Configuring BAM

Version 9.6

April 2014

This document applies to webMethods Central Configuration Version 9.6 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: OPT-CC-CDG-96-20140509

Table of Contents

About this Guide

This guide describes how to configure and deploy webMethods Business Activity Monitoring (BAM) component configurations using the webMethods Central Configuration tool. It contains information for administrators who want to initialize or edit common configuration settings and deploy them out to one or more webMethods environments.

To use this guide effectively, you should be familiar with My webMethods Server and the webMethods Optimize components you want to manage.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service* .
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Documentation Installation

You can download the product documentation using the Software AG Installer. The documentation is downloaded to a central directory named _documentation in the main installation directory (SoftwareAG by default).

Online Information

You can find additional information about Software AG products at the locations listed below.

If you want to...	Go to...
Access the latest version of product documentation.	**Software AG Documentation website** http://documentation.softwareag.com
Find information about product releases and tools that you can use to resolve problems. See the Knowledge Center to: ■ Read technical articles and papers. ■ Download fixes and service packs (9.0 SP1 and earlier). ■ Learn about critical alerts. See the Products area to: ■ Download products. ■ Download certified samples. ■ Get information about product availability. ■ Access older versions of product documentation.	**Empower Product Support website** https://empower.softwareag.com

If you want to...	Go to...
■ Submit feature/enhancement requests.	
■ Access additional articles, demos, and tutorials. ■ Obtain technical information, useful resources, and online discussion forums, moderated by Software AG professionals, to help you do more with Software AG technology. ■ Use the online discussion forums to exchange best practices and chat with other experts. ■ Expand your knowledge about product documentation, code samples, articles, online seminars, and tutorials. ■ Link to external websites that discuss open standards and many web technology topics. ■ See how other customers are streamlining their operations with technology from Software AG.	**Software AG Developer Community for webMethods** http://communities.softwareag.com/

1 Concepts

Overview

This chapter describes the concepts behind the webMethods Define Environments page, sometimes referred to as the webMethods Central Configuration tool, that is used to configure an Optimize BAM system. It describes the architecture of the webMethods BAM configuration implementation, and the process by which configuration instances are defined and deployed out to a webMethods Optimize environment. This chapter also describes the interaction between the configuration subcomponents and webMethods Optimize components.

Configuring BAM Using Define Environments

The My webMethods Define Environments page allows you, as an administrator, to initialize or edit multiple Optimize component configurations in real-time from one central user interface.

Use the Define Environments page to complete the following tasks:

- Identify and define a collection of webMethods product component resources with common configurations to manage as a group, or *environment*.
- Initialize, edit, and deploy Optimize component configurations from a single point of access.
- Access and manage database pool definitions for Optimize product components.
- View system information according to each webMethods product component environment.

BAM Configuration Components

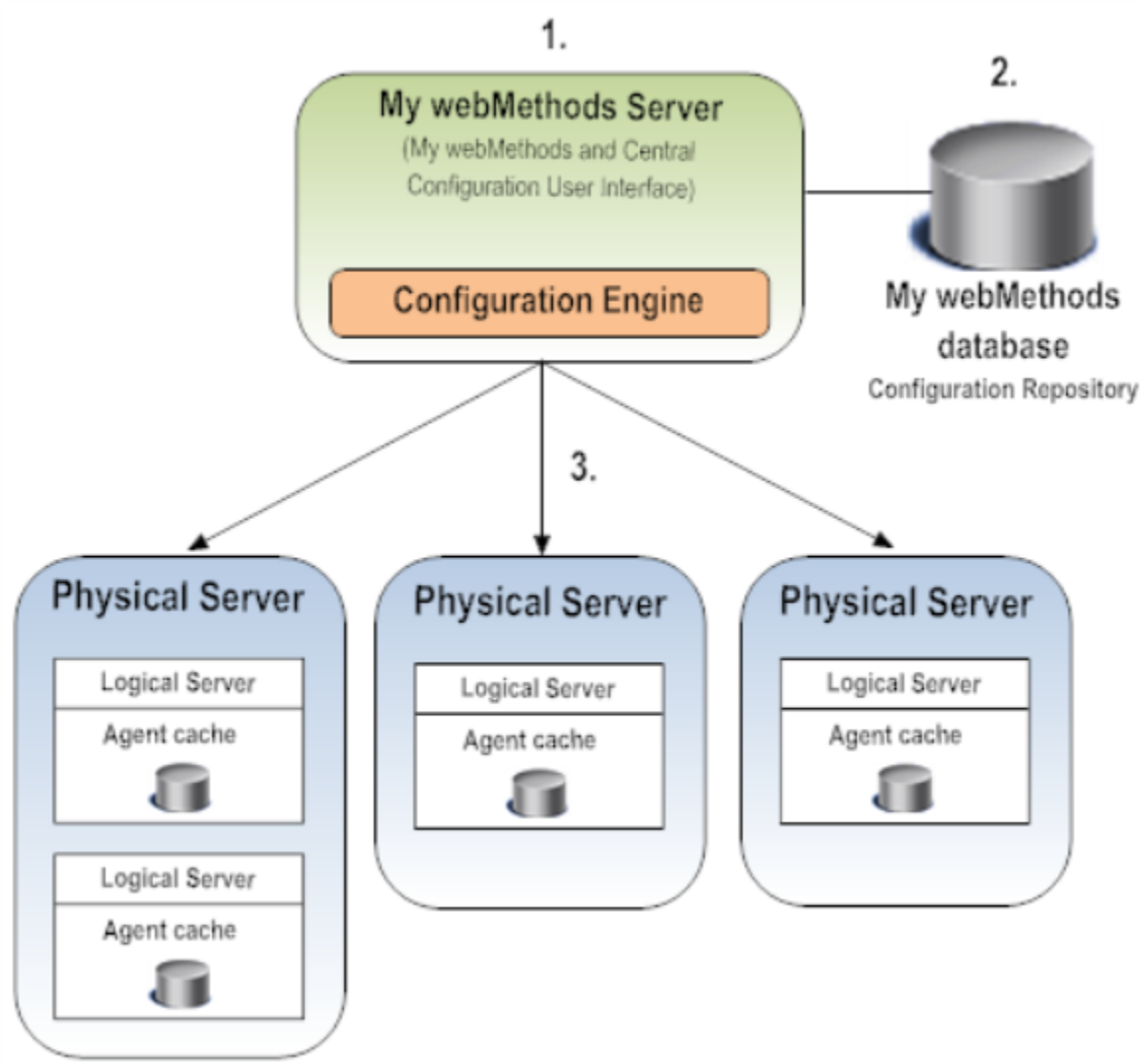
The defined Environments page is included as part of a standard My webMethods Server installation. There are four main BAM configuration components:

- **Configuration engine.**The configuration engine resides on the primary My webMethods Server host. You access, manage, and deploy webMethods component configuration information through the My webMethods user interface for the configuration engine.
- **Configuration repository.** The configuration engine stores configuration data in the configuration repository. The tables for the configuration repository reside in the primary My webMethods Server host database.
- **Configuration agent.** Each webMethods Optimize component, or *logical server*, contains a configuration agent. Configuration agents receive configuration data from the configuration engine through Web services.

After a configuration file is received by the configuration agent, the file is stored locally on the host file system of the logical server. All access to a configuration file is controlled through the configuration agent.

- BAM configuration **user interface.** During the installation process, the Define Environments page is installed and deployed on the My webMethods Server and become part of the My webMethods user interface. This page enables you to define environments and to view, configure, and deploy configuration information on the configuration engine.

Central Configuration Overview



Step	Description
1.	An environment configuration is defined and deployed using the Define Environments page in My webMethods. When the configuration is

Step	Description
	deployed, the configuration engine reads data from the configuration repository and sends that data to the remote configuration agents, where the data is written to local configuration files.
2.	The configuration files reside in the configuration repository and are deployed out to the various components.
3.	The configuration agents on the logical servers receive and cache the configuration files. A new configuration is implemented automatically after the initial deployment. Note that subsequent deployments require the logical servers in the environment to be restarted.

Logical Servers

Logical servers represent Optimize components. Optimize may contain one or more components that provide a specific function or process. For example, the Analytic Engine is an Optimize component, and its primary function is to analyze business, system, and process data.

During the environment configuration process, each logical server is mapped to one or more physical servers. Mapping logical servers to physical servers allows processes to be shared where physical infrastructure differs. For example, you could run two or more Analytic Engine processes on the same physical server. Each logical server also might have network endpoints and local configurations specified. When you deploy a configuration instance, the My webMethods Define Environments page uses the logical-to-physical server mapping to know where to deploy all of the defined configuration files to the product component.

Each logical server contains subcomponents that also can be configured using the My webMethods Define Environments page.

Logical Server Subcomponents

Each logical server contains a set of configurable subcomponents. The My webMethods BAM configuration functionality supports the webMethods components and subcomponents that are listed in the following table.

webMethods Product	Supported Logical Servers	Supported Logical Server Subcomponents
Optimize	Analytic Engine	The following Analytic Engine subcomponents are configurable through the My webMethods Define environments page: ■ Analytic Engine settings ■ Cache configuration ■ Data maintenance settings ■ JNDI configuration ■ Journal logging ■ Mail settings ■ Monitor behavior settings ■ Process Tracker settings ■ SNMP alert settings ■ Station settings ■ WS action settings
	Web Service Data Collector	The following Data Collection Service subcomponents are configurable through the My webMethods Define environments page: ■ Data collector settings ■ JNDI configuration ■ Journal logging
	Infrastructure Data Collector	The following Infrastructure Data Collector subcomponents are configurable through the My webMethods Define environments page: ■ Adabas SOA Gateway resource module settings ■ Broker resource module settings ■ Collector settings ■ Com-plete Resource module settings ■ EntireX resource module settings ■ ETS resource module settings

webMethods Product	Supported Logical Servers	Supported Logical Server Subcomponents
		■ Integration Server resource module settings ■ JNDI configuration
Integration Server	Informational only. Integration Servers provide an endpoint connection to the Analytic Engine.	
JMS Server	Informational only. The JMS Server provide an endpoint connection to the Analytic Engine.	
My webMethods Server	Informational only. My webMethods Server provides an endpoint connection to the Broker Server.	

Environments

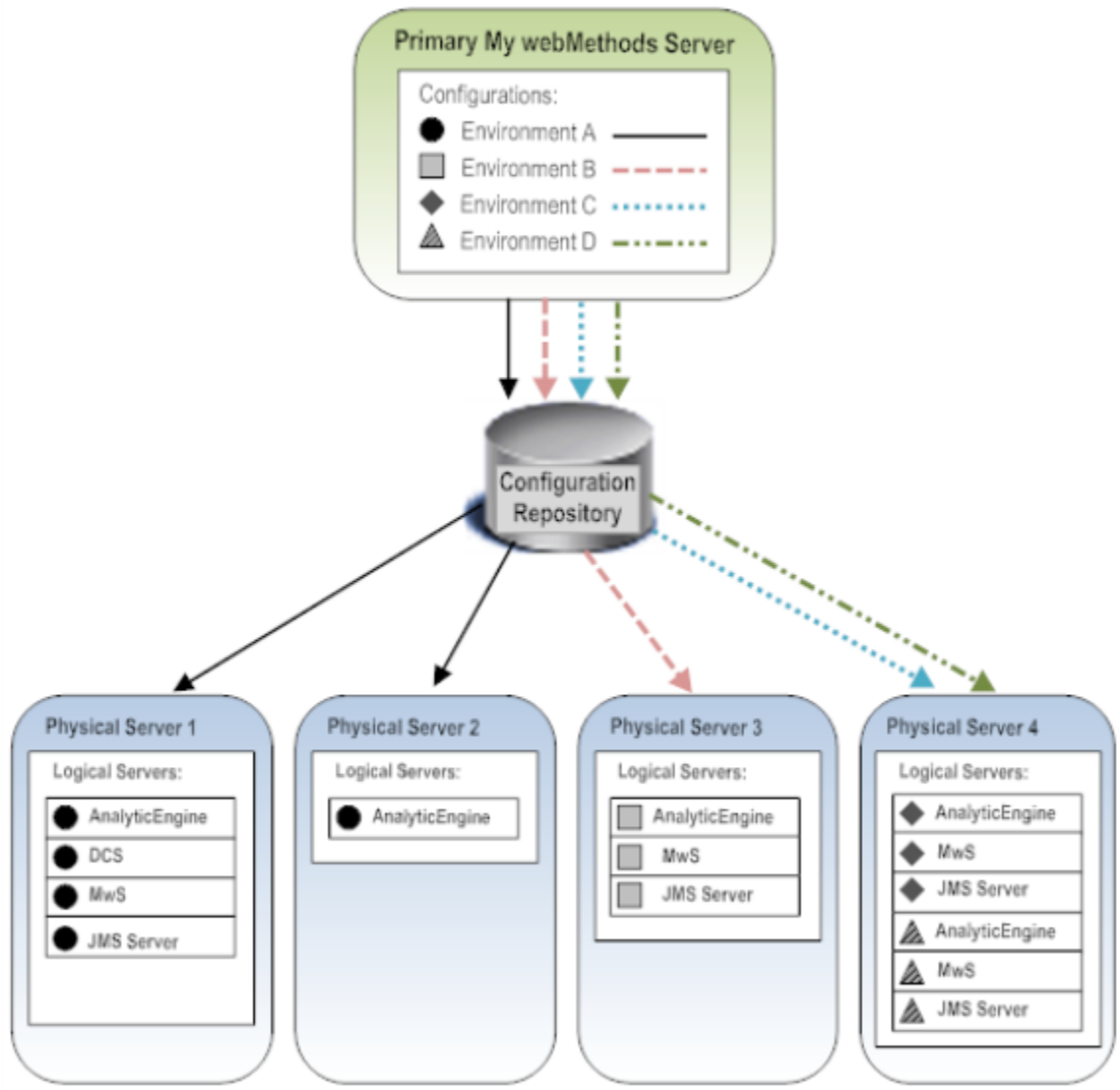
To link together several webMethods product components that you want to manage as a group, you use the My webMethods Define environments page to define a webMethods environment. An *environment* is a group of webMethods product components that share configuration settings, such as cache settings.

As shown in the diagram below, an environment may include one or more logical servers. The diagram shows four environments that are configured as described in the following list:

- Environment **A** is a full Optimize implementation containing six logical servers that reside on two different physical servers.
- Environment **B** is a minimum Optimize implementation consisting of three logical servers that reside on one physical server.
- Environments **C** and **D** have three logical servers each, with both environments residing on a single physical server.

Note that the diagram shows that a logical server can belong to only one environment at a time.

Example Environment Implementation



An environment also includes a set of configuration files for each logical server. You use the My webMethods Define environments page to define an environment and to define the configuration files. To learn more about the My webMethods Define Environments page and how to define and configure an environment, see "Getting Started" on page 19.

2 Getting Started

Overview

This chapter describes how to use the My webMethods Define Environments page to configure and deploy an Optimize BAM environment. It provides an overview of how to use the Define Environments tabs and briefly describes the steps that you must follow to successfully deploy an Optimize configuration.

Installing the Define Environments page

The Define Environments page is installed with the My webMethods installation using the webMethods installation program. See *Installing webMethods and Intelligent Business Operations Products* for detailed information about installing My webMethods.

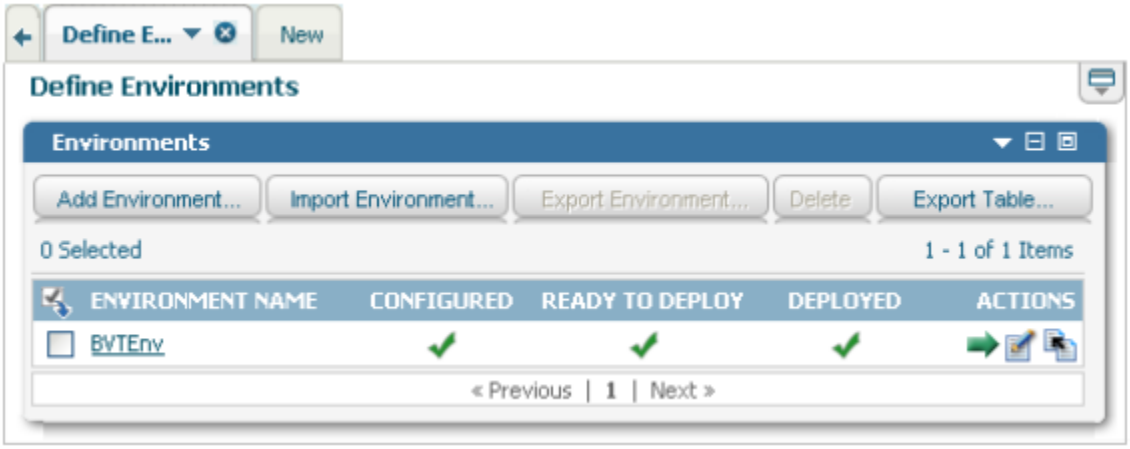
Using the Define Environments page in My webMethods

The My webMethodsDefine Environments page contains a set of configuration tabs. These configuration tabs take you through the following steps:

- Defining database pool configurations
- Adding an environment
- Defining configuration files
- Mapping servers and database pools
- Deploying the configuration files out to the logical servers in the environment

To access the Define Environments page in My webMethods, select **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**.

The Define Environments page displays all environments that are defined and the configuration and deployment status for each environment. From the Define Environments page you can add new environments or edit, delete, copy, import, or export existing environments. You also can access the Database Pool Configuration page, which allows you to set up the database pools that can be used in any environment.



Environment configuration and deployment statuses are indicated by either a red circle (O) icon or a green ch eckmark (✓) icon. The following table describes these statuses in more detail.

Configuration or Deployment Status	Description
Configured	(O Indicates that the environment has not been configured or that the configuration is incomplete. ✓ Indicates that all of the logical servers are configured and the host server endpoints and database pools are mapped in the environment.
Ready to Deploy	(O Indicates that the environment configuration has not been validated. ✓ Indicates that the environment configuration has been successfully configured and validated and is ready for deployment.
Deployed	(O Indicates that the environment has not been deployed. ✓ Indicates that the environment has been successfully deployed.

The Define Environments Page Configuration Tabs

The My webMethods Define Environments page comprises seven configuration tabs. These seven configuration tabs are designed to step you through the process of creating, defining, and validating an environment configuration.

To access the configuration tabs, navigate to the Define Environments page. Then click an environment name in the list, or click the **Edit** icon () in the **Actions** column for the environment that you want to edit.

Note: The list of environments on the Define Environments page will be empty until you add an environment. Instructions for adding an environment are laid out in "Managing webMethods Optimize Environments" on page 31.

The seven tabs on the Edit Environment page are described in the following table.

Tab	Function	For usage instructions, see...
Design Servers	Allows you to add, edit, and delete logical servers in the environment.	"Adding Logical Servers to an Environment" on page 33
Configure Servers	Allows you to specify configuration settings for all logical servers in the environment.	"Configuring Logical Servers" on page 35
Define Hosts	Allows you to define hosts (or physical servers) in the environment.	"Defining Host Servers for an Environment" on page 91
Map Servers	Allows you to map each logical server in the environment to one or more physical servers.	"Mapping Logical Servers" on page 91
Map Endpoints	Allows you to map incoming and outgoing connections for each logical server.	"Mapping Endpoints" on page 92
Map DB Pools	Allows you to associate database (DB) pools to the logical servers within the environment.	"Mapping Database Pools" on page 94
Validate	Allows you to validate the logical server settings and connections in the environment.	"Validating an Environment Configuration" on page 95

Basic Steps for Deploying a Configuration

The following high-level steps describe how to create and roll out a configuration to an environment using the My webMethods Define Environments page.

To deploy a configuration

1. Install My webMethods and other webMethods products with theSoftware AG Installer.

   For installation instructions and database configuration details, see *Installing webMethods and Intelligent Business Operations Products*.

2. Start the Analytic Engine, the Web Service Data Collector, the Infrastructure Data Collector.

   See *Administering webMethods Optimize* for instructions.

3. Configure the database pools that you want to make available to the product components in the environment.

   For detailed instructions, see "Database Pool Configuration" on page 25.

4. Define and configure a webMethods environment.

   Use the first six configuration tabs on the Edit Environment page to define and configure a webMethods environment. Start with the **Design Servers** tab and continue configuring all but the final tab, until the servers and database pools are configured and mapped. For detailed instructions, see "Managing webMethods Optimize Environments" on page 31.

5. Use the **Validate** configuration tab on the Edit Environment page to validate the configuration.

6. Once a configuration is validated, you can deploy that configuration either to the environment or to a file by clicking the **Deploy** icon ().

   When you deploy a configuration to the environment for the first time, the logical servers automatically start using the configuration; however, if you edit and re-deploy a configuration, the logical servers in the environment must be restarted for the new configuration settings to take effect. For detailed instructions, see "Managing webMethods Optimize Environments" on page 31.

After an environment configuration has been deployed, you can modify and re-deploy at any time. Use My webMethods Define Environments page as described in "Managing webMethods Optimize Environments" on page 31 to edit and re-deploy.

Security

My webMethods supports HTTP over Secure Sockets Layer (SSL) for secure communications between the configuration agents and the configuration engine. For detailed instructions about setting up SSL with My webMethods BAM Configuration, see "Security" on page 101.

3 Database Pool Configuration

Overview

This chapter describes how to define and maintain the database pools that are shared by webMethods logical servers in one or more environments.

Configuring Database Pools

You configure the webMethods components in an environment to point to a database component by completing the following steps:

- Configure a JDBC connection pool for the database component.
- Associate the database component with the JDBC connection pool that you configured.

webMethods components also use JDBC connection pools to recycle database connections and reduce connection overhead.

A JDBC connection pool identifies a database and specifies pool parameters, such as minimum and maximum connections. After you define a connection pool, you link the appropriate function to that connection pool. For example, you would define a connection pool that identifies the database that contains the Process Audit database component, and then you would link the Process Audit function to that connection pool. Multiple functions can use the same connection pool. If you installed more than one database component in the same database, you could link both functions to the same connection pool.

Once the JDBC connection pools are configured, you can associate the appropriate functions or logical servers to connection pools from the **Map DB Pools** configuration tab in webMethods Central Configuration.

Note: You cannot complete the definition of an environment until you have defined database (DB) connection pools and mapped those DB pools to database activities.

Configuring a Connection Pool

To configure database pools

1. In My webMethods, click **Navigate > Applications > Administration > System-Wide > Environments > Database Pool Configuration**.
2. Click **Add Pool**.

   The Database Pool Configuration page displays fields for defining pool information, database connections, and pool settings.
3. In the **Pool Information** panel, complete the fields as described in the following table.

Field	Entry
Name	Enter a name, or alias, for the connection pool in the **Name** field. The alias can include only characters that are valid for a file name in your operating system. If you are defining the four database pools for Optimize, you might enter their aliases as `Analysis`, `ProcessTracker`, `ProcessAuditLog`, and `MywebMethodsServer`.
Description	Describe the connection pool. In this box you could describe the purpose of the database connection pools and the location of the database schema and the Optimize database activity or activities that the pool is connecting.

4. In the **Database Connection** panel, complete the fields as described in the following table.

Field	Entry
RDBMS	Choose the type of database.

Value	RDBMS
`Oracle`	Oracle
`DB2`	DB2 for Linux, UNIX, Windows
`SQL Server`	SQL Server

URL

Enter the URL to the database schema that contains one or more database component sets. Below are sample formats.

Oracle

Sample format:

```
jdbc:wm:oracle://host_or_IPaddress:port;
serviceName=database_name
```

SQL Server

Sample format:

```
jdbc:wm:sqlserver://host_or_IPaddress:port;
databaseName=database_name;SelectMethod=cursor
```

DB2 UDB

Field	Entry

	Sample format:
	`jdbc:wm:db2://server_name_or_IP_address:port;`
	`DatabaseName=database_name[;connection_option=value ...]`
	For DB2, you must add the following additional connection parameters:
	`AlternateId=schema;showSelectableTables=false`
	Here is the sample format with the two additional options:
	`jdbc:wm:db2://server-name-or-IPaddress:port;`
	`(DatabaseName=databasename|LocationName=location-name)`
	`[;AlternateID=schema;showSelectableTables=false]`
	`AlternateId` is the name of the default schema that is used to qualify unqualified database objects in dynamically prepared SQL statements. The `schema` parameter must be capitalized.
	If you are installing to a schema other than the default schema for the specified database user, you must also add this option as `[;connection_option=value ...]`:
	`InitializationString="SET CURRENT PATH=schema"`

Database User	Specify the database user who will communicate with the database.
Database Password	Set a password for the database user.

You can use the **Test** button to test the database connection once you've completed all of the fields in the **Database Connection** panel.

5. In the **Pool Settings** panel, inspect the default pool settings and make changes if needed, as described in the following table.

Field	Entry
Minimum Connections	Minimum number of connections that the connection pool must keep open at all times. If you use this pool alias for more than one function, each connection pool instance keeps the specified number of connections open. If your logging volume has sudden spikes, you can improve performance by making sure the connections needed to handle the increased volume open quickly. You can minimize connection startup time during spikes by setting this value higher, so that more connections remain open at all times.

Field	Entry
	The default value for minimum connections is 8.
Maximum Connections	Maximum number of connections that the connection pool can have open at one time. Calculate this value as part of the total possible number of connections that could be opened simultaneously by all functions and applications that write to the database. Make sure the total number does not exceed the database's connection limit. If one of the applications opens more connections than the database allows, the database will reject subsequent requests for connections from *any* application. The default value for maximum connections is 60.
Idle Connection Timeout	Period of time, in seconds, that the connection pool can keep an unused connection open. After the specified period of time, unused connections that are not needed to satisfy the **Minimum Connections** value are closed. If your logging volume has sudden spikes, you can improve performance by making sure the connections needed to handle the increased volume open quickly. You can minimize connection startup time during spikes by setting this value higher, so that more connections remain open at all times. The default value for the idle connection timeout is 20 seconds.
Ramp-up Delay	Period of time, in milliseconds, that the connection pool can wait between opening new connections. When the connection pool is started, the pool attempts to create the connections needed to achieve the specified minimum connections. For an environment in which several connection pools start up at the same time and share the same database, the database server can be overwhelmed with requests. Even though the database may have the capacity, the database listener can be overwhelmed by the sudden volume of requests. Use this parameter to force a connection pool to wait after a connection is created. The default value for the ramp-up delay is 0.
Connection Tries	Maximum number of times that the pool can attempt to open a connection. If a connection cannot be opened, the pool writes an exception to the log.

Field	Entry
	Both the minimum value and the default value for connection tries is `1`.
Retries Backoff	Period of time, in milliseconds, that the connection pool can keep trying after an unsuccessful attempt to open a connection. The default value is `0`. Set the value to a whole number greater than `1`. Note that during ramp-up, if a connection try is unsuccessful, the **Retries Backoff** value replaces the value set for **Ramp-up Delay**; the two values will not be aggregated.
Allow Statement Caching	This optional parameter specifies the maximum number of SQL statements to store in the cache for each connection. Enabling this option can improve performance because an application that executes the same statement repeatedly will not have to be recompiled for each execution. Instead, the application re-uses the statement from the cache. When the application returns the connection, the cache is purged. To enable statement caching, check the **Allow Statement Caching** check box and enter a whole number in the **Max per Connection** field.

6. Click **Save**.

   The connection pool appears in the **JDBC Pools** list on the Database Pools Configuration page.

7. Repeat the preceding sequence of steps for each new database connection pool you want to configure.

   To update or edit a connection pool, click the name of the pool in the **JDBC Pools** list or click the **Edit** icon () that appears beside the name of the pool.

   To associate logical servers to connection pools, use the **Map DB Pools** tab in webMethods Central Configuration. For instructions, see "Managing webMethods Optimize Environments" on page 31.

4 Managing webMethods Optimize Environments

Overview

This chapter describes how to use the My webMethods Define Environments page to configure and manage an Optimize environment. It also describes how to validate and deploy configuration settings to the logical servers that are required for using Optimize for Process and Optimize for Infrastructure environments. Also, it describes how to plan and implement Analytic Engine clustering.

The My webMethods Define Environments page enables you to configure, modify, and view the status of Optimize environments. If there are existing Optimize environments defined within your system, this page shows them and their status. Buttons to the right of each listed environment enable you to edit, copy, or deploy these existing environments and other controls enable you to create new environments.

Note that there are several modes for the My webMethods Define Environments page. The primary mode that is displayed when you first activate the page shows all currently configured environments for your system. If you click on an environment, the Edit Environment mode is activated, which displays the Environment Name and a Description in the Environment Information area, and details about the logical servers configured for that environment in the Environment Configuration area. The Environment Configuration pane contains a series of tabs that enable you to configure, validate, and deploy the logical servers that make up an Optimize configuration.

Creating a webMethods Optimize Environment

In order to configure Optimize for your specific needs and environment, you must appropriately configure the fields related to the following areas on the Define Environments page.

- Add appropriate logical servers to the environment
- Configure logical servers and necessary subcomponents of each logical server
- Map logical servers to a host server and to network endpoints

After you have created an environment and you have defined the configuration settings, you can deploy the configuration settings to the logical servers in the environment. You can update and re-deploy the configuration settings to the environment using the Define Environments page at any time.

Adding a New Environment

The following procedure outlines the high level steps required to create a new Optimize environment. Each step is amplified later in this chapter to provide more detailed information about the process represented by the steps.

Note that you can also import and export existing environments. Refer to "Exporting an Environment Configuration" on page 99 and "Importing an Environment Configuration" on page 99 for more information.

To add an environment

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses. If environments have not been defined, the **Environments** list is empty.

2. Click **Add Environment** to activate the Environment Information panel and begin defining a new environment.

3. On the **Environment Information** panel, type a name for the environment.

   You might want to choose a name that reflects the environment functionality, such as “development” or “production”, or choose a name that reflects the applicable process, such as “quote to collect”.

4. In the **Description** field, enter a description for the environment.

5. Click **Save**.

   The newly created environment appears in the **Environments** list.

6. To configure the environment, click the environment name or click the **Edit** icon ( ) in the **Actions** column for the environment name. The first step in configuring the environment is to add the logical servers, as described in the next section.

Adding Logical Servers to an Environment

The Edit Environment page enables you to add one or more logical servers to the environment. You can add a logical server to an environment using either of the following two methods:

- **Select from a list of known logical servers.**

  The list of known logical servers is populated from the definition templates that are installed with the configuration system. These definition files provide general information about each logical server as well as the requirements for other logical servers within the system. To deploy a configuration successfully, the environment must contain a complete set of related logical servers, for example:

  - One or more Analytic Engines
  - A JMS Server or server cluster, either webMethods Universal Messaging or Broker Server
  - Appropriate Data Collection Service

- My webMethods Server

- **Choose a template.**

  The Design Servers tab in the Environment Configuration area of Edit Environments page provides built-in templates to help you define Optimize environments. To access these templates, click the **Add From Template** button. These templates include webMethods Optimize for Infrastructure and webMethods Optimize for Process.

  Each template specifies common configurations and logical servers that are associated with the product. After you select a template, the configuration engine populates the environment with the logical servers that are defined in that template.

  The set of logical servers typically required for Optimize for Process is as follows:

  - Analytic Engine
  - webMethods Universal Messaging or Broker Server
  - Web Service Data Collector
  - My webMethods Server

  The set of logical servers typically required for Optimize for Infrastructure is as follows:

  - Analytic Engine
  - webMethods Universal Messaging or Broker Server
  - Infrastructure Data Collector
  - My webMethods Server

  Tip: Although you cannot edit the templates, you can change the name of any logical server by clicking the name of the server or by clicking the **Edit** icon () in the **Actions** column for the server.

To add a logical server

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the **Environments** list, click the **Edit** icon () in the row of the environment that you want to define or edit. You can also click the name of the environment in the **Environments** list.

3. On the Edit Environment page, click the **Design Servers** tab if that tab is not already active.

   The **Design Servers** tab displays a list of logical servers, the type of each logical server, and the name of the template used to create each server.

4. Do one or both of the following:

- Click **Add** to choose from a list of known logical servers. Select one or more logical servers from the list and click **OK**.
- Click **Add From Template** to choose a template.

Note: The **Add From Template** option is cumulative rather than additive. If you add a My webMethods Server and a Broker Server individually to an environment using the **Add** button and then select the Optimize for Infrastructure template using the **Add From Template** button, only an Infrastructure Data Collector and an Analytic Engine will be added to the environment. If you want to add other logical servers after selecting a template, select those additional servers using the **Add** button. The Optimize for Infrastructure template contains Logical Servers only for Analytic Engine, JMS Server, Infrastructure Data Collector, and My webMethods Server. The Optimize for Process template contains Logical Servers only for Analytic Engine, JMS Server, My webMethods Server, and Web Service Data Collector.

The next step in configuring an environment is to configure each logical server, as described in the next section.

Configuring Logical Servers

To configure logical servers, use the **Configure Servers** tab on the Define Environments page. The **Configure Servers** tab contains a tree that lists the common configurations and logical servers within the environment. The **Configure Servers** tab also displays the configuration requirements for each logical server.

Under each logical server is a list of required subcomponent configurations. The subcomponents vary according to the type of logical server. For example, to configure a Analytic Engine (one type of logical server), some of the subcomponent configurations required are as follows:

- JNDI configuration
- Database settings
- Journal logging

Click the name of a subcomponent configuration in the list to define that subcomponent's settings. An editing area is displayed beside the tree, allowing you to edit the configuration. If the **Use Default** check box is checked for a subcomponent, you must first clear the check box before you can edit the configuration.

After deployment, the configuration settings are stored in a configuration file that is cached locally on the logical server.

To configure a logical server

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the **Environments** list, click the **Edit** icon () in the row of the environment that you want to define or edit. You can also click the name of the environment in the **Environments** list.
3. On the Edit Environment page, click the **Configure Servers** tab.

   The **Configure Servers** tab displays a tree that lists the logical servers and logical server subcomponents within the environment. The tree also displays the configuration requirements for each logical server and the version of the configuration definition that was used when the logical server was created.
4. Click the name of a logical server configuration to edit the configuration.

   An editing area displays all configurable settings in the space beside the tree.
5. If necessary, edit or define the settings for the logical server subcomponent. The following table indicates where you can find more information about each subcomponent.

To define...	See...
Default settings	"Using Global Configuration Settings" on page 38
Analysis Engine settings	"Defining Analysis Engine Settings" on page 49
Adabas SOA Gateway Resource Module settings	"Defining Adabas SOA Gateway Resource Module Settings" on page 77
Broker Resource Module settings	"Defining Broker Resource Module Settings" on page 78
Analytic Engine Cluster configuration	"Defining Analytic Engine Cluster Settings" on page 46
Collector settings (for Infrastructure Data Collector)	"Defining Collector Settings" on page 79
Com-plete Resource Module settings	"Defining Com-plete Resource Module Settings" on page 81
Database settings	"Defining Database Settings" on page 50
Data maintenance settings	"Defining Data Maintenance Settings" on page 51

To define...	See...
E-mail settings	"Defining E-Mail Settings" on page 60
EntireX Resource Module settings	"Defining EntireX Resource Module Settings" on page 82
ETS Resource Module settings	"Defining ETS Resource Module Settings" on page 83
Infrastructure Data Collector settings	"Defining Logical Server Subcomponents for the Infrastructure Data Collector" on page 76
IS Resource Module settings	"Defining Integration Server Resource Module Settings" on page 85
JMSEventAction settings	"Defining JMSEventAction Settings" on page 54
JNDI configuration	"Defining JNDI Configuration Settings" on page 55
Journal logging	"Defining Journal-Logging Settings" on page 57
Monitor behavior settings	"Defining Monitor Behavior Settings" on page 62
Process Tracker settings	"Defining Process Tracker Settings" on page 63
SNMP alert settings	"Defining SNMP Alert Settings" on page 64
Station configuration	"Defining Station Configuration Settings" on page 65
Web service data collector settings	"Defining DataCollector Settings for the Web Service Data Collector" on page 70
Web service action settings	"Defining WSAction Settings" on page 66

6. Optionally, assign global environment settings for Analytic Engine cluster configuration, JNDI configuration, and journal logging, and lock configuration subcomponents by placing a check mark in the check box next to the subcomponent name, in the **Use Default** column. The check box in the **Use Default** column only appears next to those subcomponents that can

be shared across all logical servers. For more information about global configuration settings, see "Using Global Configuration Settings" on page 38 below.

7. Click **Save**, and **Finish** if needed, to complete the configuration tasks on the **Configure Servers** tab.
8. The next step in configuring the environment is to define one or more host (physical) servers to which the environment configuration is deployed. See "Defining Host Servers for an Environment" on page 91 for instructions.

Using Global Configuration Settings

Default settings are global or common settings that are configured at the environment level and can be shared between logical servers. Common default configuration settings include cache configurations, journal logging, and JNDI configuration.

When a configuration setting of this type is required by a logical server, a check box for default settings appears next to the subcomponent configuration in the logical server list. Check the **Use Default** check box to indicate that the logical server should use the common configuration settings for that subcomponent, or clear the **Use Default** check box to define a custom configuration.

Configuring Default Settings for Logical Servers

When you configure the Default setting for logical servers in the following categories, they are used by default for all applicable logical servers in your environment.

Defining JNDI Configuration Settings

You can use the JNDI Configuration setting on the Configure Servers tab to define the Uniform Resource Identifier (URI) that is required for JNDI/JMS between any applicable webMethods component and a JMS provider, and you can configure a secure connection (SSL) if desired. Also, if you are using a Universal Messaging cluster as your JMS Provider, you can point to the cluster appropriately.

If a single key and trust store are desired for the environment you can configure the JNDI Configuration under the Default Settings node in the configuration tree (on the Configure Servers tab). If you have individual certificates (Keystores) for each Optimize component you may override the default JNDI configuration by un-checking the **Use Default** check box for JNDI Configuration under the appropriate component in the configuration tree and updating the setting as appropriate.

To define default JNDI settings

1. Under the **Default Settings** node in the configuration tree (on the **Configure Servers** tab), click **JNDI Configuration**.
2. In the JNDI configuration area of the **JNDI Configuration for Environment Default Settings** area, the first two settings, **Broker Name** and **Naming Factory Type** are required for all systems, and the remaining settings are applicable based on your SSL needs:

Field	Description
Broker Name	If you choose "webMethods Universal Messaging" (the default) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select Broker as the JMS server in the **Naming Factory Type** field, this field should display the appropriate Broker name.
Naming Factory Type	Select the appropriate JMS server, either "Broker" or "Universal Messaging". If you choose "Universal Messaging", then **Broker Name** field should be empty. If you choose "Broker", then the **Broker Name** field must list the appropriate broker server name.
Enable SSL	Select this check box to enable a secure connection. The check box is unchecked by default and is not required.
Encryption	Select this check box to enable an encrypted connection. The check box is unchecked by default and is not required.
Key Store File	Type the absolute path to the private key file that is located on a server accessible by the component. This field is required only if you enable SSL.
Key Store Type	Click the drop down to select the Key Store Type (PKCS12 or JKS). This field is required only if you enable SSL.
Distinguished Name	Type the name of the appropriate certificate identifier. This field is required only if you enable SSL.
Trust Store File	Type the absolute path to the CA certificate file on a server location accessible by Analytic Engine. This field is required only if you enable SSL.
Trust Store Type	Click the drop down to select the Trust Store Type (PKCS12 or JKS). This field is required only if you enable SSL.

Field	Description
Key and Trust Store Password	Type the pass phrase to access the specified Key and Trust Stores. This field is required only if you enable SSL.
JMS Cluster URL	If your system uses a **Universal Messaging** cluster as the JMS provider, type the appropriate URL to identify the cluster. Note that, if provided, this setting overrides the JMS Provider URL generated from the JMS Endpoint and the information in the **Broker Name** field. This field should be coordinated with the value specified in the **Naming Factory Type** field. The format for this URL is <protocol>:// <host>:port, <protocol>://<host>:port, etc. Valid protocols are nsp, nsps, nhp, and nhps. Also, note that you must configure your **Universal Messaging** cluster in an appropriate manner for your needs and system configuration. Refer to the *webMethods Universal Messaging Clustering Guide* for more information about configuring and managing **Universal Messaging** clusters.

3. Click **Save** to save changes you have made in the JNDI configuration area, or click **Reset** to discard any changes.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

4. Click **Finish** to close the **Configure Servers** tab and return to the Define Environments page.

Defining Journal-Logging Settings

You can define the common or shared journal-logging targets that you want to make available to Optimize (or its logical servers) in the environment. Each journal-logging target represents a file that can receive and store logging data of a specified type from webMethods components in the environment.

The Configure Servers tab on the Define Environment page enables you to configure default targets for the environment. These targets must be defined before the webMethods components in the environment can start logging.

When you configure journal-logging settings for a component, you set the logging level, or threshold, and assign targets to journal loggers. Each logger receives log events from application source code and forwards those log events to all targets that are assigned to the logger as well as to all targets that are assigned to its ancestor loggers. The logger determines whether to forward each event by comparing the event's level to the logger's threshold.

To define journal-logging settings

1. Under the **Default Settings** node in the configuration tree (on the **Configure Servers** tab), click **Journal Logging**.
2. In the **Journal Loggers for Environment Default Setting** panel, define the threshold level for the root journal logger in the journal-logging configuration area.

   Threshold levels indicate the level of information that you want to log to a specified target. If a journal logger is not assigned a threshold level, that logger inherits the assigned level from the root logger. The seven predefined threshold levels are listed below in order of *decreasing* importance:

   - Fatal
   - Error
   - Warn
   - Info
   - Debug
   - Trace
   - None, or off

   With the exception of None (off), you can assign one of these predefined threshold levels for each target. The journal logger logs messages that are of the specified threshold level and also logs messages that exceed the specified threshold level. For example, if you specify the threshold level as Warn, the logger will log Warn, Error, and Fatal messages.

   Note: The available predefined threshold levels are determined at the product level.

3. If necessary, edit any target in the **Target Name** list by clicking the name of the target.

   In the **Add/Edit Target** area, you can change the name, description, and threshold level for that target. (For more information about threshold levels, see the preceding step.)

4. If you edit a target that is written to a file, you also can specify various rollover details related to the log file. Do one of the following:

   - To change any of these rollover details, review the following several steps.
   - To leave all of the rollover details as is, proceed to step 10.

5. In the **File** field (in the **Add/Edit Target** area on the **Configure Servers** tab), enter the name of the file to which you want the logging data to be written.
6. Use the **Rollover file by** field to indicate how often you want to close the log file and open a new log file. Choose a rollover method from the following three options:

   - **Time Period only**

     Note: All times are calculated according to the logical server's local time zone.

- **File Size only**

  If you choose **File Size only**, complete the **Max File Size** field by entering the size (in megabytes) at which you want the log file to be rolled over. The file will always be rolled over at midnight in the logical server's local time zone, but if the file reaches the specified maximum file size before midnight, the file will be rolled over at that point *as well*.

- **Size or Period**

  If you choose **Size or Period**, complete both the **Period** field (see the next step for more information) and the **Max File Size** field (see the previous bullet for more information).

7. If you specify log file rollover by time period, you must also complete the **Period** field.

   When you specify log file rollover by time period, each new log file is saved and named with the current date and time using ISO 8601 format (*yyyy-mm-ww-dd-hh:mm:ss,sss*). The option specified in the **Period** field determines which date/time characters are appended to each log file name (see the following table for more information). All times are calculated according to the logical server's local time zone.

   If more than one log file is rolled over in a 24-hour period, the file names also will include a sequential marker, beginning with the characters `-1`, after the date/time characters. So, for example, the first log file rolled over in a 24-hour period would simply be named with the current date and time, using the format *yyyy-mm-ww-dd-hh:mm:ss,sss* . If a second log file is rolled over within that same 24-hour period, the characters `-1` would be added to the second file name after the date/time characters. If a third log file is rolled over within that same 24-hour period, the characters `-2` would be added at the end of the third file name, and so on.

   Select from the following options for the **Period** field:

Field	Description	Date/Time Characters Appended to Log File Name
Midnight	Creates a new log file at midnight. Entries are added to that log file up until midnight of the next day.	*yyyy-mm-dd*
Minute	Creates a new log file every minute.	*yyyy-mm-dd-hh:mm*
Hourly	Creates a new log file every hour.	*yyyy-mm-dd-hh* , where *hh* uses a 24-hour clock
Half Daily	Creates a new log file at noon and then again at midnight.	*yyyy-mm-dd* -{AM|PM}

Field	Description	Date/Time Characters Appended to Log File Name
Weekly	Creates a new log file each Sunday at midnight.	*yyyy-ww*
Monthly	Creates a new log file on the last day of each month at midnight.	*yyyy-mm*

8. In the **Max # Log Files** field, specify the maximum number of log files that you want to be retained.
9. In the **Rollover Destination** field, specify the directory where rolled over log files should be stored.

   If you choose **Directory** as your rollover destination, complete the additional fields as described in the following table.

Field	Description
Log Directory	Enter the name of the directory where the rollover log files will be saved.
Host Name	By default, this field displays the name of the host for the current target. Change the host name if needed.
Date Format	This field shows the date format that will be used for rolled over log files. No editing is needed.
Time Format	This field shows the time format that will be used for rolled over log files. No editing is needed.

10. When you are satisfied with the configuration of each target, click **Save** in the **Add/Edit Target** area to save the configurations, or click **Cancel** to discard any changes you have made.

After the targets are defined and deployed using the My webMethods Define Environments page, journal logging can be further configured in the webMethods components that use journal logging.

Defining the Terracotta Server Array Setting

The Terracotta Server Array must be configured for systems that use Analytic Engine clustering.When configuring this setting, it is important to understand all of the options and considerations related to Analytic Engine clustering. Refer to the following section for more information about Analytic Engine clustering and Terracotta Server Array configuration.

Analytic Engine Clustering

Analytic Engine clustering distributes the Optimize information processing load across multiple Analytic Engines, either to facilitate system high availability or to maximize Analytic Engine data throughput. Analytic Engine clustering requires a Terracotta Server Array to coordinate system throughput. A Terracotta Server Array (TSA) is a combined hardware software solution to managing data processing that facilitates scaling and clustering. It uses in memory storage and processing rather than a conventional database. Terracotta based clustering is appropriate for mission-critical implementations and for some systems that experience high levels of data throughput. Depending on your system availability needs and resources, you can configure either a single node TSA or a distributed TSA.

This section describes important considerations and potential costs and benefits of implementing Analytic Engine clustering. Also, it explains how to implement clustering. Planning, implementing, and configuring a TSA involves many variables and considerations that are beyond the scope of this document. For information about planning, configuring, and running a Terracotta Server Array, refer to the Terracotta web site and product documentation. For information about installing Terracotta in a webMethods environment, refer to *Installing webMethods and Intelligent Business Operations Products*.

Optimize users should understand that clustering is not appropriate for every system configuration, and they should weigh the requirements and effort of implementation against potential benefits. In general, a non-clustered system is appropriate if you have adequate hardware resources to manage your data load with a single Analytic Engine. Also, a non-clustered configuration is appropriate if hardware is limited and your system is not mission critical, and you are not concerned with system downtime in the event of a hardware/software failure.

Also, while clustering can increase system data throughput, it may or may not be an effective means of enhancing system performance, depending on your specific configuration and data volume. In fact, configuring two Analytic Engine nodes typically causes a reduction in data throughput on individual machines due to the TSA overhead involved in managing data across nodes, though this throughput penalty should disappear as you add more Analytic Engine nodes.

Clustering is an appropriate option for Optimize users who require system high availability and want to minimize the risk of a single Analytic Engine-related failure point. In a clustered system, data is distributed across all nodes and redundancy is enforced so that failure of a single node will not result in system down time. For most system configurations, a two node Analytic Engine cluster provides a reasonable balance of high availability and system throughput. Note that with a TSA based clustering configuration, you can add Analytic Engine nodes at any time without any system re-configuration.

Also, note that while a clustered system configured with an appropriate TSA and multiple Analytic Engines nodes can provide a significant degree of system high availability, you must plan for and implement database and Broker high availability

separately, if total system high availability is required for your system configuration. In addition, to meet typical high availability requirements, you must configure a distributed TSA as opposed to a single node implementation, which requires appropriate hardware and a fairly complex system configuration.

From a performance perspective, clustering is most effective in maximizing data throughput for situations that involve a loading characteristics with many KPI instances and a small number of events per minutes occurring for each KPI instance. The quantity of rules defined against the KPI instances also affects performance. If you have multiple rules defined against each KPI instance, increasing the number of Analytic Engine nodes will generally maximize data throughput.

Analytic Engine clustering is available for all Optimize environments that use a TSA. To implement clustering, you must set up the appropriate number of machines running Analytic Engines and configure the environment as described in "Defining Analytic Engine Cluster Settings" on page 46. The system will automatically configure the `sag.opt.clusterable.caches.xml` file located on each computer in the cluster, based on the number of machines/Analytic Engines available.

Implementing Analytic Engine Clustering

The following procedure describes the high level steps required to configure Optimize for Analytic Engine clustering with links to the appropriate sections of this chapter containing more specific information. Before completing these steps, you should understand your goals for clustering and make certain that you have the appropriate hardware available and that it is configured appropriately. Also, you must have a Terracotta Server Array set up and configured. Refer to the Terracotta documentation and web site for information about setting up and configuring a TSA.

When you first implement clustering, or if you change from a clustered to a non-clustered system, you must configure your Analytic Engines and deploy your configuration as described in the following procedure. Note that you can add or remove Analytic Engines in an existing cluster, without redeploying your environment.

Note: Each Analytic Engine in a cluster must have a unique identity; that is to say, each Analytic Engine must have its own logical server. When you create a logical server, the system creates a unique ID and that ID is pushed to the Analytic Engine when the environment is deployed. If you configure multiple Analytic Engines that share a single ID, they will not function in a clustered environment.

To implement an Analytic Engine cluster:

1. On the Central Configuration Design Servers tab, add the appropriate number of Analytic Engine logical servers for your system. See "Adding Logical Servers to an Environment" on page 33 for more information.
2. Review Analytic Engine sub-component settings for each logical server.

   Ensure that all (global configuration) settings across the Analytic Engine logical servers are the same. For instance, if you decide to change the "Data Maintenance

Settings", you must make the identical change for each Analytic Engine logical server in the cluster.

Also, ensure that the TSA URL is populated in either the default TSA configuration or each Analytic Engine logical server TSA configuration. See "Defining Analytic Engine Cluster Settings" on page 46 for more information.

3. Define the appropriate hardware hosts for your Analytic Engine cluster as appropriate on the Define Hosts tab. See "Defining Host Servers for an Environment" on page 91 for more information.
4. Map your Analytic Engine logical servers to the desired host server machines as appropriate on the Map Servers tab. See "Mapping Logical Servers" on page 91 for more information.
5. Map endpoints as appropriate for your system configuration. See "Mapping Endpoints" on page 92 for more information. When mapping endpoints for a cluster you should consider the following:
   - When deploying more than one Analytic Engine to the same host, ensure that each Analytic Engine uses unique ports for the Configuration Agent and WS Registry under the "INCOMING CONNECTIONS".
   - When deploying multiple Analytic Engines to unique hosts, each Analytic Engine can use the same port numbers.
   - The JMS provider must be the same across all Analytic Engines participating in the cluster.
6. Map the database pools as appropriate for your cluster. See "Mapping Database Pools" on page 94 for more information.
7. Validate your configuration as described in "Validating an Environment Configuration" on page 95.
8. Deploy your configuration to the host systems as described in "Deploying an Environment" on page 95.

Defining Analytic Engine Cluster Settings

There are two types of Analytic Engine cluster configurations used for Optimize systems, depending on the number of available Analytic Engines and whether or not you have configured a Terracotta Server Array. Note that if you want to use Analytic Engine clustering, you must have a TSA:

- Non-clustered - A system configuration in which there is no TSA and a single local Analytic Engine for all data processing.
- Clustered - Used when there is a TSA and one or more Analytic Engines in a system configuration. All data is all maintained on the TSA and handed to the Analytic Engine nodes when those nodes are working on the data.

After your machines are configured appropriately, you must update the environment configuration as described in the following procedure.

To configure the Terracotta Server Array setting on the My webMethods Edit Environment page

1. Under the **Default Settings** node in the configuration tree (on the **Configure Servers** tab), click **Terracotta Server Array Configuration**.
2. On the **Terracotta Server Array for Environment Default Settings** panel, type the URL for the TSA in the **Terracotta Server Array URL** field.

   Typically, the format for a single node TSA URL is as follows: <host>:<port>

   For a multiple node TSA, the URL would use the following format: <server_host1>:<port>,<server_host2>:<port>
3. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without clicking **Save**, any changes made to the settings will be lost.
4. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
5. Click **Finish** to close this page and return to the Define Environments page.

If you receive an error indicating that one or more machines cannot join the cluster, you may need to change the cluster port to a different setting.

Terracotta Configuration Guidelines for Analytic Engine Clustering

Software AG has developed some configuration guidelines and suggestions to maximize Optimize system performance when using a Terracotta Server Array. Some of these configuration changes are requirements for Analytic Engine clustering, and some are additions and modifications to a standard Terracotta configuration that testing has shown will enhance system performance in typical clustered environments. This section describes these guidelines and the benefits and trade-offs involved in various Terracotta configurations. This section does not describe the specific procedures required for setting up a TSA in a particular environment as those procedures are beyond the scope of this document. For information on setting up a TSA, refer to the Terracotta web site and associated documentation.

Note that if you are setting up a clustered system with various components running under different operating systems, you must be aware of and plan for some complications in regards to Terracotta and timestamps. For instance, you can configure your TSA on hardware running under Linux and have your Analytic Engines on hardware running Microsoft Windows. In such cases, problems can arise due to the fact that the different operating systems use different timestamp protocols, and because Terracotta uses these timestamps to keep track of data. If you are configuring a system that uses different operating systems you must be aware of these issues and implement the appropriate strategies to deal with them. Refer to the Terracotta discussion boards for information about identifying and dealing with these issues.

There are four particular areas in which users should consider customizing their Terracotta installation in an Analytic Engine clustered environment, based on system configuration considerations and desired performance characteristics.

- Persistence Setting
- JVM arguments in start up script
- Optional ulimit settings to resolve log errors
- Mandatory property updates

Adjusting Java Virtual Machine Arguments

There are several JVM arguments that may be required in the Terracotta startup script in order for the TSA to function optimally with an Analytic Engine cluster. These include settings for security and the appropriate path to the TSA install bin folder. These changes are implemented in the start-tc-server.bat file (Windows) or start-tc-server.sh file (Linux).

Note that the instructions in this section assume that you have installed and configured a Terracotta Server Array as appropriate for your system needs in accordance with the Terracotta instructions.

Before making these changes, ensure that the path to the tc.server.sh/tc.server.bat file is correct for your system. This path varies according to each system installation and is specified by the <log> tag in the tc-config.xml file.

- The following Java command (`-Xmx4g -Xms4g`) changes the default available RAM from 1G to 4G. You can set it to any value that is appropriate for your system, but testing has shown that 4G provides optimal performance for most typical clustered systems.
- The JVM `-XX:+UseParallelGC` and `-XX:+UseParallelOldGC` arguments will enhance performance for most systems. You should consider adding one, but not both, of these arguments if you are concerned about performance.
- The `XX:+HeapDumpOnOutOfMemoryError` command is a recommended addition and makes it easier for users to debug problems if the TSA malfunctions for some reason.

Configure the Persistence Mode Setting

The persistence mode setting determines how your TSA manages data. For systems that use a distributed TSA configured for high availability, persistent swap is usually the appropriate choice. If you have configured a single-node TSA and are not concerned about high availability, you should configure the persistence mode to temporary-swap-only, as shown in the following example.

```
<persistence>
<mode>temporary-swap-only</mode>
   <offheap>
    <enabled>false</enabled>
    <maxDataSize>40g</maxDataSize>
   </offheap>
<persistence>
```

Mandatory Property Updates

The following property must be added to the Terracotta tc-config.xml file in order for a Analytic Engine clustering to work correctly with a TSA.

```
<maxDataSize>40g</maxDataSize>
```

Changing the Terracotta ulimit Settings

If your TSA operates in a Linux environment and you encounter errors in the Terracotta server log (whose location is specified in your tc-config.xml file), you may need to adjust your ulimit setting. This setting defines the maximum number of open files. If you encounter file errors, try increasing the number of open files to 10240.

Defining Logical Server Subcomponents for the Analytic Engine

This section contains description of the procedures for defining logical server subcomponents for the Analytic Engine. For the Analytic Engine, there are thirteen subcomponents for which you must review or configure settings:

- Analysis Engine settings
- Database settings
- Data Maintenance settings
- Event Publication settings
- JMSEventAction settings
- JNDI Configuration settings
- Journal Logging settings
- Email settings
- Monitor Behavior settings
- Process Tracker settings
- SNMPAlert settings
- Station Configuration settings
- WSAction settings

Defining Analysis Engine Settings

You can adjust the tolerance and threshold values for the Analytic Engine to increase or reduce the amount of activity from diagnosis warnings.

When a tolerance value is met, the Analytic Engine creates a diagnosis of above normal or below normal. That diagnosis appears in My webMethods on the Analytics Overview page and other pages throughout the system, and a notification is issued if a rule has been created for that condition.

To define Analysis Engine settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment for which you want to define Analysis Engine settings.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Analysis Engine Settings**.
5. In the **Analysis Engine Settings for Analytic Engine** area, define the Analysis Engine settings that you want the logical server to use:

Field	Description
Tolerance	The number of standard deviations to allow before creating a diagnosis. The default value is 1.0.
Trending Threshold	The number of consecutive trends that can occur before a diagnosis is made. The default value is 3.
Trending Tolerance	A number between 0 and 1 that indicates a percentage. Optimize will only trigger trending if the difference between readings is greater than the percentage indicated in this field.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.
7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining Database Settings

You can define database settings to enable and disable the automatic execution of data definition language (DDL) statements on the database. Automatic execution of DDL statements is enabled by default.

When the automatic execution of DDL statements is disabled, the Analytic Engine runs in Static DB Schema mode. This means that you cannot create, edit or delete dimensions, event maps, KPI hierarchies or KPI definitions and all respective buttons in the My webMethods user interface will be dimmed.

Disabling the automatic DDL execution also affects the deployment process for Optimize assets. For more information about deploying Optimize deployment sets when Analytic Engine runs in Static DB Schema mode, see the PDF publication *webMethods Deployer User's Guide*.

To define database settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Database Settings**.
5. In the **Database Settings for Analytic Engine** area, click **Disable DDL Statements** to disable the automatic execution of DDL statements.
6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   Note: If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.
7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining Data Maintenance Settings

You can define data maintenance settings to control the number of days to retain business event data, the number of days to retain aggregated business event data, and the frequency with which Optimize recalculates the values in the `OPERATION_PARAMETER` table.

To define data maintenance settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment for which you want to define data maintenance settings.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Data Maintenance Settings**.
5. In the **Data Maintenance Settings for Analytic Engine** area, define the data maintenance settings as follows:

Field	Description
Business Days To Retain	Number of days to retain business event data in the Analysis database component. The default value is 30 days. Business data includes metrics about processes (such as cycle time, instance count, or error count) as well as data captured from within processes (such as order revenue, items ordered, or line count). After the specified number of days have passed, the data is eligible to be purged by the purge procedures.
Data Maintenance Interval	Number of hours before the Analytic Engine updates the `Operation_parameter` table with dynamic table names. The default value is 4 hours.
Aggregated Business Days To Retain	Number of days to retain aggregated business event data in the Analysis database component. The default value is 365 days. Aggregated business data represents a consolidation of business event data that is used to improve the performance of the KPI Summary and KPI Instance Detail graphs. Aggregated business data takes up much less space and therefore can be kept for a longer period of time without consuming excessive disk space or affecting system performance.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

8. Click **Finish** to close this page and return to the Define Environments page.

Defining Event Publication Settings

Event publication settings enable you to configure Optimize to publish events for consumption by other Software AG applications. You can choose to publish events for three Optimize related areas: KPI readings, KPI statistics, and processes tracking. Note

that if you make changes to event publication settings, deployment is automatic; you do not need to restart Analytic Engine for the changes to be implemented.

Note: If your Optimize system operates in a EDA enabled environment using NERV, then you must configure the NERV JMS Provider setting in the profile for the Platform Manager (SPM) that is running on the same machine as your Analytic Engine. By default, the NERV JMS Provider is defined as `nsp://<host_name> :9000`. For specific instructions, refer to the instructions in the "Modifying Transport Layer Configuration" section of the *Implementing Event Driven Architecture with webMethods Products* guide. For more information about configuring Optimize for subscription and publication of events in an EDA environment, refer to "Configuring EDA/NERV Settings for Optimize" in the "Configuring Optimize" chapter of *Administering webMethods Optimize*.

To define event publication settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment for which you want to define data maintenance settings.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Event Publication Settings**.
5. In the **Event Publication Settings for Analytic Engine** area, click the appropriate check box to publish the desired events as follows:

Field	Description
Publish Events for KPI Readings	Publishes all events related to Optimize KPI readings.
Publish Events for KPI Statistics	Publishes all events related to Optimize KPI statistics.
Publish Events for Process Tracking	Publishes all events related to Optimize process tracking

6. If applicable, use the **KPI Filter Options Include** and **Exclude** radio buttons and the **Filtered KPI Names** box to include or exclude specific KPI events for publication. Type the desired KPI names in the **Filtered KPI Names** box and then select the appropriate radio button depending on whether you want to include or exclude the specified KPIs.
7. If applicable, use the **Process Filter Options Include** and **Exclude** radio buttons and the **Filtered Process Names** box to include or exclude specific process events for publication. Type the desired process names in the **Filtered Process Names** box and then select the appropriate radio button depending on whether you want to include or exclude the specified processes.

Note: If the **Filtered KPI Names** or **Filtered Process Names** box is empty and you select the **Exclude** radio button, then events for all KPIs or processes will be published. If you select the **Include** radio button with the **Filtered KPI Names** or **Filtered Process Names** box empty, then no events for any KPIs or processes will be published.

8. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

Defining JMSEventAction Settings

You can configure JMSEventActions to facilitate Optimize-based temporal analysis on rule violations. In essence, this functionality enables you to monitor rule violations over time so that you can perform mores sophisticated analysis of events and processes than standard Optimize rule-related functionality supports. Using this functionality, you can send JMS messages to a Broker (local or non-local) in response to rule violations. These JMS messages can be monitored by a KPI that monitors rule violations over a specified time frame and sends the appropriate notifications regarding the rule violations to a designated user.

To configure a JMSEventAction, you must edit the provided XML snippet with the appropriate parameters.

To configure JMSEventActions

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **JMSEventAction Settings**.

   A new area appears beside the configuration tree, labeled **JMSEventAction Settings for Analytic Engine**.
5. Identify the JMSEventAction to be published, along with the attributes that the event will contain, by editing the XML snippet in the text editor in the settings area. As installed, the XML is configured as a comment (<!-- xml -->). Remove the comment formatting and modify the XML according to the instructions in the XML, commenting out the instructions and notes.

   Optimize displays the JMSEventAction you identified in place of *Test Action* on the Add/Edit/Copy Rule page and the Rule Details page.
6. To identify more than one JMSEventAction, copy and edit the XML shown in the example box above (step 5) for each action.

7. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.
8. Apply the changes by restarting the Analytic Engine.

   For more information about restarting the Analytic Engine, see *Administering webMethods Optimize*.

Defining JNDI Configuration Settings

By default an Analytic Engine logical server uses the JNDI configuration settings defined for the Default Settings configuration on the Configure Servers tab. If you want to use custom settings for an Analytic Engine, un-check the **Use Default** check box and the complete JNDI Configuration settings will be displayed in an editable form.

If a single key and trust store are desired for the environment you can configure the JNDI Configuration under the Default Settings node in the configuration tree (on the Configure Servers tab). If you have individual certificates (Keystores) for each Optimize component you may override the default JNDI configuration by un-checking the **Use Default** check box for JNDI Configuration under the appropriate component in the configuration tree and updating the setting as appropriate.

To define JNDI Configuration settings for an Analytic Engine

1. Under the **Default Settings** node in the configuration tree (on the **Configure Servers** tab), click **JNDI Configuration**.
2. Un-check the **Use Default** check box to display the JNDI Configuration settings in an editable format.
3. In the JNDI configuration area under the **JNDI Configuration for Analytic Engine** heading, the first two settings, **Broker Name** and **Naming Factory Type** are required for all systems, and the remaining settings are applicable based on your SSL needs:

Field	Description
Broker Name	If you choose "webMethods Universal Messaging" (the default) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select Broker as the JMS server in the **Naming Factory Type** field, this field should display the appropriate Broker name.
Naming Factory Type	Select the appropriate JMS server, either "Broker" or "Universal Messaging". If you choose "Universal Messaging", then **Broker Name** field should be empty. If you choose "Broker", then the **Broker Name** field must list the appropriate broker server name.

Field	Description
Enable SSL	Select this check box to enable a secure connection. The check box is unchecked by default and is not required.
Encryption	Select this check box to enable an encrypted connection. The check box is unchecked by default and is not required.
Key Store File	Type the absolute path to the private key file that is located on a server accessible by the component. This field is required only if you enable SSL.
Key Store Type	Click the drop down to select the Key Store Type (PKCS12 or JKS). This field is required only if you enable SSL.
Distinguished Name	Type the name of the appropriate certificate identifier. This field is required only if you enable SSL.
Trust Store File	Type the absolute path to the CA certificate file on a server location accessible by Analytic Engine. This field is required only if you enable SSL.
Trust Store Type	Click the drop down to select the Trust Store Type (PKCS12 or JKS). This field is required only if you enable SSL.
Key and Trust Store Password	Type the pass phrase to access the specified Key and Trust Stores. This field is required only if you enable SSL.
JMS Cluster URL	If your system uses a **Universal Messaging** cluster as the JMS provider, type the appropriate URL to identify the cluster. Note that, if provided, this setting overrides the JMS Provider URL generated from the JMS Endpoint and the information in the **Broker Name** field. This field should be coordinated with the value specified in the **Naming Factory Type** field. The format for this URL is <protocol>://<host>:port, <protocol>://<host>:port, etc. Valid protocols are nsp, nsps, nhp, and nhps. Also, note that you must configure your **Universal Messaging** cluster in an appropriate manner for your needs

Field	Description
	and system configuration. Refer to the *webMethods Universal Messaging Clustering Guide* for more information about configuring and managing **Universal Messaging** clusters.

4. Click **Save** to save changes you have made in the JNDI configuration area, or click **Reset** to discard any changes.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

5. Click **Finish** to close the **Configure Servers** tab and return to the Define Environments page.

Defining Journal-Logging Settings

You can define the common or shared journal-logging targets that you want to make available to Optimize (or its logical servers) in the environment. Each journal-logging target represents a file that can receive and store logging data of a specified type from webMethods components in the environment. By default an Analytic Engine logical server uses the Journal Logging settings defined for the Default Settings configuration on the Configure Servers tab. If you want to use custom settings for an Analytic Engine, un-check the **Use Default** check box and the complete Journal Logging settings will be displayed in an editable form.

When you configure journal-logging settings for a component, you set the logging level, or threshold, and assign targets to journal loggers. Each logger receives log events from application source code and forwards those log events to all targets that are assigned to the logger as well as to all targets that are assigned to its ancestor loggers. The logger determines whether to forward each event by comparing the event's level to the logger's threshold.

To define journal-logging settings

1. Under the **Analytic Engine** node in the configuration tree (on the **Configure Servers** tab), click **Journal Logging**.
2. Un-check the **Use Default** check box to display the Journal Logging settings in an editable format.
3. In the **Journal Loggers for Environment Default Setting** panel, define the threshold level for the root journal logger in the journal-logging configuration area.

   Threshold levels indicate the level of information that you want to log to a specified target. If a journal logger is not assigned a threshold level, that logger inherits the assigned level from the root logger. The seven predefined threshold levels are listed below in order of *decreasing* importance:

   - Fatal
   - Error

- Warn
- Info
- Debug
- Trace
- None, or off

With the exception of None (off), you can assign one of these predefined threshold levels for each target. The journal logger logs messages that are of the specified threshold level and also logs messages that exceed the specified threshold level. For example, if you specify the threshold level as Warn, the logger will log Warn, Error, and Fatal messages.

Note: The available predefined threshold levels are determined at the product level.

4. If necessary, edit any target in the **Target Name** list by clicking the name of the target.

   In the **Add/Edit Target** area, you can change the name, description, and threshold level for that target. (For more information about threshold levels, see the preceding step.)

5. If you edit a target that is written to a file, you also can specify various rollover details related to the log file. Do one of the following:
   - To change any of these rollover details, review the following several steps.
   - To leave all of the rollover details as is, proceed to step 10.

6. In the **File** field (in the **Add/Edit Target** area on the **Configure Servers** tab), enter the name of the file to which you want the logging data to be written.

7. Use the **Rollover file by** field to indicate how often you want to close the log file and open a new log file. Choose a rollover method from the following three options:
   - **Time Period only**

     Note: All times are calculated according to the logical server's local time zone.

   - **File Size only**

     If you choose **File Size only**, complete the **Max File Size** field by entering the size (in megabytes) at which you want the log file to be rolled over. The file will always be rolled over at midnight in the logical server's local time zone, but if the file reaches the specified maximum file size before midnight, the file will be rolled over at that point *as well*.

   - **Size or Period**

     If you choose **Size or Period**, complete both the **Period** field (see the next step for more information) and the **Max File Size** field (see the previous bullet for more information).

8. If you specify log file rollover by time period, you must also complete the **Period** field.

When you specify log file rollover by time period, each new log file is saved and named with the current date and time using ISO 8601 format (*yyyy-mm-ww-dd-hh:mm:ss,sss*). The option specified in the **Period** field determines which date/time characters are appended to each log file name (see the following table for more information). All times are calculated according to the logical server's local time zone.

If more than one log file is rolled over in a 24-hour period, the file names also will include a sequential marker, beginning with the characters `-1`, after the date/time characters. So, for example, the first log file rolled over in a 24-hour period would simply be named with the current date and time, using the format *yyyy-mm-ww-dd-hh:mm:ss,sss* . If a second log file is rolled over within that same 24-hour period, the characters `-1` would be added to the second file name after the date/time characters. If a third log file is rolled over within that same 24-hour period, the characters `-2` would be added at the end of the third file name, and so on.

Select from the following options for the **Period** field:

Field	Description	Date/Time Characters Appended to Log File Name
Midnight	Creates a new log file at midnight. Entries are added to that log file up until midnight of the next day.	*yyyy-mm-dd*
Minute	Creates a new log file every minute.	*yyyy-mm-dd-hh:mm*
Hourly	Creates a new log file every hour.	*yyyy-mm-dd-hh* , where *hh* uses a 24-hour clock
Half Daily	Creates a new log file at noon and then again at midnight.	*yyyy-mm-dd* -{AM|PM}
Weekly	Creates a new log file each Sunday at midnight.	*yyyy-ww*
Monthly	Creates a new log file on the last day of each month at midnight.	*yyyy-mm*

9. In the **Max # Log Files** field, specify the maximum number of log files that you want to be retained.
10. In the **Rollover Destination** field, specify the directory where rolled over log files should be stored.

    If you choose **Directory** as your rollover destination, complete the additional fields as described in the following table.

Field	Description
Log Directory	Enter the name of the directory where the rollover log files will be saved.
Host Name	By default, this field displays the name of the host for the current target. Change the host name if needed.
Date Format	This field shows the date format that will be used for rolled over log files. No editing is needed.
Time Format	This field shows the time format that will be used for rolled over log files. No editing is needed.

11. When you are satisfied with the configuration of each target, click **Save** in the **Add/Edit Target** area to save the configurations, or click **Cancel** to discard any changes you have made.

After the targets are defined and deployed using the My webMethods Define Environments page, journal logging can be further configured in the webMethods components that use journal logging.

Defining E-Mail Settings

You can configure the Analytic Engine to send an e-mail when a rule is violated that contains an alert to a specified recipient. E-mail alerts are formatted according to alert templates. When a certain rule is violated, you can specify that the e-mail alert that is sent is formatted according to the specified alert template. You can specify multiple template property tags to create multiple rule and alert template-file combinations.

When an e-mail alert is formatted, the Analytic Engine attempts to retrieve the template from the location specified in the **Mail Settings** panel, according to the following two options:

- If no template is specified or if the template cannot be found, the e-mail alert is formatted with the default formatting. The default formatting cannot be changed. For more information about creating custom alert templates, see the PDF publication *Administering webMethods Optimize*.
- If a template is specified, the e-mail alert is formatted using that specified template.

To define e-mail settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.

4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Mail Settings**.

   A new area appears beside the configuration tree, labeled **Mail Settings for Analytic Engine**.

5. Configure the settings in the mail settings area:

Field	Description
Mail Server	Name of the host system on which the SMTP server resides that sends e-mail containing alerts. For example, this server might be `smtp.client.com`.
Authentication Required	Click this check box if the server requires user authentication.
Server User	User ID that is required if the server uses authentication.
Server Password	Password that is required if the server requires authentication.
Sender Domain	E-mail domain for the e-mail address specified in the **Default Sender** field (such as `webmethods.com`).
Default Sender	Sender to specify in the **From** field of the alert e-mails. Make sure the e-mail address provided in this field is within the e-mail domain specified in the **Sender Domain** field.
Admin Address	E-mail address of the recipient that you want to receive copies of e-mail alerts. The e-mail address in this field is also used as the default address for sending test e-mail messages.
Socket Timeout	The amount of time in milliseconds before the Analytic Engine terminates an idle connection with the SMTP server.
Default Mail Encoding	Default encoding to use for e-mail that is sent to users. You can specify any MIME registered encoding name. The default value is UTF-8.
Templates	Used to specify e-mail alert templates and to assign those templates to specific rules. When a rule is violated, the e-mail alert that gets sent will be formatted based on the associated alert template. If no rule or alert template is specified, the Analytic Engine uses the default format for the e-mail alert.

Field	Description
	For more information about creating custom alert templates, see the PDF publication *Administering webMethods Optimize*.

6. Click **Test** if you want to test the SMTP server connection. This button tests the connection and sends a message to the admin e-mail address.
7. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.
8. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
9. Click **Finish** to close this page and return to the Define Environments page.

Defining Monitor Behavior Settings

Optimize uses statistical intervals to generate statistics and to evaluate rules. A *statistical interval* is a period of time from which the Analytic Engine takes collected data samples and creates common statistical values such as mean and standard deviation.

You can group the days of the week or the minutes in a day to create statistical intervals. For example, you can choose to generate a new average, mean, and standard deviation for each type of data every work day (Monday through Friday). Alternatively, you can choose to generate a new mean and standard deviation for each type of data every 12 hours.

To define Monitor behavior settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Monitor Behavior Settings**.

   A new area appears beside the configuration tree, labeled **Monitor Behavior Settings for Analytic Engine**.
5. Choose from the three options in the **Default Days** list box:

Field Option	Description
all	Choose this option to combine all seven days of the week into one group. Optimize will generate statistical values from the data collected during all seven days to use for rules.
work	Choose this option to divide the days of the week into two groups: work days (Monday through Friday) and weekend days (Saturday and Sunday). Then, for each work day, Optimize will generate statistical values from the data collected on all five work days to use for rules. For each weekend day, Optimize will generate statistical values from the data collected on the two weekend days to use for rules.
day	**day** is the default option. Choose this option to keep each day independent. Optimize will generate statistical values from the data collected on each day of the week to use for rules on that day only. For example, Optimize will generate statistical values from data collected only on Mondays to use for rules.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

8. Click **Finish** to close this page and return to the Define Environments page.

Defining Process Tracker Settings

You can set a Process Tracker attribute to control the number of days to retain process data in the Process Tracker database. Process data is data about process execution, such as an Order Process started at 10:52:31, step 1 succeeded, step 2 failed, and so on.

To define the Process Tracker setting

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

2. On the Define Environments page, click the name of the environment you want to work with.

3. On the Edit Environment page, click the **Configure Servers** tab.

4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Process Tracker Settings**.

A new area appears beside the configuration tree, labeled **Process Tracker Settings for Analytic Engine**.

5. In the **Days To Retain Process** field, specify the number of days to retain process data in the Process Tracker database. The default value for this field is 60 days.

   After the specified number of days has passed, the data is eligible to be purged by the purge operation.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this setting.

   If you click **Finish** without first clicking **Save**, any changes made to this setting will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

8. Click **Finish** to close this page and return to the Define Environments page.

Defining SNMP Alert Settings

When you define an SNMP trap and a rule is violated, the Analytic Engine sends the webMethods Alert SNMP trap alert to the SNMP manager associated with that violated rule.

Specify the SNMP managers that will receive these alerts using the SNMP alert settings. The SNMP managers that you define in the SNMP alert settings will appear on the Add / Edit / Copy Rule page when you configure alerts.

The SNMP manager can retrieve the subject and body strings according to the structure defined in the Optimize Management Information Base (MIB) file webMethods-common-mib.txt, located in the directory *Optimize_directory*\analysis\conf\MIB.

For more information about configuring SNMP alert settings, see *Administering webMethods Optimize*.

Note: To complete this procedure, you will need to obtain an encrypted SMTP community name for step 5. For more information about configuring SNMP Data Collector, see information about configuring the SNMP Data Collector in the PDF publication *Administering webMethods Optimize*.

To specify one or more SNMP managers

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

2. On the Define Environments page, click the name of the environment you want to work with.

3. On the Edit Environment page, click the **Configure Servers** tab.

4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **SNMPAlert Settings**.

5. In the text editor that appears in the SNMP alert settings area, remove the comment tags in the XML snippet to specify the following:

Setting	Description
managerName	Name of the SNMP manager that you want to receive alerts. This value is a symbolic name and can be any string denoting the SNMP manager host. This name also appears on the Add / Edit / Copy Rule page when you configure alerts.
host	Actual host name of the SNMP manager to receive the alert. Do not specify `localhost`; you must use the host name or IP address of the SNMP manager.
port	Port number of the SNMP manager to receive the alert.
community	Encrypted community name for the Analytic Engine to use to communicate with the SNMP manager. Encrypt the community name as described in Chapter 9, "Configuring and Using the SNMP Data Collector" of *Administering webMethods Optimize*. Paste the encrypted community name into the *community* attribute.

   Optimize displays the SNMP managers you identify here in the **Alerts** panel on the Add / Edit / Copy Rule page. For more information about selecting an SNMP manager on the Add / Edit / Copy Rule page, see the Optimize documentation.

6. If you want to identify more than one SNMP manager to receive alerts, copy and edit the commented XML in the text editor; add and configure a new section for each SNMP manager.

7. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

8. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

9. Click **Finish** to close this page and return to the Define Environments page.

Defining Station Configuration Settings

You use station configuration settings allow you to enable and configure data-level security (DLS) for Analytic Engines.

When data-level security (DLS) is enabled, user access to KPIs and business processes is controlled by user role. DLS is disabled by default, and when DLS is enabled, all user roles are denied access to KPIs or business processes by default. For users to have access to KPIs or business processes after DLS is enabled, it is necessary to assign access to specific user roles.

For more information about KPIs, see *Administering webMethods Optimize*.

To define station configuration settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **Station Settings**.

   A new area appears beside the configuration tree, labeled **Station Settings for Analytic Engine**.
5. In the station settings area, edit the fields as desired:

Field	Description
DLS Enabled	Enables data-level security (DLS). The default setting is `false`, or off (the check box is cleared).
DLS Cache Enabled	Enables a data-level security cache. The default setting is `false`, or off (the check box is cleared). Set to `true` (select the check box) if DLS is enabled.
SAML Enforced	Enables SAML security for Web services. The default setting is `true`, or on (the check box is selected).

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.
7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining WSAction Settings

When a rule is violated, you can have Optimize trigger a Web service action. For example, suppose that you defined a rule to determine when an adapter goes offline.

Also suppose that the Integration Server on which that adapter is running contains a service that attempts to restart adapters. You can define an action so that, when the adapter rule is violated, that action invokes a Web service that executes the Integration Server service to restart the adapter.

You also can provide optional authentication parameters for the Web service action by specifying a user login and an encrypted password.

To configure Web service actions

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the appropriate **Analytic Engine** logical server node in the configuration tree, click **WSAction Settings**.

   A new area appears beside the configuration tree, labeled **WSAction Settings for Analytic Engine**.
5. Identify the Web service to execute, along with the location of the services associated WSDL document, by editing the XML snippet in the text editor in the settings area. As installed, the XML is configured as a comment (<!-- xml -->). Remove the comment formatting and modify the XML according to the instructions in the XML.

   The XML to uncomment and edit appears in the following example.

```
<properties>
<!-- (Remove this line to uncomment the file.)
   <property name="action">
      <string meta="name">{Test Action}</string>
      <string meta="url">{host:port/path/service.wsdl}</string>
      <string meta="method">{operation}</string>
      <list>
         <!-Place parameter name and types here -->
         <element><string>{paramName1}</string></element>
         <element><string>{paramName2}</string></element>
         <element><string>{paramName...}</string></element>
      </list>
    <property name="login">
      <string meta="user">{username}</string>
      <!--  Must use a unique handle for each different password
      which is used for encryption -->
      <string meta="handle">{passwordHandle}</string>
      <string meta="password">{password}</string>
      </property>
   </property>
--> (Remove this line to uncomment the file.)
</properties>
```

For this property...	Substitute...
Test Action	A unique name to identify the action.

For this property...	Substitute...
host:port/path/service.wsdl	The host name or IP address and port number of the system on which the Web service will be executed, and the location and name of the WSDL document.
operation	The name of the method to invoke. This name should match the method name provided in the WSDL document.

The Web service method called must have a signature matching the parameters listed below.

List of Parameters...	Description...
Literal Attributes	Enter the names of these attributes literally to include this information in the action.
`RuleName`	Use this text literally to represent the string containing the name of the rule instance that was violated.
`RuleDefinition`	Use this text literally to represent the string containing the definition of the rule.
`RuleEvaluation`	Use this text literally to represent the string containing the evaluation of the rule.
`Attributes`	Use this text literally to represent an array of strings containing key/value pairs of all attributes in the rule diagnosis.
`Time`	Use this text literally to represent the time that the rule was violated, in string format.
`ProcessInfo`	Use this text literally to represent an array of strings containing information about a process, such as process name, step names, and instance IDs.
`ProcessInfo`	Array of strings containing information about a process. Use this text literally to represent an array of strings containing information about a

List of Parameters...	Description...
	process, such as process name, step names, and instance IDs.
Additional Attributes	Enter additional attributes by specifying Monitor data field names.
paramName	Allows you to specify a particular field from the underlying data of the Monitor. For example, if the rule is on business data containing a ProcessInstanceId field, you would specify `ProcessInstanceId`.
Optional Parameters	One or more parameters to pass to the Web service. These parameters should match the parameters provided in the WSDL document.
username	Specify a user name. The user name must be accompanied by a password.
passwordHandle	Specify the handle used for identifying the password specified in the password string. Use `WSActionHandle` if you will be specifying only one password for Web service action settings, or use a unique handle for each password if you will be specifying multiple passwords.
password	Specify a password. The password must be accompanied by a user name. Once this settings file is saved and processed, the password will be encrypted, and it will be displayed only as asterisks.

Optimize displays the Web service action you specify on the Add / Edit / Copy Rule page and the Rule Details page.

6. To identify more than one Web service action, copy and edit the XML in the text editor; add and configure a new section for each new Web service action.
7. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.
8. Apply the changes by restarting the Analytic Engine.

   For more information about restarting the Analytic Engine, see *Administering webMethods Optimize*.
9. To make sure the new settings work:

a. In My webMethods: **Navigate > Applications > Administration > Analytics > Rules > Rule List**.

b. On the Rule List page, click **Create Rule**.

c. In the **Actions** panel on the Add / Edit / Copy Rule page, click **Add Action**.

d. Ensure that the **Action Name** list contains the Web service actions you added to WSActionConfiguration.properties in this procedure.

   If the **Action Name** list does not contain the Web service actions you identified, check the syntax of the XML in the WS action settings area (on the **Configure Servers** tab).

Defining Logical Server Subcomponents for the Web Service Data Collector

There are three subcomponents that you must review or address when configuring settings for a Web Service Data Collector logical server. These subcomponents are as follows:

- DataCollector settings
- JNDI Configuration settings
- Journal Logging settings

You can configure the queues that the Web Service (WS) Data Collector uses to store events and dimensional data.

Defining DataCollector Settings for the Web Service Data Collector

To define DataCollector settings

1. Under the **Web Service Data Collector** node in the configuration tree (on the **Configure Servers** tab), click **DataCollector Settings**.

   A new area that displays editable fields for the selection appears beside the configuration tree, labeled **DataCollector Settings for Web Service Data Collector**.

2. Edit any data collector settings as desired:

Field	Description
jmsResendQueueSize	The value in this field controls the queue size for storing events processed by the WS Data Collector if the specified message server is not responding. The default value is 50,000 messages (displayed without the comma).

Field	Description
eventQueueSize	The value in this field determines the queue size for process-related events. The default value is 50,000 messages (displayed without the comma).
dimensionQueueSize	The value in this field determines the queue size for dimensional data. The default value is 50,000 messages (displayed without the comma).

3. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

4. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

5. Click **Finish** to close this page and return to the Define Environments page.

Defining JNDI Configuration Settings

By default the Web Service Data Collector uses the JNDI configuration settings defined for the Default Settings configuration on the Configure Servers tab. If you want to use custom settings for the Web Service Data Collector, un-check the **Use Default** check box and the complete JNDI Configuration settings will be displayed in an editable form.

If a single key and trust store are desired for the environment you can configure the JNDI Configuration under the Default Settings node in the configuration tree (on the Configure Servers tab). If you have individual certificates (Keystores) for each Optimize component you may override the default JNDI configuration by un-checking the **Use Default** check box for JNDI Configuration under the appropriate component in the configuration tree and updating the setting as appropriate.

To edit JNDI configuration settings for the Web Service Data Collector

1. Under the **Web Service Data Collector** node in the configuration tree (on the **Configure Servers** tab), click **JNDI Configuration**.

2. Un-check the **Use Default** check box to display the JNDI Configuration settings in an editable format.

3. In the JNDI configuration area of the **JNDI Configuration for Web Service Data Collector** area, the first two settings, **Broker Name** and **Naming Factory Type** are required for all systems, and the remaining settings are applicable based on your SSL needs. Refer to the following table for more information about specific JNDI Configuration settings:

Field	Description
Broker Name	If you choose "webMethods Universal Messaging" (the default) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select Broker as the JMS server in the **Naming Factory Type** field, this field should display the appropriate Broker name.
Naming Factory Type	Select the appropriate JMS server, either "Broker" or "Universal Messaging". If you choose "Universal Messaging", then **Broker Name** field should be empty. If you choose "Broker", then the **Broker Name** field must list the appropriate broker server name.
Enable SSL	Select this check box to enable a secure connection. The check box is unchecked by default and is not required.
Encryption	Select this check box to enable an encrypted connection. The check box is unchecked by default and is not required.
Key Store File	Type the absolute path to the private key file that is located on a server accessible by the component. This field is required only if you enable SSL.
Key Store Type	Click the drop down to select the Key Store Type (PKCS12 or JKS). This field is required only if you enable SSL.
Distinguished Name	Type the name of the appropriate certificate identifier. This field is required only if you enable SSL.
Trust Store File	Type the absolute path to the CA certificate file on a server location accessible by Analytic Engine. This field is required only if you enable SSL.
Trust Store Type	Click the drop down to select the Trust Store Type (PKCS12 or JKS). This field is required only if you enable SSL.

Field	Description
Key and Trust Store Password	Type the pass phrase to access the specified Key and Trust Stores. This field is required only if you enable SSL.
JMS Cluster URL	If your system uses a **Universal Messaging** cluster as the JMS provider, type the appropriate URL to identify the cluster. Note that, if provided, this setting overrides the JMS Provider URL generated from the JMS Endpoint and the information in the **Broker Name** field. This field should be coordinated with the value specified in the **Naming Factory Type** field. The format for this URL is <protocol>:// <host>:port, <protocol>://<host>:port, etc. Valid protocols are nsp, nsps, nhp, and nhps. Also, note that you must configure your **Universal Messaging** cluster in an appropriate manner for your needs and system configuration. Refer to the *webMethods Universal Messaging Clustering Guide* for more information about configuring and managing **Universal Messaging** clusters.

4. Click **Save** to save changes you have made in the JNDI configuration area, or click **Reset** to discard any changes.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

5. Click **Finish** to close the **Configure Servers** tab and return to the Define Environments page.

Defining Journal-Logging Settings

When configuring an Optimize system, you can define the common or shared journal-logging targets that you want to make available to Optimize (or its logical servers) in the environment. Each journal-logging target represents a file that can receive and store logging data of a specified type from webMethods components in the environment. By default a Web Service Data Collector logical server uses the Journal Logging settings defined for the Default Settings configuration on the Configure Servers tab. If you want to use custom settings for a Web service Data Collector, un-check the **Use Default** check box and the complete Journal Logging settings will be displayed in an editable form.

When you configure journal-logging settings for a component, you set the logging level, or threshold, and assign targets to journal loggers. Each logger receives log events from application source code and forwards those log events to all targets that are assigned to the logger as well as to all targets that are assigned to its ancestor loggers. The logger determines whether to forward each event by comparing the event's level to the logger's threshold.

To define journal-logging settings

1. Under the **Web Service Data Collector** node in the configuration tree (on the **Configure Servers** tab), click **Journal Logging**.
2. Un-check the **Use Default** check box to display the Journal Logging settings in an editable format.
3. In the **Journal Loggers for Web Service Data Collector** panel, define the threshold level for the root journal logger in the journal-logging configuration area.

   Threshold levels indicate the level of information that you want to log to a specified target. If a journal logger is not assigned a threshold level, that logger inherits the assigned level from the root logger. The seven predefined threshold levels are listed below in order of *decreasing* importance:

   - Fatal
   - Error
   - Warn
   - Info
   - Debug
   - Trace
   - None, or off

   With the exception of None (off), you can assign one of these predefined threshold levels for each target. The journal logger logs messages that are of the specified threshold level and also logs messages that exceed the specified threshold level. For example, if you specify the threshold level as Warn, the logger will log Warn, Error, and Fatal messages.

   Note: The available predefined threshold levels are determined at the product level.

4. If necessary, edit any target in the **Target Name** list by clicking the name of the target.

   In the **Add/Edit Target** area, you can change the name, description, and threshold level for that target. (For more information about threshold levels, see the preceding step.)

5. If you edit a target that is written to a file, you also can specify various rollover details related to the log file. Do one of the following:

   - To change any of these rollover details, review the following several steps.
   - To leave all of the rollover details as is, proceed to step 10.

6. In the **File** field (in the **Add/Edit Target** area on the **Configure Servers** tab), enter the name of the file to which you want the logging data to be written.
7. Use the **Rollover file by** field to indicate how often you want to close the log file and open a new log file. Choose a rollover method from the following three options:

- **Time Period only**

  Note: All times are calculated according to the logical server's local time zone.

- **File Size only**

  If you choose **File Size only**, complete the **Max File Size** field by entering the size (in megabytes) at which you want the log file to be rolled over. The file will always be rolled over at midnight in the logical server's local time zone, but if the file reaches the specified maximum file size before midnight, the file will be rolled over at that point *as well*.

- **Size or Period**

  If you choose **Size or Period**, complete both the **Period** field (see the next step for more information) and the **Max File Size** field (see the previous bullet for more information).

8. If you specify log file rollover by time period, you must also complete the **Period** field.

   When you specify log file rollover by time period, each new log file is saved and named with the current date and time using ISO 8601 format (*yyyy-mm-ww-dd-hh:mm:ss,sss*). The option specified in the **Period** field determines which date/time characters are appended to each log file name (see the following table for more information). All times are calculated according to the logical server's local time zone.

   If more than one log file is rolled over in a 24-hour period, the file names also will include a sequential marker, beginning with the characters `-1`, after the date/time characters. So, for example, the first log file rolled over in a 24-hour period would simply be named with the current date and time, using the format *yyyy-mm-ww-dd-hh:mm:ss,sss* . If a second log file is rolled over within that same 24-hour period, the characters `-1` would be added to the second file name after the date/time characters. If a third log file is rolled over within that same 24-hour period, the characters `-2` would be added at the end of the third file name, and so on.

   Select from the following options for the **Period** field:

Field	Description	Date/Time Characters Appended to Log File Name
Midnight	Creates a new log file at midnight. Entries are added to that log file up until midnight of the next day.	*yyyy-mm-dd*
Minute	Creates a new log file every minute.	*yyyy-mm-dd-hh:mm*
Hourly	Creates a new log file every hour.	*yyyy-mm-dd-hh* , where *hh* uses a 24-hour clock

Field	Description	Date/Time Characters Appended to Log File Name
Half Daily	Creates a new log file at noon and then again at midnight.	*yyyy-mm-dd* -{AM|PM}
Weekly	Creates a new log file each Sunday at midnight.	*yyyy-ww*
Monthly	Creates a new log file on the last day of each month at midnight.	*yyyy-mm*

9. In the **Max # Log Files** field, specify the maximum number of log files that you want to be retained.
10. In the **Rollover Destination** field, specify the directory where rolled over log files should be stored.

    If you choose **Directory** as your rollover destination, complete the additional fields as described in the following table.

Field	Description
Log Directory	Enter the name of the directory where the rollover log files will be saved.
Host Name	By default, this field displays the name of the host for the current target. Change the host name if needed.
Date Format	This field shows the date format that will be used for rolled over log files. No editing is needed.
Time Format	This field shows the time format that will be used for rolled over log files. No editing is needed.

11. When you are satisfied with the configuration of each target, click **Save** in the **Add/Edit Target** area to save the configurations, or click **Cancel** to discard any changes you have made.

Defining Logical Server Subcomponents for the Infrastructure Data Collector

If you are setting up an Optimize for Infrastructure system, you must configure an Infrastructure Data Collector logical server. For the Infrastructure Data Collector (InfraDC or IDC), you must configure the following subcomponents:

- Adabas SOA Gateway Resource Module settings

- Broker Resource Module settings
- Collector settings
- Com-plete Resource Module settings
- EntireX Resource Module settings
- ETS Resource Module settings
- Integration Server Resource Module settings
- My webMethods Server Resource Module settings
- Universal Messaging Resource Module settings
- JNDI Configuration settings

Define these settings to configure the queue that Infrastructure Data Collector uses to store resource data.

Defining Adabas SOA Gateway Resource Module Settings

The Adabas SOA Gateway provides KPIs for the highest and the lowest operation times. These times are calculated from the start of the Adabas SOA Gateway. If you want to have these values in one poll interval instead, you must reset the counters after each call.

Note: This only works correctly if the Adabas SOA Gateway is monitored only from a single Infrastructure Data Collector, and only if other SOA tools do not reset the counters.

To define Adabas SOA Gateway Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **Adabas SOA Gateway Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **Adabas SOA Gateway Resource Module Settings for Infrastructure Data Collector**.
5. Define the value for the **Reset** property:
   - Set to *0 - no reset* (default) to avoid resetting the Adabas SOA Gateway statistics with each call.
   - Set to *1 - reset* to reset the Adabas SOA Gateway statistics with each call.
6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining Broker Resource Module Settings

To define Broker Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **Broker Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **Broker Resource Module Settings for Infrastructure Data Collector**.
5. Define the Broker Resource Module settings that you want the logical server to use:

Field	Description
Status Poll Interval	The amount of time that passes (in seconds) before the system starts to poll the Broker Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	The interval (in seconds) between retries of polling the Broker Resource Module. The default value is 30 seconds.
Status Poll Retry Count	The number of times you want the system to retry polling the Broker Resource Module. The default value is 3 retries.
Auto Discovery Interval	The interval at which Broker auto-discovery runs.# The default value is 30 seconds.
Auto Discovery Flag	Indicates whether Broker auto-discovery is enabled. By default, auto-discovery is disabled. Broker auto-discovery is available on all Optimize Infrastructure Data Collectors, though it is disabled by

Field	Description
	default due to potential performance issues in some system configurations. When Broker auto-discovery is enabled, Brokers and related components such as document types, custom adapters, territories are automatically updated for any Broker Servers that have been discovered. Note that Broker Servers must still be discovered manually when Broker auto-discovery is enabled.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

8. Click **Finish** to close this page and return to the Define Environments page.

Defining Collector Settings

The settings in this section enable you to define the name of the data collector and the interval at which KPIs are polled. In addition, you can select infrastructure asset types for which you want to load metadata information. Controlling metadata loading improves initial startup of Infrastructure Data Collector, limits the number of tables created in the Analysis database, and improves performance of Reporting metadata synchronization.

To define collector settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

2. On the Define Environments page, click the name of the environment you want to work with.

3. On the Edit Environment page, click the **Configure Servers** tab.

4. Under the **Infrastructure Data Collector** node in the configuration tree, click **Collector Settings**.

   A new area appears beside the configuration tree, labeled **Collector Settings for Infrastructure Data Collector**.

5. Define the collector settings that you want the logical server to use. These fields also enable you to specify the asset types for which you want to load metadata information. By default, metadata load is disabled for all components, but you must select at least one component for metadata loading.

   Note: Disabling metadata load for a component after enabling does not delete metadata or data previously collected.

Field	Description
Collector Name	The name of the current Infrastructure Data Collector. The default name is InfraDC. The name of the data collector is used to route asset discovery requests. A unique name should be set during configuration. The collector name gets associated with any and all assets found during the asset discovery process. If the collector name is changed after asset discovery, all assets to which that collector name is associated will be inaccessible.
Monitor Polling Interval	The interval (in minutes) between attempts to poll monitored KPIs. The default value is 5 minutes.
Load Adabas Definitions	Indicates whether to load Adabas metadata.
Load Adabas SOA Gateway Definitions	Indicates whether to load Adabas SOA Gateway metadata.
Load Applinx Definitions	Indicates whether to load Applinx metadata.
Load Broker Definitions	Indicates whether to load Broker metadata.
Load Com-plete Definitions	Indicates whether to load Com-plete metadata.
Load EntireX Definitions	Indicates whether to load EntireX metadata.
Load Integration Server Definitions	Indicates whether to load Integration Server metadata.
Load My webMethods Server Definitions	Indicates whether to load My webMethods Server metadata.
Load Natural Ajax Definitions	Indicates whether to load Natural Ajax metadata.
Load Natural Definitions	Indicates whether to load Natural metadata.
Load SNMP Definitions	Indicates whether to load SNMP metadata.

Field	Description
Infrastructure Data Collector Log Level	Indicates the logging level for the Infrastructure Data Collector messages. Available settings are as follows: ■ OFF - Nothing is logged. ■ FATAL - Only very severe error messages are logged. ■ ERROR - FATAL and error messages are logged. ■ WARN - FATAL, ERROR, and potentially harmful messages are logged. ■ INFO - FATAL, ERROR, WARN, and informational messages are logged. ■ DEBUG - FATAL, ERROR, WARN, INFO, and fine-grained messages are logged. ■ TRACE - FATAL, ERROR, WARN, INFO, DEBUG, and even finer-grained messages are logged. ■ ALL - All types of messages are logged.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining Com-plete Resource Module Settings

To define Com-plete Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **Com-plete Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **Com-plete Resource Module Settings for Infrastructure Data Collector**.

5. Define the value for the **Name** property.

The possible values are described in the following table.

Value	Description
`0 - instance name is jobname (product) and port (discovery)`	Default. Job name as defined in the environment and port number as specified during discovery.
`1 - instance name is CompleteName (product) and port (discovery)`	Com-plete name as defined in the environment and port number as specified during discovery.
`2 - instance name is jobname (product)`	Job name as defined in the environment.
`3 - instance name is CompleteName (product)`	Com-plete name as defined in the environment.
`4 - instance name is port (discovery)`	Port number as specified during discovery.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.
7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining EntireX Resource Module Settings

To define EntireX Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **EntireX Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **EntireX Resource Module Settings for Infrastructure Data Collector**.
5. Define the value for the **Name** property.

The possible values are described in the following table.

Value	Description
`0 - exxbrokername is port (discovery)`	EntireX broker name is port number as specified during discovery.
`1 - exxbrokername is brokername (product)`	EntireX broker name is broker name as defined for the product.
`2 - exxbrokername is port (discovery) and brokername (product)`	EntireX broker name is both port number as specified during discovery and broker name as defined for the product.
`3 - exxbrokername is brokername (product) and port (discovery)`	EntireX broker name is both broker name as defined for the product and port number as specified during discovery.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.
7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining ETS Resource Module Settings

To define ETS Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **ETS Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **ETS Resource Module Settings for Infrastructure Data Collector**.
5. Define the value for the **Hostname** property.

   The possible values are described in the following table.

Value	Description
`0 - hostname is hostname (discovery)`	Default. Host name is host name as specified during discovery.
`1 - hostname is hostname (product)`	Host name as defined in the environment.
`2 - hostname is sysplexname (product)`	Host name is sysplex name as defined in the environment.
`3 - hostname is sysplexname (product) and hostname (product)`	Host name is both sysplex name and host name as defined in the environment.
`4 - hostname is sysplexname (product) and hostname (discovery)`	Host name is sysplex name as defined in the environment and host name as specified during discovery.

6. Define the value for the **Trace Level** property.

   The possible values are described in the following table.

Value	Description
`0 - fatal`	Log critical errors. (Currently not used.)
`1 - error`	Log error messages.
`2 - warn`	Log warnings.
`3 - info`	Log a summary information line for each call to the Infrastructure Data Collector.
`4 - debug`	Default. Log debug information for each call to the Infrastructure Data Collector. Not used by the Adabas and Natural Data Collectors.
`5 - trace`	Write trace data.
`6 - verbose-1`	Generic name for a product-specific trace level.

Value	Description
`7 - verbose-2`	Generic name for a product-specific trace level.
`8 - verbose-3`	Generic name for a product-specific trace level.
`9 - verbose-4`	Generic name for a product-specific trace level.
`10 - verbose-5`	Generic name for a product-specific trace level.

7. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

8. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
9. Click **Finish** to close this page and return to the Define Environments page.

Defining Integration Server Resource Module Settings

To define Integration Server Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **IS Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **IS Resource Module Settings for Infrastructure Data Collector**.

5. Define the Integration Server (IS) Resource Module settings that you want the logical server to use.

   The settings are described in the following table.

Field	Description
Status Poll Interval	This is the amount of time that passes (in seconds) before the system starts to poll the IS Resource Module.

Field	Description
	The default value is 30 seconds.
Status Retry Interval	This is the interval (in seconds) between retries of polling the IS Resource Module. The default value is 18 seconds.
Status Poll Retry Count	This is the number of times you want the system to retry polling the IS Resource Module. The default value is 3 retries.
Notification Poll Interval	This is the interval at which polling for Integration Server notifications is performed. The default value is 60 seconds.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining My webMethods Server Resource Module Settings

To define My webMethods Server Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.
4. Under the **Infrastructure Data Collector** node in the configuration tree, click **MWS Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **MWS Resource Module Settings for Infrastructure Data Collector**.

5. Define the MWS Resource Module settings that you want the logical server to use.

   The settings are described in the following table.

Field	Description
Status Poll Interval	This is the amount of time that passes (in seconds) before the system starts to poll the MWS Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	This is the interval (in seconds) between retries of polling the MWS Resource Module. The default value is 30 seconds.
Status Poll Retry Count	The number of times you want the system to retry polling the MWS Resource Module. The default value is 3 retries.
Connection Timeout	The interval (in seconds) after which the connection will timeout. The default value is 60 seconds.
Auto Discovery Interval	The interval at which MWS auto-discovery runs. The default value is 6 minutes.
Auto Discovery Flag	Indicates whether MWS auto-discovery is enabled/disabled. By default, auto discovery is enabled.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.
8. Click **Finish** to close this page and return to the Define Environments page.

Defining Universal Messaging (UM) Resource Module Settings

To define Universal Messaging Resource Module settings

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**
2. On the Define Environments page, click the name of the environment you want to work with.
3. On the Edit Environment page, click the **Configure Servers** tab.

4. Under the **Infrastructure Data Collector** node in the configuration tree, click **UM Resource Module Settings**.

   A new area appears beside the configuration tree, labeled **UM Resource Module Settings for Infrastructure Data Collector**.

5. Define the UM Resource Module settings that you want the logical server to use.

   The settings are described in the following table.

Field	Description
Status Poll Interval	This is the amount of time that passes (in seconds) before the system starts to poll the UM Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	This is the interval (in seconds) between retries of polling the UM Resource Module. The default value is 30 seconds.
Status Poll Retry Count	The number of times you want the system to retry polling the UM Resource Module. The default value is 3 retries.
Connection Timeout	The interval (in seconds) after which the connection will timeout. The default value is 60 seconds.
Auto Discovery Interval	The interval at which UM auto-discovery runs. The default value is 6 minutes.
Auto Discovery Flag	Indicates whether UM auto-discovery is enabled/disabled. By default, auto discovery is enabled.

6. Click **Save** to save changes or click **Cancel** to discard any changes you have made in this settings area.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

7. To configure another logical server subcomponent, click the name of that subcomponent in the configuration tree.

8. Click **Finish** to close this page and return to the Define Environments page.

Defining JNDI Configuration Settings

By default the Infrastructure Data Collector uses the JNDI configuration settings defined for the Default Settings configuration on the Configure Servers tab. If you want to use custom settings for the Infrastructure Data Collector, un-check the **Use Default** check box and the complete JNDI Configuration settings will be displayed in an editable form.

If a single key and trust store are desired for the environment you can configure the JNDI Configuration under the Default Settings node in the configuration tree (on the Configure Servers tab). If you have individual certificates (Keystores) for each Optimize component you may override the default JNDI configuration by un-checking the **Use Default** check box for JNDI Configuration under the appropriate component in the configuration tree and updating the setting as appropriate.

To edit JNDI configuration settings for Infrastructure Data Collector

1. Under the **Infrastructure Data Collector** node in the configuration tree (on the **Configure Servers** tab), click **JNDI Configuration**.
2. Un-check the Use Default check box to display the JNDI Configuration settings in an editable format.
3. In the JNDI configuration area of the **JNDI Configuration for Environment Default Settings** area, the first two settings, **Broker Name** and **Naming Factory Type** are required for all systems, and the remaining settings are applicable based on your SSL needs. Refer to "Defining JNDI Configuration Settings" on page 38 in the section on Configuring Default Settings for Logical Servers for more information about specific JNDI Configuration settings.:

Field	Description
Broker Name	If you choose “webMethods Universal Messaging” (the default) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select Broker as the JMS server in the **Naming Factory Type** field, this field should display the appropriate Broker name.
Naming Factory Type	Select the appropriate JMS server, either “Broker” or “Universal Messaging”. If you choose “Universal Messaging”, then **Broker Name** field should be empty. If you choose “Broker”, then the **Broker Name** field must list the appropriate broker server name.
Enable SSL	Select this check box to enable a secure connection. The check box is unchecked by default and is not required.

Field	Description
Encryption	Select this check box to enable an encrypted connection. The check box is unchecked by default and is not required.
Key Store File	Type the absolute path to the private key file that is located on a server accessible by the component. This field is required only if you enable SSL.
Key Store Type	Click the drop down to select the Key Store Type (PKCS12 or JKS). This field is required only if you enable SSL.
Distinguished Name	Type the name of the appropriate certificate identifier. This field is required only if you enable SSL.
Trust Store File	Type the absolute path to the CA certificate file on a server location accessible by Analytic Engine. This field is required only if you enable SSL.
Trust Store Type	Click the drop down to select the Trust Store Type (PKCS12 or JKS). This field is required only if you enable SSL.
Key and Trust Store Password	Type the pass phrase to access the specified Key and Trust Stores. This field is required only if you enable SSL.
JMS Cluster URL	If your system uses a **Universal Messaging** cluster as the JMS provider, type the appropriate URL to identify the cluster. Note that, if provided, this setting overrides the JMS Provider URL generated from the JMS Endpoint and the information in the **Broker Name** field. This field should be coordinated with the value specified in the **Naming Factory Type** field. The format for this URL is <protocol>://<host>:port, <protocol>://<host>:port, etc. Valid protocols are nsp, nsps, nhp, and nhps. Also, note that you must configure your **Universal Messaging** cluster in an appropriate manner for your needs and system configuration. Refer to the *webMethods Universal Messaging Clustering Guide* for more information about configuring and managing **Universal Messaging** clusters.

4. Click **Save** to save changes you have made in the JNDI configuration area, or click **Reset** to discard any changes.

   If you click **Finish** without first clicking **Save**, any changes made to these settings will be lost.

5. Click **Finish** to close the **Configure Servers** tab and return to the Define Environments page.

Defining Host Servers for an Environment

A *host server* is a physical server to which you want to deploy an environment configuration. You can define one or more host servers for each environment configuration.

To define a host server

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the list of environments, click the **Edit** icon () in the same row as the environment that you want to define or edit. You can also click the name of an environment to edit that environment.

   The Edit Environment page displays a group of configuration tabs.

3. Click the **Define Hosts** tab.

   The **Define Hosts** tab displays a list of available physical servers. If no hosts are defined, the list is empty.

4. Click **Add Host**.

5. In the **Add/Edit Host** dialog box, enter a unique display name for the host. Also enter the host name or IP address for the host server.

6. Click **OK**.

   The host server appears in the list of servers on the **Define Hosts** tab.

The next step in configuring an environment is to map each logical server to a host server.

Mapping Logical Servers

When you configure an environment, one step is mapping each logical server to a host server. In addition, you can map one logical server to multiple host servers.

To map a logical server to a host server

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the list of environments, click the **Edit** icon ( ) in the same row as the environment that you want to define or edit. You can also click the name of an environment to edit that environment.

   The Edit Environment page displays a group of configuration tabs.

3. Click the **Map Servers** tab.

   The **Map Servers** tab displays a list of the logical servers in the environment. It also displays the number of hosts to which each logical server is mapped for deployment. If a logical server is not mapped, a red circle icon ( ) appears in the **Mapped** column. If a logical server is mapped, a green check mark icon ( ) appears in the **Mapped** column.

4. To map a logical server to a host, click the **Edit** icon ( ) in the **Actions** column.

   The **Edit Host Mapping** dialog box displays a list of hosts that are available for mapping and a list of hosts that already are mapped to the logical server.

   Alternatively, you can click the **Map All** button to map all hosts to logical servers.

5. Select one or more hosts from the **Available Hosts to Map** list and use the arrow buttons between the list boxes to move hosts to the **Mapped Hosts to This Logical Server** list.

6. Click **Save**.

7. Repeat steps 4-6 until each logical server is mapped to at least one host server.

   Note: Leaving a logical server unmapped results in an invalid environment.

The next step in configuring an environment is to map the incoming and outgoing connections for each logical server.

Mapping Endpoints

Endpoint connections represent the incoming and outgoing connections for each logical server subcomponent. The endpoint connections that are available for a given logical server subcomponent are specified in a definition file on that logical server.

When an environment configuration is deployed, a configuration file containing the server mappings and the endpoint connections goes out to each logical server in the environment.

To map the incoming and outgoing connections for logical server subcomponents

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the list of environments, click the **Edit** icon (in the same row as the environment that you want to define or edit. You can also click the name of an environment to edit that environment.

   The Edit Environment page displays a group of configuration tabs.

3. Click the **Map Endpoints** tab.

   The **Map Endpoints** page displays a list of the logical servers in the environment, along with the subcomponents of each logical server. The default values are displayed for the incoming connections and the outgoing connections for each subcomponent.

   Note: The JMS provider for logical servers must match the JMS provider specified on the JNDI Configuration page. For example, if you choose "Broker" as the **Naming Factory Type** on the JNDI Configuration page, then on Endpoint Configuration panel on the Map Endpoints page, the protocol of **JMS Provider** must be "Broker". If you choose "webMethods Universal Messaging" as the **Naming Factory Type**, then on the Endpoint Configuration panel, the protocol of **JMS Provider** must be nsp,nsps, nhp, or nhps.

4. If needed, edit the incoming connections for the affected subcomponents.

   Important: If more than one subcomponent of the same type is planned for deployment to the same host, make sure each port number is unique for the incoming connections of the same subcomponent type. For example, if the configuration agent for the Analytic Engine and the configuration agent for the JMS Server are planned for deployment to the same host, make sure each port number is unique for the incoming connections for both configuration agents. The environment configuration will be valid only if each port number is unique for both configuration agents. If you accept the default port numbers, those numbers will be unique.

5. If needed, edit the outgoing connections for the affected subcomponents.

   The drop-down list for each outgoing connection shows the available options.

6. Click **Save**.

The next step in configuring an environment is to map the database pools for each database component.

Note: Configuration agent credentials are also defined from the **Map Endpoints** tab. For more information about configuration agent credentials and authentication, see "Security" on page 101.

Mapping Database Pools

Specify the database pools to use for each database component in the environment. The database connection pools are defined on the **Navigate > Applications > Administration > System-Wide > Environments > Database Pool Configuration** page. See "Database Pool Configuration" on page 25, for information about defining the connection pools. You can associate the appropriate functions or logical server subcomponents to database connection pools from the **Map DB Pools** tab.

To map database pools

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the list of environments, click the **Edit** icon (in the same row as the environment that you want to define or edit. You can also click the name of an environment to edit that environment.

   The Edit Environment page displays a group of configuration tabs.

3. Click the **Map DB Pools** tab.

   The **Map DB Pools** tab displays a list of database components in the environment.

4. Select a pool for each database component using the appropriate **Pool** drop-down list. Be sure to associate each database component to the appropriate connection pool.

   For example, assume that the four database connection pools for Optimizeare named `Analysis`, `MwS` (My webMethods Server), `ProcessAudit`, and `ProcessTracker`. Using these four database pools, you would assign the database components as shown in the following table.

Database Component	Database Pool
`analysis.engine` - Analytic Engine	`Analysis`
`common.directory` - Analytic Engine	`MwS`
`process.history` - Analytic Engine	`ProcessTracker`
`process.model` - Analytic Engine	`ProcessAudit`
`process.work` - Analytic Engine	`ProcessTracker`

5. Click **Save**.

The next step is to validate the environment configuration.

Validating an Environment Configuration

After an environment is configured and each configuration tab (except for the **Validate** tab) displays a green check mark (✓), that environment configuration is ready for validation. Validation ensures that the environment configuration conforms to the rules specified in the definition templates. During validation, each logical server is checked to ensure that all of the required configurations are present in the model and that all endpoints can be resolved.

To validate an environment configuration

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. In the list of environments, click the **Edit** icon (in the same row as the environment that you want to validate. You can also click the name of an environment to validate that environment.

   The Edit Environment page displays a group of configuration tabs.

3. Click the **Validate** tab.

   The **Validate** tab indicates whether the environment configuration is valid. If validation fails, the **Validate** tab lists the problems found in the environment configuration.

   Note: webMethods Central Configuration does not validate the logical server subcomponent settings. If there is a configuration error in the subcomponent settings, an error is generated by the logical server subcomponent after the configuration is deployed. Such a logical-server-subcomponent error is not displayed by webMethods Central Configuration.

Deploying an Environment

After an environment has been configured and successfully validated, you can either deploy that environment configuration to the logical servers or save that environment configuration to a file.

When an environment configuration is deployed to the logical servers, Central Configuration pushes the configuration details out to the specified subcomponents. You have the option to deploy all configuration files to all logical servers in an environment or to deploy only the modified configuration files to the affected logical servers in the

environment. You also can deploy a configuration to a file; see "Deploying to a File" on page 97.

When you deploy a configuration to an environment for the first time, the logical servers automatically start using that configuration. If you edit a configuration and deploy only the updates, the logical servers in the environment must be restarted for the new configuration settings to take effect. In fact, regardless of whether you deploy only updates or deploy all of an edited configuration, the logical servers in the environment must be restarted for the new configuration settings to take effect.

If you want to use the Analytic Engine in Static DB Schema mode, when you deploy an environment configuration for the first time, you must make sure that:

- The **Disable DDL Statements** check box is cleared. For more information, see "Defining Database Settings" on page 50.
- The analysis database pool is configured with a DB user that has the required permissions to execute DDL statemens. For more information, see "Mapping Database Pools" on page 94.

This will ensure that the intrinsic metrics will be correctly created and populated.

Note: Errors generated during deployment are written to the My webMethods log file, located in *Software AG_directory*/*My webMethods Server_directory*/server/default/logs. If an error is generated during this time, review the log file for additional information about the error.

To deploy an environment

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. Click the **Deploy** icon in the **Actions** column.

   The Deploy Configuration Files page displays.

3. Click **Deploy All** to deploy all configuration files to all logical servers in an environment, or click **Deploy Updates** to deploy only the modified configuration files to the affected logical servers in the environment.

   The status of the deployment operation appears after the operation is completed. The status includes a list of the files that were deployed to the environment and also lists any errors that occurred.

   Note: If you edit a configuration and deploy only the updates, the logical servers in the environment must be restarted for the new configuration settings to take effect.

To deploy an environment configuration to a file, see "Deploying to a File" on page 97.

Deploying to a File

If you deploy an environment configuration to a file, the configuration settings are written to a local archive file. This makes it possible to copy configuration data to disk and import that configuration data to any server from which a direct network connection is not accessible.

Deploying the Environment Configuration to an Archive File

Deploying your environment configuration to an archive file provides a way to back up or store the configuration, or to commit the logical server configurations to source control.

To deploy the configuration to an archive file

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. Click the **Deploy** icon in the **Actions** column.

3. On the Deploy Configuration Files page, click **Deploy to File**.

   The archive file has a .zip file name extension. The .zip file contains a configuration file for each logical server in the environment.

4. Save the .zip file.

Applying Configuration Settings from the Environment Configuration Archive

If you deploy an environment configuration to an archive file, you can later apply the configuration settings to other servers.

Important: You need to stop the logical servers to which you will apply the new configuration settings and then start them after the procedure is completed.

To apply the configuration settings from one logical server to another

1. Extract the contents of the archive file.

   The extracted folder contains a contents.html file, as well as the configuration settings grouped in sub-folders under the internal identifier for each server.

2. Open the contents.html file.

   It contains links to the configuration folders for each logical server in the environment.

3. Click the link for the source server to open the folder containing its configuration settings.

4. Copy the contents of the config folder and paste them into the configuration folder for the destination server.

Transferring Logical Server Passwords

Passwords for logical servers are stored in the Password Manager repository. If you have set up Central Configuration to use SSL, you should transfer any updated passwords. If any passwords have been updated since the archive file was created for an environment configuration, transfer those updated passwords to the Password Manager repository in the appropriate configuration engine. For example, if there are any new database pool passwords for the Analytic Engine, transfer those new passwords to the Password Manager repository in the Analytic Engine.

The Password Manager repository is created using the webMethods Password Administrator utility, which is installed with My webMethods. For more information about the webMethods Password Administrator utility, see "Security" on page 101.

To transfer passwords, open the archive file for the environment configuration and locate the Password Manager repository in the /config directory. Use the webMethods Password Administrator utility to transfer the passwords.

To transfer passwords with the webMethods Password Administrator utility

1. Start the webMethods Password Administrator utility:
   - On Windows systems, run `password_admin.bat`.
   - On UNIX systems, run `password_admin.sh`.

   The webMethods Password Administrator starts in a command-line window.
2. To transfer the passwords, enter the following command:

   ```
   password-admin -t deployed-passman repository/optimize
   ```

   Where *repository* is the path to the Password Manager file. Note that the default Password Manager file for Optimize is located in the directory optimize/conf/security/passman/.

For more information about setting up Central Configuration to use SSL and the webMethods Password Administratorutility, see "Security" on page 101.

Exporting and Importing an Environment Configuration

webMethods Central Configuration allows you to export and import complete environment configurations from one My webMethods Server installation to another. Each environment configuration includes host server(s), database pools, mappings of both endpoints and database pools, and logical server configurations.

You also can use exporting and importing to archive environment configurations for backup or storage, or to commit an entire environment configuration to source control.

Exporting an Environment Configuration

To export an environment configuration

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. Select the check box beside one or more environments that you want to export, and then click **Export Environment**.

   A standard download dialog box appears.

3. Enter a file name and location for the exported environment, and then click **Save**.

   The exported environment is saved as an XML file.

Importing an Environment Configuration

To import an environment configuration

1. In My webMethods: **Navigate > Applications > Administration > System-Wide > Environments > Define Environments**

   The Define Environments page displays a list of defined environments and their configuration and deployment statuses.

2. Click **Import Environment**.

   The **Import Environments** dialog box appears.

3. Enter the file name or browse to the location of an exported environment file, and then click **OK**.

   Central Configuration imports the environment and displays the environment name in the list of environments on the Define Environments page.

5 Security

Overview

This chapter describes how to use Central Configuration with SSL (Secure Sockets Layer) to ensure that data is transmitted privately and that the content of the data is not altered during transit.

Using SSL with Central Configuration

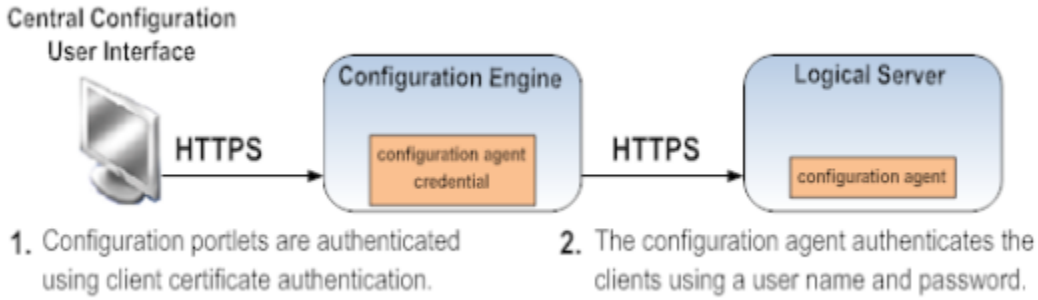
Central Configuration supports the use of SSL to secure and encrypt communications between the configuration agents and the configuration engine as well as between the configuration system's own internal communications. All that is needed to enable SSL is to obtain and install certificates in the webMethods glueKeyStore and glueTrustStore of the various components and to change their configuration protocol to HTTPS.

If you are configuring SSL for use in production, Software AG recommends that you secure access to the Central Configuration administrator files. Software AG also recommends that you change the master encryption password. All of these tasks are explained in this chapter.

Central Configuration Security Implementation

To encrypt communications between each configuration agent on each BAM logical server and the configuration engine, the certificates installed during SSL configuration are used to negotiate the SSL handshake. In addition, the certificates are used between the Central Configuration UI components and the Central Configuration Web application to provide client certificate authentication. All other authentication between Central Configuration and the BAM logical servers' configuration agents is done through simple user name and password authentication.

Central Configuration and SSL implementation



If you want to install certificates for SSL the instructions are outlined in "Installing Certificates" on page 103.

Installing Certificates

You can add the authentication credentials for Central Configuration portlets by obtaining and installing certificates on the configuration engine and configuration agents. For information and instructions on installing KeyStore and TrustStore certificates, refer to *Administering My webMethods Server*.

Securing the Central Configurator in My webMethods Server 8.0

Communications between the configuration portlets and back-end web-application can be secured even if the primary My webMethods Server instance is not. This is because the Central Configuration web-application creates its own Glue based HTTP server. This server can be configured to require the use of SSL for its in bound connections. To enable SSL in the Central Configurator, several configuration files must be changed manually. There are no options to change these defaults within the **My webMethods** user interface.

Note: There is a problem with the value of the java.net.ssl.trustStore property being set wrong prior to the Glue server initialization. Glue must be initialized with the settings from the configuration file, or the Central Configuration web application will not be able to start the HTTP(S) server.

To secure the Central Configurator in My webMethods Server 8.0:

1. Open the <*install directory*> \MWS\server\default\config\glue\glueSSLConfiguration.xml.
2. Locate the SSL properties section of the file and update it as appropriate for your system configuration.

   A default Key and Trust store is installed with the Central Configurator. These are the glueKeyStore.jks and glueTrustStore.jks files. By default, the KeyStore is empty and the TrustStore is simply a copy of the standard Java cacerts file. You can use these stores for the SSL configuration; however, it may be easier to simply provide a PKCS12 based key store and JKS trust store rather than to update these defaults.
3. Save the glueSSLConfiguration.xml file and close it.
4. Open the <*install directory*> \82\MWS\server\default\config\engine \GlueServiceRegistryProperties.xml file.
5. Locate the web service protocol configuration, and update it from “http” to “https”.
6. Save and close the GlueServiceRegistryProperties.xml file.
7. Restart My webMethods Server and examine the log to verify that it started cleanly.

You can verify that the Configuration portlets are using the https protocol for calls to the back end by looking at the ServiceRegistry.xml properties file in <*install directory*> \82\MWS\server\default\config\engine. On start up the CCS web application writes the locations of its web services endpoints to this file. Verify that each WSDL location listed in this file specifies the https protocol in its URL.

Securing the Central Configurator in My webMethods Server 8.2

The preceding section covered how to secure the Configurator in the 8.0.x version ofMy webMethods Server. In that case, the Configurator laid down default key and trust stores as well as a configuration file. The process of securing the Configurator in My webMethods Server 8.2 is similar to the 8.0.x version with the exception of how the SSL key and trust stores are specified in My webMethods Server. The SSL key and trust stores are now configured through My webMethods Server server.properties batch or shell script.

The "server.properties.bat" (Windows) or "server.properties.sh" (Unix) files contain the environment settings for the My webMethods Server instance. In this file the JVM, debug, SSL, JMX, HTTP and other options are specified. For SSL, the key and trust stores. variables define the key and trust store file locations, their type, and finally the access password. Note that once the My webMethods Server instance is started or restarted the password variables values will be encrypted. For more information see *Administering My webMethods Server*.

To secure the Central Configurator in My webMethods Server 8.2:

1. In Windows, open the <*install directory*> \\821\MWS\server\default\bin \server.properties.bat file. If you have a Unix-based system, the filename will be server.properties.sh.
2. Edit the appropriate section of the file based on the example that follows. Note that the paths and values shown may not be applicable to your system configuration:

```
set JAVA_KEYSTORE=d:\SoftwareAG\821\Certificates\ccs.p12

set JAVA_KEYSTORETYPE=pkcs12

set JAVA_KEYSTORE_PASSWORD=manage

set JAVA_TRUSTSTORE=d:\SoftwareAG\821\sagCA.jks

set JAVA_TRUSTSTORETYPE=jks

set JAVA_TRUSTSTORE_PASSWORD=manage
```

3. Save your changes to the serverproperties.xml file and close it.
4. Open the <*install directory*> \82\MWS\server\default\config\engine \GlueServiceRegistryProperties.xml file.
5. Locate the web service protocol configuration, and update it from “http” to “https”.
6. Save and close the GlueServiceRegistryProperties.xml file.
7. Restart My webMethods Server and examine the log to verify that it started cleanly.
8. Open the <*install directory*> \optimize\<*component*> \conf\system\`EndpointRegistry.xml` file in an appropriate text editor. Note that <component> should be replaced with the Optimize component for which SSL is being configured.

9. Edit the Configuration Agent protocol entry to be “https”.

   To save time, you can change the protocol for all applicable web services at the same time, if it makes sense for your situation.

There are several ways to verify SSL configuration. If you are running the Analytic Engine as a console application on a Windows server, you can check the console window. You should see the following messages in this window.

```
[STARTUP] Glue 8.0 Fix 2 build 3
[STARTUP] soap/http server started on https://<server name and
domain>:15000/services
[STARTUP] soap/http server started on https://<server name and
domain>:12503/services
```

Starting the Configuration Agent

The Configuration Agent HTTP server should be started using the “https” protocol on port 15000. Also, you can verify the server and web service by opening the following WSDL file in a browser.

`https://<server name and domain>:15000/services/RemoteConfiguration.wsdl`

The browser should display a warning message about the certificate but allow you to load the page. If the page loads, the SSL configuration was successful.

webMethods Password Administrator Utility

The Central Configuration installation includes the webMethods Password Administrator utility. The utility is located in the *Software AG_directory*/MWS/ccs/tools directory.

The webMethods Password Administrator utility can be used to change the master encryption password and to secure 8.0 Optimize components.

In order to use the webMethods Password Administrator utility, the utility files must first be extracted and the `LIB_HOME` and `JAVA_HOME` CLASSPATHs must be set.

To prepare the webMethods Password Administrator utility

1. Navigate to the *Software AG_directory*/MWS/ccs/tools directory.
2. Extract `ccs-admin.zip` into any location to decompress the webMethods Password Administrator utility.

   When you decompress the .zip file, the ccs-admin/bin and /lib directories get created.
3. Open the webMethods Password Administrator utility in a text editor and set the following CLASSPATHS:
   - `LIB_HOME`. Set to the LIB directory, ccs-admin/lib directory.
   - `JAVA_HOME`. Set to the Java root installation directory.
4. Save and close the file.

Backward Compatibility

The default security configuration is not backward compatible with Optimize components that are earlier than version 8.0. To support earlier releases, enable SSL and export the older default Glue certificates. Once exported configure the SSL settings in the My webMethods server as illustrated in the above section.

Customizing the Default Security Configuration

The following sections provide information about customizing the default security configuration.

Changing the Master Encryption Password

The Password Administrator repository includes a set of files that contain security information. The master encryption password is stored in two files: `model.ds`and `model.mpw`. Use the Password Administrator utility to change the master encryption password in each file.

Before using the webMethods Password Administrator utility ensure that the utility files are extracted and the `LIB_HOME` and `JAVA_HOME` CLASSPATHs are set. See the task steps in this section for instructions.

Note: The master encryption password should be changed immediately after the SP2 installation, before environments are created and configured. If a 7.0 or 7.0 SP1 installation was migrated to SP2 and the master encryption password is changed by the administrator, the environments should be recreated and passwords updated in the system.

To change the master encryption password

1. Start the webMethods Password Administrator utility. On Windows systems, run `password_admin.bat`. On Unix systems, run `password_admin.sh`.

   The webMethods Password Administrator opens a command line window.

   Note: When using the password_admin tool to change the master password, the repository is referenced simply as "model".

2. Enter the following command:

   ```
   password_admin -l model
   ```

3. When prompted, enter the encryptionMaster password:

   ```
   manage
   ```

4. To change the password, enter the following command:

```
password_admin -u model -h encryptionMaster -p NewPassword
```

Replace *NewPassword* with a new master encryption password.

The master encryption password is updated.

Securing Access to the Central Configuration Administrator Files

The Central Configuration installation includes the webMethods Password Administrator utility. The utility is contained in the ccs-admin.zip file that is located in the *Software AG_directory*/MWS/CCS/Tools directory.

The ccs-admin.zip file contains password administrator utilities that can be a potential vulnerability. As a good security practice it is recommended that access to the ccs-admin.zip file is secured by either removing the file and placing it in a secure location or by denying access to the file using OS access permissions.

A Optimize Installation Checklist

This list describes the minimum information required for a successful default installation and configuration of Optimize for Infrastructure or Optimize for Process. It is intended for general information only and is only a supplement to *Installing webMethods and Intelligent Business Operations Products*. See *Installing webMethods and Intelligent Business Operations Products* for up-to-date, detailed installation information.

Before you begin a default Optimize installation and configuration, be sure you have the information listed in the following table.

Component	Default Server:Primary Port
Analytic Engine	<*configuration agent*>:15000
webMethods Broker	<*host name or IP address*>:6849
Infrastructure Data Collector	<*host name or IP address*>:6666
Infrastructure Data Collector, configuration agent	<*configuration agent*>:15005
My webMethods Server (MwS)	<*host name or IP address*>:8585
WS Data Collector	<*configuration agent*>:15001

It's also helpful to have on hand a copy of *Installing webMethods and Intelligent Business Operations Products* and *Administering webMethods Optimize*.

Database Information

A typical implementation of Optimize involves placing the five Optimize database component sets - Analysis, Central Configuration System (CCS), Process Tracker, Process Audit Log, and My webMethods Server (MwS) - in five database schemas. For more information, see the chapter titled "Installing Optimize Database Component Sets" in *Administering webMethods Optimize* before you install.

Index

R

S

T

U

V

W