCO ActiveMatrix Service Gateway Central Logger

1. Within TIBCO Spotfire Professional, open the Information Designer from the menu bar as: Tools -> Information Designer.

2. In the left panel of the Information Designer dialog, expand the data source name by clicking on the + sign. Expand the asguser schema.
3. Verify that following database tables are listed from the Central Logger database schema:
   - ASG_KPI
   - ASG_LOG_MESSAGES
   - ASG_THROTTLE_MESSAGES
   - ASG_THROTTLE_USAGE
   - ASG_TRANSACTION_DETAILS
   - ASG_TRANSACTION_KEYS
   - ASG_TRANSACTION_MESSAGES
   - ASG_TRANSACTIONS
4. Shift-click on the first table and the last table to select all tables. With all the tables selected, right-click on your mouse and select the option **Create default Information Model** from the context menu. Click the OK button on the next new dialog window.
5. On the **Create Default Information Model Settings** window, ensure that the radio button **`Automatically assign a new name to the created item`** is selected. Click the **OK** button.
6. Verify that the Information Links has been created for each table of the Central Logger database, which appears in the left pane of the Information Designer on the Elements tab.
7. Click the **Close** button to close the Information Designer.

Task D Deploy the default TIBCO ActiveMatrix Service Gateway Audit Trail DXP file on TIBCO Spotfire server

1. Within TIBCO Spotfire Professional, open the *ASG_HOME*`/templates/spotfire/`**`ASG_LogData.dxp`** Spotfire visualization file by selecting the menu option File -> Open.

2. If the analysis file opens with *Missing Information Link* warnings, follow the steps:
   - Select the **Browse for the missing information link** radio button and click the OK button.
   - In the **Select Information Link** dialog window, select the ASG_TRANSACTIONS information link (with the paperclip icon next to it) and click the OK button.
   - Repeat these steps for the following missing information links:

     ASG_TRANSACTION_DETAILS

     ASG_TRANSACTION_MESSAGES

     ASG_KPI

     ASG_THROTTLE_USAGE
3. Note that the analysis file opens without any errors. This file does not show any data as the ASG Central Logger database is empty.
4. Save the **`ASG_LogData.dxp`** analysis file in the Spotfire Server by selecting menu option File -> Save as -> Library Item. In the Save As Library Item dialog window that opens, click on the Finish button.
5. Click the Close button in the dialog window.
6. After the tests are run, use the menu option File -> Reload Data to refresh the data. Verify that the audit trail data from the database is loaded in the visualization file.
7. Select File -> Exit to close the Spotfire Professional.

Chapter 8 Advanced Features

This chapter describes some advanced functionality of TIBCO ActiveMatrix Service Gateway .

Topics

Cache Agent

The cache agent stores cache data and are responsible for object management. A processing unit can have one cache agent only.

Using cache clustering technology, object data is kept in memory caches, with redundant storage of each object for reliability and high availability. Cache data is shared across all the gateway engines participating in the cluster. The purpose of cache agents is to store and serve cache data for the cluster.

The cache agents are used to implement the association cache and response cache functionality. The size of a cache can be unlimited or limited. Performance is best when all the data is in cache.

Only use an unlimited cache if you deploy enough cache agents to handle the data. Otherwise out of memory errors may occur.

How to run Cache Agent

You can start the cache agent as follows:

1. Open a terminal window.
2. Navigate to *ASG_HOME*/bin directory.
3. Type the following command to start up the cache agent:

```
./asg-engine -u  asg-cache
```

Following command starts up the gateway core engine in cache agent enabled mode:

```
./asg-engine -u asg-caching-core
```

Hot Deployment Overview

You can make certain changes to a set of TIBCO ActiveMatrix Service Gateway configuration, without having to shut down the gateway engine. This is known as hot deployment.

Hot deployment process suspends north inbound channels and waits for the time delay specified before hot deploying the configuration.

Due to a potential to cause an unexpected result to the transaction in process, it is strongly recommended that the hot deployment is used only in a development and testing environment.

Enabling Hot Deployment

By default, the hot deployment is not enabled. To enable the hot deployment feature, perform the following steps:

1. Navigate to the *ASG_CONFIG_HOME*/`asg` directory.
2. Edit the `asg.properties` file to change the value of the following property to **`true`**:

   ```
   tibco.clientVar.ASG/Deployments/AllowHotUpdate=true
   ```

3. Deploy and restart the gateway engines that need to be hot deployed engines. For example, asg_core or asg_caching_core.

Invoking Hot Deployment

Hot deployment can be invoked from an MBean client by invoking the **`refreshConfig`** method in `com.tibco.asg domain` for the appropriate gateway engine.

`refreshConfig` method uses following parameters:

- The first parameter is the location of the configuration directory used by the gateway engine.
- The second parameter is the delay in milliseconds. If the delay value is -1, the gateway engine does not shutdown on any error during hot deployment. Otherwise, the gateway engine shuts down on error.

Extension Mechanism

Extension mechanism capability allows you to add custom stages in the default transaction processing pipeline. The custom stage is developed using the rules language within TIBCO ActiveMatrix Service Gateway Studio.

Following features are implemented using the extension mechanism:

Association Cache

The gateway engine provides a mechanism to cache the previous acquired information retrieved from the external systems during the lookups. It uses that information later to optimize the time taken for routing of the requests.

Response Cache

The gateway engine has the capability to store the responses of requests in the cache clusters. It uses these responses for later requests. Response cache is implemented using association caches. This functionality allows the gateway engine to process the requests faster and also off load the service endpoints.

Sequential Orchestration

TIBCO ActiveMatrix Service Gateway supports sequential orchestration. Sequential orchestration allows you to access multiple service endpoints by making a number of sequential calls to fulfill or authorize a request. With sequential orchestration, there is effectively a single outbound service invocation, preceded by one or more sidebound service invocations.

Sequential orchestration may use the associative and responses cache features to accelerate the processing of future requests, which helps to minimize the load on back-end systems.

Sequential orchestration allows you to access the external systems when one or more service requests are pipelined.

Field Translation

The gateway engine has the ability to call the external services to perform the translations of certain fields required for the main request processing. For example, if the requestor sends an product id in the request but the backend service requires the product name, then the gateway engine calls a east side service to translate the product id into product name. This translated value (product name in this case) is replaced in the main request payload and is used to invoke the back end service.

This cross referencing for lookups and data enrichments can use the association cache functionality for the faster processing of requests.

Content based Authorization

TIBCO ActiveMatrix Service Gateway supports the partner authorization based on the content of the incoming request message. This functionality allows the partners to authenticate the references of the partner (for example, customers of the partners) which are sent in the content of the message.

For content based authorization, you have a single request which contains one or more customer references. TIBCO ActiveMatrix Service Gateway supports the authorization of each of the request individually by parsing the content of the request message.

Content based authorization uses Extension Mechanism capability which allows you to use the association cache functionality.

LDAP Based Authorization for Partner References

TIBCO ActiveMatrix Service Gateway allows you to authorize the partner's references in the content of the request message with a LDAP system. TIBCO ActiveMatrix Service Gateway provides a set of catalog and custom rule functions to support this functionality.

Appendix A Configuration Tasks

This appendix explains few configuration tasks required for some functionality of the product.

Topics

Enable detail level logging for Gateway

This section explains the steps required to enable the detail level logging for basic authentication.

1. Open a terminal window.
2. Navigate to *ASG_CONFIG_HOME*/config directory.
3. Edit the **`asg.properties`** file.
4. Search the following property in the file:

   `tibco.clientVar.ASG/Logging/MinLogLevel=1`
5. Set the highest log level by changing the value of the property as below:

   `tibco.clientVar.ASG/Logging/MinLogLevel=0`
6. Search the following property to enable the detail logging for common logger logging by:

   `tibco.clientVar.ASG/Logging/clLogLevel=0`
7. Enable detailed logging by changing the value of the property as below:

   `tibco.clientVar.ASG/Logging/clLogLevel=1`

Configure TIBCO Enterprise Message Service

This section describes the steps to setup and configure TIBCO Enterprise Messaging services destination required for TIBCO ActiveMatrix Service Gateway runtime.

This section assumes that TIBCO ActiveMatrix Service Gateway has access to a running instance of TIBCO Enterprise Messaging services.

1. Make sure TIBCO EMS server is running.
2. Create the following queues in the TIBCO EMS server. For example using **`tibemsadmin`** tool:

```
create queue asg.out.request
create queue asg.out.request.reply.0.0
create queue asg.out.request.reply.0.1
create queue asg.out.request.reply.0.2
create queue asg.in.request
create queue asg.in.request.reply.0
create queue asg.custom.request
create queue asg.custom.request.reply.0
```

The default configuration assumes that TIBCO Enterprise Message Service is running on the localhost and port 7222. Depending on the host and port where TIBCO Enterprise Message Service runs, you can edit the TIBCO Enterprise Message Service configuration parameters in the ASG_CONFIG_HOME/**`config/asg.properties`** file.

For example, the steps are listed using the TIBCO Enterprise Message Service **`tibemsadmin`** tool:

1. Navigate to the TIBCO Enterprise Message Service installation directory.
2. Change to the *EMS_HOME*/**bin** directory.
3. Start the **`tibemsadmin`** tool, as below:

```
tibemsadmin server "tcp://host:port" user adminuser
```

4. Type the following command to create queues:

```
create queue asg.in.request
create queue asg.in.request.reply.0
create queue asg.out.request
create queue asg.out.request.reply.0.0
create queue asg.out.request.reply.0.1
create queue asg.out.request.reply.0.2
create queue asg.custom.request
```

```
create queue asg.custom.request.reply.0
```

5. Exit the **tibemsadmin** utility.

Setup and Run Apache HTTP Server

This section describes the steps required to set up the Apache HTTP server.

Installing Apache HTTP Server

1. Download Apache httpd server from **`http://httpd.apache.org`**. Refer to the TIBCO ActiveMatrix Service Gateway software readme for Apache server version requirements.
2. Compile the source and install it. Refer to the Apache httpd server documentation for complete installation details. You may download the binary installation for your distribution.

Configuring Apache HTTP Server for TIBCO ActiveMatrix Service Gateway

1. Open the *Apache_HOME/conf/httpd.conf* file for editing.
2. Add the following line in the file:

   ```
   Include ASG_HOME/modules/http_server/apache/mod_ASG.conf
   ```

Replace the ASG_HOME to the directory location where TIBCO ActiveMatrix Service Gateway product is installed.

3. Modify the listening port, as required:

   ```
   Listen 8080
   ```

Edit the listen port only if you run the Apache Server as a normal user. If you run it as a super user, change in port is not required.

4. Verify that you have setup the system environment variables.

Running Apache HTTP Server

1. Navigate to directory:

   Apache_HOME/bin
2. Run the command:

   ```
   ./apachectl start
   ```

Configure Apache web server for Basic Authentication

This section explains the steps required to configure the Apache web server for basic authentication.

Enable Basic Authentication on the Apache web-server

1. Open a terminal window.
2. Navigate to *ASG_HOME*/modules/http_server/apache directory.
3. Edit the **`mod_ASG.conf`** file.
4. Search the following section in the file:

```
<Location/>
    SetHandler asg_rv_inbound_handler
    AsgSubject _LOCAL.asg.north.request
    AsgTimeout 30
</Location>
```

5. Insert the configuration for a new location as below:

```
    <Location /asg/ba>
    AuthType Basic
    AuthName "ASG"
    # (Following line optional, file is default)
    AuthBasicProvider file
    AuthUserFile /home/asg/apache/htpasswd/htpasswords
    Require validuser
    SetHandler asg_rv_inbound_handler
    AsgSubject _LOCAL.asg.north.request
    AsgTimeout 30
    </Location>
```

The change will look as :

```
    <Location /asg/ba>
    AuthType Basic
```

AuthName "ASG"

(Following line optional, file is default)

AuthBasicProvider file

AuthUserFile /home/asg/apache/htpasswd/htpasswords

Require validuser

SetHandler asg_rv_inbound_handler

AsgSubject _LOCAL.asg.north.request

AsgTimeout 30

</Location>

<Location / >

SetHandler asg_rv_inbound_handler

AsgSubject _LOCAL.asg.north.request

AsgTimeout 30

</Location>

This change enforces the user access with basic authentication

6. Save the **mod_ASG.conf** file.

Create a password file for Apache Basic Authentication

1. Open a terminal window.
2. Navigate to *APACHE_HOME*
3. Create a **htpasswd** sub-directory to store the password file.
4. Navigate to the *APACHE_HOME*/bin directory.
5. Create a partner identity using the Apache **htpasswd** utility for the user asgpartner01 with password asgpartner01, shown as below:

```
./htpasswd APACHE_HOME/htpasswd/htpasswords asgpartner01
 New Password:asgpartner01
 Enter New Password:asgpartner01
```

6. Create a second partner identity using the Apache **htpasswd** utility for the user asgpartner02 with password asgpartner02, shown as below:

```
./htpasswd APACHE_HOME/htpasswd/htpasswords asgpartner02
 New Password:asgpartner02
```

```
Enter New Password:asgpartner02
```

Reload the configuration file for the Apache web-server

1. Open a terminal window.
2. Navigate to *APACHE_HOME*/bin directory.
3. Restart the Apache HTTP server to reload the updated configuration:

   ```
   ./apachectl restart
   ```

Configuring JMS northbound transport for XML

This section describes the steps to configure the JMS transport on the northbound side for XML message.

TIBCO Designer Configuration

In this scenario, TIBCO BusinessWorks is used as a client to send the XML payload to the TIBCO ActiveMatrix Service Gateway engine using the JMS transport.

1. Open the TIBCO Designer and create a new project.
2. Create a new `JMS Application Properties` activity resource. Type a name for this resource (for example, JMSProperty1) Add a new property for this resource as follows:
   - PropertyName: Operation
   - Type: string
   - Cardinality: required
3. Create a new `Process Definition`. Type a name for this process (for example, SendJMSMessage).
4. Create a new activity `JMS Queue Requestor` resource in the process. Type a name for this resource (for example, SendJMSRequest). Configure the `JMS Queue Requestor` resource as follows:
   - Configure the `JMS Connection` parameter. Select a pre-configured JMS connection resource.
   - Click the `Advanced` tab. For the `JMS Application Properties` field, select the configured `JMS Application Properties` resource (for example, JMSProperty1). Click Apply.
   - Click the `Input` tab. Verify that the **`Operation`** field appears under `OtherProperties`. Specify a value for the `Operation`. This value of the operation field matches the **`SOAP Action`** parameter of the operation configuration in the `Operations` tab of the gateway configuration UI.
   - Click `Apply` and `Save` the project.

Operations Configuration in Gateway UI

1. Start the gateway UI.
2. Click **`Operations`** tab. Configure a new operation as below:

- Operation Name: Any logical name. For example, getLocationBW.
- SOAP Action: Specify the value of this parameter to prepend `/ESB` followed by the value of Operation, a JMS application property.

  For example, if the value of Operation (JMS application property) is specified as **`getLocation`**, type the value of SOAP Action parameter as **`/ESB/getLocation`**. Operation is identified by Operation/SOAP Action.

3. Click **`Save`** to save the configuration.

Enable the ESB Channels in the CDD file

By default, the ESB channels are disabled. To enable the channels, perform the following steps:

1. Open the *ASG_HOME*/bin/**`asg_core.cdd`** file for editing in a text editor.
2. Search the following property.

   <property-group comment="" name="Channel">

   `<property name="be.channel.deactivate"`
   value="/DefaultImplementation/Channels/SouthboundEsb0Channel,/DefaultImplementation/Channels/SouthboundEsb1Channel,/DefaultImplementation/Channels/SouthboundEsb2Channel,/DefaultImplementation/Channels/North_ESBChannel,/DefaultImplementation/Channels/North_HTTPChannel"/>
3. To enable the channels, rename the property name to `be.channel.deactivate.test` as below:

   <property-group comment="" name="Channel">

   `<property name="be.channel.deactivate.test"`
   value="/DefaultImplementation/Channels/SouthboundEsb0Channel,/DefaultImplementation/Channels/SouthboundEsb1Channel,/DefaultImplementation/Channels/SouthboundEsb2Channel,/DefaultImplementation/Channels/North_ESBChannel,/DefaultImplementation/Channels/North_HTTPChannel"/>
4. Save the file.

Update TIBCO Enterprise Message Service Libraries

To include the TIBCO Enterprise Message Service libraries, perform **`only one`** of the following steps:

- Open the *ASG_HOME*/bin/**`asg-engine.tra`** file for editing. Set the `tibco.env.EMS_HOME` property to the TIBCO Enterprise Message Service installation home.

For example,

tibco.env.EMS_HOME=/home/asg/tibcoasg/ems/5.1

- Copy the **`jms.jar`** and **`tibjms.jar`** files from TIBCO Enterprise Message Service installation (*EMS_HOME*/lib) to *ASG_HOME*/**`lib/ext/tpcl`**

Changing the Stack Size

To address the high memory usage requirements of TIBCO ActiveMatrix Service Gateway server, it is recommended to set a smaller stack size, for example 256 KB. This can be done in the following ways

You must have super user privileges to change the stack size.

To Permanently Change the Stack Size

1. Open the following file for editing:

   `/etc/security/limits.conf`

2. Add the following lines (example values shown):

```
*    soft stacksize 256
*    hard stacksize 256
```

Refer to your Operating System's manual for details and set the appropriate values.

To Temporarily Change the Stack Size

You can change the stack size for the current session at the command line. To do so type a command like the following at a command prompt:

```
ulimit -s 256
```

The above command sets the stack size to 256 KB for the current session only.

Modify Unicast Discovery URL in CDD file

When TIBCO ActiveMatrix Service Gateway server configuration is manually copied to any server from where it was installed and configured, it requires to edit the discover URL and listen URL.

Discover URL

The discover URL specifies how the gateway engine (node) listens for discovery requests from nodes attempting to join the cluster. When a cluster starts up, and also when new members join a cluster, a discovery process enables the members to discover each other. The discover URL specifies how an gateway engine (node) listens for discovery requests from nodes attempting to join the cluster.

The discovery URL for well-known address configuration uses the following format:

```
tcp://ip:port[;ip:port]*
```

After the discovery is complete, the members communicate internally using a listen URL.

Listen URL

The listen URL is used for direct communication between the members of the metaspace. The listen URL value must be different for each cluster member.

The listen URL uses this format:

`tcp://`*interface*`:`*port*`[-`*EndPort*` |*]/`

The cluster member binds to the specified interface and the specified port when creating the TCP socket. Specify the parameters as follows.

Parameter	Notes
interface	To specify a value, use the desired IP address. The value for *interface* must be the same in both the discovery and the listen URLs for a node. If there are multiple interfaces on one machine, specify the interface you want to use and do not rely on the default value. The default value for *interface* is the first available interface provided by the operating system for the machine.

Parameter	Notes
port	To specify a single port use the port number in the listen URL, as shown in this example: `tcp://interface:6000/` The default value is the first available port in the 50000+ range.

How to edit the Discover URL and Listen URL

To edit the discover URL and listen URL, perform the following steps:

1. Open the *ASG_HOME*/bin/**`asg_core.cdd`** file for editing.
2. Edit the following properties and set the value to the **actual IP address** of the machine, and an unused port.

For example:

```
<property name="be.engine.cluster.as.discover.url"
value="tcp://127.0.0.1:6000/"/>

<property name="be.engine.cluster.as.listen.url"
value="tcp://127.0.0.1:6000-*/"/>
```

3. Save the file.

Configure JMX Properties in Gateway Engine TRA File

After the gateway engines are started, and join the cluster, they use JMX MBeans to expose monitoring and management information to the MM server, and to allow remote method invocation. The JMX port number must be specified before the engine's JVM starts. A variable for the port number is provided in the TRA file so that the actual value can be specified before the engine starts.

To Configure JMX Properties

JMX properties are provided in the shipped ASG_HOME/bin/asg-engine.tra file but are commented:

```
#java.property.be.engine.jmx.connector.port=%jmx_port%
#java.property.be.engine.jmx.connector.authenticate=false
```

To Enable Monitoring and Management

To expose JMX for monitoring and management (without authentication), uncomment this property:

```
java.property.be.engine.jmx.connector.port=%jmx_port%
```

Ensure that the value of the port property is set to this literal value: `%jmx_port%`. The actual value is substituted at runtime. When you use the MM console to start the gateway engines remotely, MM reads the port number from the PU configuration settings in the site topology file.

When more than one PU (engine) is deployed to the same host, ensure that a different JMX port is used for each of the PUs, in the site topology file.

To Enable JMX MBeans Authentication

The following property enables authentication:

```
java.property.be.engine.jmx.connector.authenticate=true
```

Site Topology Overview

The site topology file contains deployment time information such as which processing units to deploy to specific hosts in your environment. You need to know information about the computers that will host the agents you plan to deploy, for example the operating system and IP address.

The MM server uses the SSH software as the remote invocation software to start remote processes on unix machines.

- Changes to the EAR file do not affect the topology configuration. However if the cluster, processing unit, or agent definitions in the CDD file change, you must recreate the site topology file using the updated CDD.
- If you change the site topology, you must restart the MM server.

1. Configure Cluster Properties

2. Add Deployment Units (DUs) Add DUs as needed. For each DU, specify the following:

- The deployment location of the CDD and EAR files. MM copies the files to the specified location at deploy time.
- One or more *processing unit configurations* (PUCs). You'll configure the PUCs in the next step.

3. Add Processing Unit Configurations (PUCs) to DUs For each PUC, select one processing unit (PU) from the list of PUs defined in the CDD file. Set deploytime properties such as the JMX ports used by MM to communicate with the deployed engine.

4. Add Hosts Here you specify the host configuration, including the software used on the remote hosts to start remote gateway engines (processes). You must map the DU's to the hosts where you want to deploy them. Multiple hosts can reuse the same deployment unit configuration.

In the topology file, you reference two locations for the CDD and EAR files. The files in each location must be *exact copies*:

- Master CDD and EAR files: You specify the location of the master CDD and EAR files. These copies must be manually copied to the specified location on the MM server host, for use in deployment.
- Deployed CDD and EAR files: In the deployment unit settings, you specify where MM will place the CDD and EAR files when it performs deployment to a remote host.

The project and master CDD can have the same location if you run the gateway engines and the MM server in the same host.These two sets of fields are available in case you are configuring the topology on a different machine from the MM server machine.

- All locations specified must already exist. The software does not create directories.
- Use the correct path for the operating system of the host (For example, linux versus windows).
- **Limitation: One CDD and EAR file per Cluster Machine**: Currently deployment is at the Machine level and each machine can have only one copy of the deployed CDD and EAR files. If you specify multiple DUs for the same host, problems may occur because CDD and EAR files are copied only to the location specified for the first DU.

Deployment-Specific Processing Units

In general, you can reference one processing unit multiple times to create different processing unit configurations (PUCs). However processing units that have deployment-specific settings cannot be used in this flexible manner.

Agent-Instance-Specific Properties

If a processing unit contains agent-instance-specific properties (such as agent key and priority settings), you must use it in only one PUC, which is used in only one DU, that is itself used only once in the deployment.

Host-Specific Processing Units

Processing units can host-specific settings. If a deployment unit contains a PUC that references such a processing unit, you must link it only to the appropriate host for deployment.

Site Topology Reference

Site Settings

Table 25 Site Topology — Site Settings

Property	Notes
Site Name	Site name. Default value is the name of the site topology file.
Description	Description of the site, as desired.

Cluster Settings

Table 26 Site Topology — Cluster Settings

Property	Notes
Cluster Name	Read-only field displaying the cluster name specified in the CDD. This name is set in the Cluster Name field of the CDD editor.
Project CDD	Location and name of project CDD. This is the location used by TIBCO ActiveMatrix Service Gateway Studio for configuration of the site topology.
Master CDD	Location and name of the master CDD. This is the location used by the MM server.
Master EAR	Location and name of the master EAR. This is the location used by the MM server.

Deployment Unit Settings

Table 27 Site Topology — Deployment Unit Settings

Property	Notes
Deployment Unit Name	Name of the deployment unit. It can be helpful to include the operating system of the host to which you will deploy this DU in the DU name. If a DU contains any host-specific settings, it's also a good idea to put the host name in the DU name. **Note** Paths in different operating systems are specified using different tokens. Even if the DUs are identical in all other respects, you must create different DUs for different operating systems. Default value is DU_*n* where n is a number that increments each time you add a DU to the diagram.
Deployed CDD	Absolute file path to the location where the MM server will deploy the copy of the master CDD used by this DU.
Deployed EAR	Absolute file path to the location where the MM server will deploy the copy of the master EAR used by this DU.
Processing Unit Configurations	Displays a list of processing unit configurations.

Processing Unit Settings

Table 28 Site Topology — Processing Unit Settings

Property	Notes
Processing Unit Configuration Name	The name that identifies this configuration of the processing unit, as specified in the Processing Unit setting (see below). The processing units settings are configured in the CDD.

Table 28 Site Topology — Processing Unit Settings (Cont'd)

Property	Notes
Use As Engine Name	Check this checkbox to use the value of the Processing Unit Configuration Name field as the engine name. It is recommended that you use the same choice across all processing units in the cluster.
Processing Unit	Select the processing unit you want to use. Only processing units configured in the CDD selected as the Project CDD appear in the list. The same processing unit can be used in multiple PUCs.
Number of Agents	Displays the number of agents in the selected processing unit. (Not present in the site topology XML file.)
JMX Port	JMX port used by MM to perform monitoring and management. Required. When more than one PU is deployed on the same host (in one DU or multiple DUs), you must ensure the JMX port in each of these PUs is different.

Host Settings

Table 29 Site Topology — Host Settings

Property	Notes
General Settings	
Host Name	Machine name of the host (including the domain extension). Used for remote access. Used to identify the host in the MM user interface. Required. To validate the hostname, ping the host using this name from the MM server machine. **Note**: Specify the exact name of the host. Errors in the host name result in the host appearing in the MM Console UI as an unpredefined machine. Do not, for example, use `localhost`.
IP	IP address of the host machine. Used for remote access. Required. You can use 127.0.0.1 (`localhost` loop back IP address) for engines running on the same machine as the MM server.
User Name	User name to log onto the host machine. The user credentials are used for remote deployment and execution, including starting a process unit. At runtime, a dialog box pops up to authenticate the user, for example when deploying a PU. If you provide a username and password here, then the dialog is prepopulated with these values. You can enter different values as needed. If you do not provide the credentials here, then you must provide them at the pop-up dialog. You can specify a local user or a domain user. Enter details for the user you specified for the remote connection utility you are using. For example, if you use SSH utility, you specify the username you use for SSH utility.

Table 29 Site Topology — Host Settings (Cont'd)

Property	Notes
Password	Password of the user referenced in the User Name field. The password is encrypted. See notes in User Name section.
Operating System	Operating system of the host machine. See the product readme for a list of supported platforms.

Working with Cluster Explorer

Active and inactive nodes are shown in Cluster Explorer for a quick view of system health. Using Cluster Explorer, you can use functionality available at various nodes on the left, and you can view information about that node level on the right.

The Cluster Explorer shows the hierarchy of cluster members. Inactive agents (which could be standby agents or failed agents) are dimmed.

The structure of the cluster member hierarchy is as follows:

```
Site
   Cluster
      Machine (host name)
         Process (Processing Unit or Deployment Unit or JVM process ID)
            Agent (mm agent)
      Cache Objects
```

Where:

- Site is the root.
- Cluster shows the name of the cluster being monitored.
- Machine shows one or more machines within the cluster. They run the cluster processes (process units or engines).
- Process shows each of the JVM processes (gateway engines) running on a machine. The label for a process that was predefined in the topology file is the process unit ID assigned in the file, concatenated with the process ID enclosed in parentheses. The label for an unpredefined process is the JVM process ID.
- Agent lists all agents of each type running in the JVM process.
- The Cache Objects panel shows all the objects stored in the cache, without regard to their physical location in the TIBCO ActiveMatrix Service Gateway cluster.
- Machines, TIBCO ActiveMatrix Service Gateway engines, and agents are all *members* of the TIBCO ActiveMatrix Service Gateway cluster.

Navigating Cluster Explorer

To navigate the cluster explorer, do any of the following:

- Expand Cluster Explorer and select the member you want to work with or whose metrics you want to see. Metrics display on the right.
- Click an inactive cluster member to display the last available health metrics for that member.

- Click the minimize button in the Cluster Explorer title bar to minimize the explorer panel.

Starting, Stopping, Pausing, and Resuming Gateway Engines

To Start, Stop, Pause, or Resume an Engine

1. From Cluster Explorer, select the engine you want to start, stop, pause, or resume. (You resume a paused engine.)
2. Click the appropriate icon: **Start, Stop, Pause, or Resume**
3. Verify the login details and click **OK**.

Glossary

A

Always do this

In order to implement pop-up glossary entries in HTML, glossaries can contain only paragraphs like this, and letters dividing the entries alphabetically.

ActiveEnterprise

One of the three product groupings within TIBCO Software Inc, focused on enterprise application integration. The other two are ActivePortal™ and ActiveExchange™ (see www.tibco.com for details).

alert

A notification to an end-user, for example, scheduled alerts deliver portal headlines to a chosen device. See also *real-time alert*.

C

common page

A logical name-value binding to identify a portal page in page layout templates, regardless of where that portal page exists in the page tree. Provides for flexibility and ease of template maintenance.

community site

A type of portal *site* created and used by business users in support of team communications.

D

deployment descriptor

An XML file that describes the configuration of a web application. It's located in the WEB-INF directory of the application's WAR file.

DMZ

An acronym for demilitarized zone, this term is used metaphorically for that part of a network between an inner and an outer fire wall. Machines placed in the DMZ may be available to authorized users outside the firewalls, whereas machines placed behind the DMZ are protected from outside access.

domain

In TIBCO Administrator, two kinds of domains are used. See also *administration domain* and *application domain*.

S

SOAP (Simple Object Access Protocol)

A basic web services standard for making web services available remotely. See www.w3C.org/TR/SOAP/for details. See also *UDDI*, *WSIL*, *WSDL*, *WSRP*

Index