Configuring BAM

Version 10.3

October 2018

WEBMETHODS

This document applies to webMethods Optimize 10.3 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-103-20200422

Table of Contents

Introduction to Configuring Business Activity Monitoring (BAM)

The topics in Configuring Business Activity Monitoring (BAM) describe how to configure and deploy webMethods Business Activity Monitoring (BAM) component configurations using the webMethods Central Configurator tool. These topics provide information for administrators who want to initialize or edit common configuration settings and deploy them out to one or more webMethods environments.

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

Concepts

Overview

The following topics provide information about:

- the concepts behind the webMethods **Define Environments** page, sometimes referred to as the webMethods Central Configurator tool, that is used to configure an Optimize BAM system
- the architecture of the webMethods BAM configuration implementation
- the process by which configuration instances are defined and deployed out to a webMethods Optimize environment
- 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.

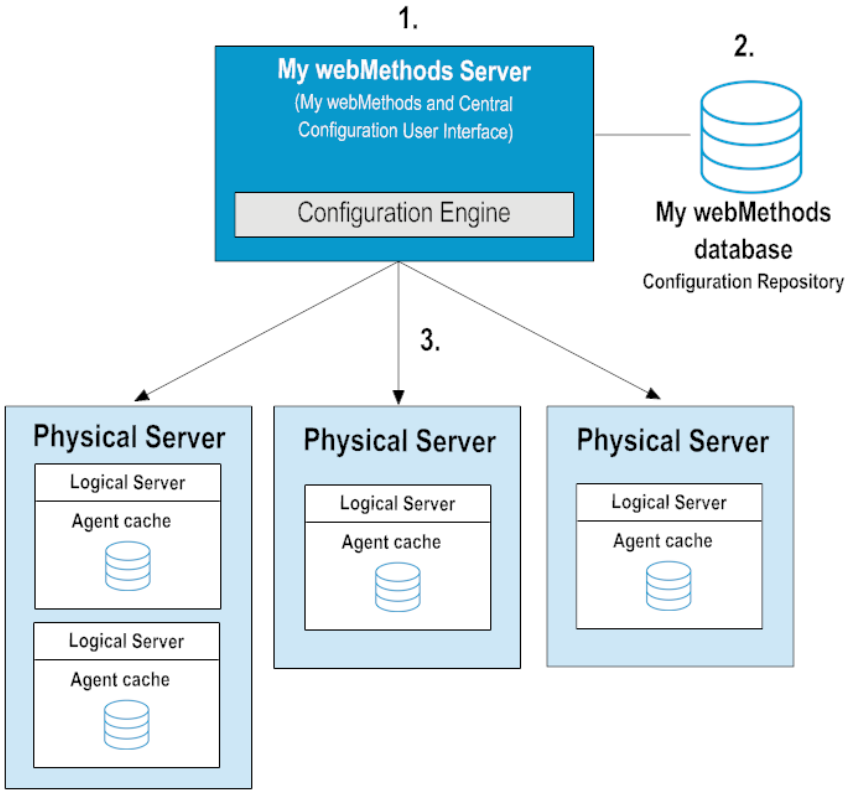
BAM Configuration Components

- **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.

Figure 1. Central Configurator Overview



The following table describes the configuration steps.

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

Step	Description
	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 following table lists the webMethods components and subcomponents that the My webMethods BAM configuration functionality supports.

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 ■ Database settings ■ Data maintenance settings

webMethods Product	Supported Logical Servers	Supported Logical Server Subcomponents
		■ Event Publication settings ■ JMSEventAction 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 ■ Apama 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 ■ JNDI configuration ■ MashZone NextGen resource module settings

webMethods Product	Supported Logical Servers	Supported Logical Server Subcomponents
		■ My webMethods Server resource module settings ■ Presto resource module settings ■ Terracotta resource module settings ■ Universal Messaging Cluster resource module settings ■ Universal Messaging resource module settings
Integration Server	Informational only. Integration Servers provide an endpoint connection to the Analytic Engine.	
JMS Server	Informational only. The JMS Server provides an endpoint connection to the Analytic Engine.	
My webMethods Server	Informational only. My webMethods Server provides an endpoint connection to the Broker Server (deprecated).	

Environments

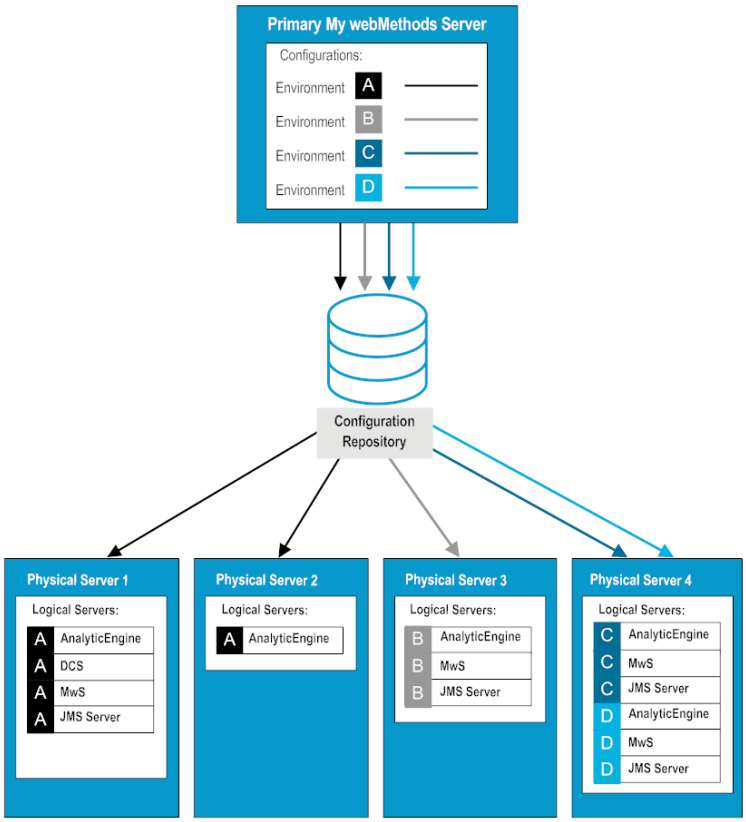
To link together several webMethods product components that you want to manage as a group, you use the My webMethods **Define Environments** page to define a webMethods environment. An *environment* is a group of webMethods product components that share configuration settings, such as cache settings.

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.

Figure 2. Example Environment Implementation



An environment also includes a set of configuration files for each logical server. To learn more about defining and configuring an environment, see "Getting Started" on page 11.

Getting Started

The following topics provide overall information about:

- how to use the My webMethods **Define Environments** page to configure and deploy an Optimize BAM environment,
- how to use the **Define Environments** tabs, and
- 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 Software AG Products* for more information about installing My webMethods.

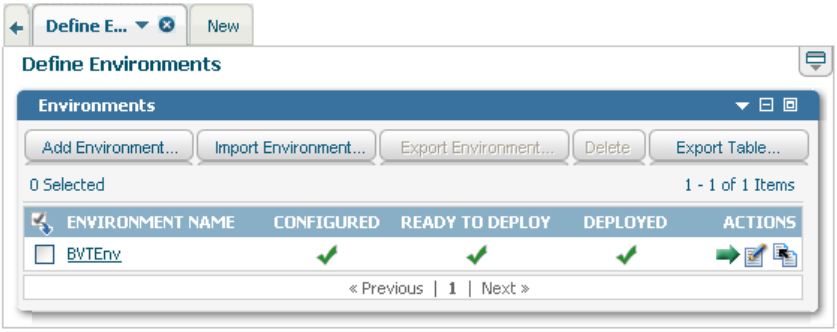
Using the Define Environments page in My webMethods

The My webMethods**Define 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.



The following table provides information about the environment configuration and deployment statuses and the icons that indicate them.

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 "Getting Started" on page 11.

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 22
Configure Servers	Allows you to specify configuration settings for all logical servers in the environment.	"Configuring Logical Servers" on page 24
Define Hosts	Allows you to define hosts (or physical servers) in the environment.	"Defining Host Servers for an Environment" on page 92
Map Servers	Allows you to map each logical server in the environment to one or more physical servers.	"Mapping Logical Servers" on page 93
Map Endpoints	Allows you to map incoming and outgoing connections for each logical server.	"Mapping Endpoints" on page 94
Map DB Pools	Allows you to associate database (DB) pools to the logical servers within the environment.	"Mapping Database Pools" on page 95
Validate	Allows you to validate the logical server settings and connections in the environment.	"Validating an Environment Configuration" on page 96

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 Software AG Products*.

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

   For more information, see *Administering webMethods Optimize*.

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

   For more information, see "Database Pool Configuration" on page 14.

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 more information, see "Managing webMethods Optimize Environments" on page 21.

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 more information, see "Managing webMethods Optimize Environments" on page 21.

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 21 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 more information about setting up SSL with My webMethods BAM Configuration, see "Security" on page 135.

Database Pool Configuration

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 Configurator.

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 Database 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 that are 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 that are 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
`MySQL`	MySQL Enterprise Edition
`MySQL Community Edition`	MySQL Community Edition If you want to use a MySQL Community Edition database, you must install a JDBC Driver for MySQL Community Edition 5.7. For more information about installing a JDBC Driver for MySQL Community Edition 5.7, see "Installing a JDBC Driver for MySQL" on page 20.
`PostgreSQL`	PostgreSQL

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 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:

Field	Entry

	`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"`
	
	MySQL sample format:
	`jdbc:wm:mysql://server-name-or-IPaddress:port`
	`databaseName=database-name`
	`[;connectOption=value ...]`
	
	MySQL Community Edition sample format:
	`jdbc:mysql://server-name-or-IPaddress:port/database-name`
	`[;connectOption=value ...]`
	
	For MySQL Community Edition, you must add the following required connection parameters:
	`relaxAutoCommit=true`
	
	The *relaxAutoCommit* parameter prevents exceptions in the `Connection.commit()` and `rollback()` methods when the connection object is in autoCommit mode. If the *relaxAutoCommit* parameter is not specified, Analytic Engine does not start.
	`useSSL=false`
	
	By default, SSL connection is enabled. You must either provide truststore for server certificate verification, or set the *useSSL* parameter to false to disable the SSL connection.
	`useLegacyDatetimeCode=false`
	
	The *useLegacyDatetimeCode* parameter forces the driver to consistently convert the time zones of the server and the client.

Field	Entry

	For JDBC Driver for MySQL versions later than 5.1, the *useLegacyDatetimeCode* parameter is set to *false* by default.
	`serverTimezone=GMT`
	To ensure that the server is running in a time zone that does not use daylight saving time, set the value of the *serverTimezone* parameter to *GMT*.
	Following is the sample format with the required parameters:
	`jdbc:mysql://server-name-or-IPaddress:port/`
	`database-name?relaxAutoCommit=true&useSSL=false`
	`&useLegacyDatetimeCode=false&serverTimezone=GMT`
	PostgreSQL sample format:
	`jdbc:wm:postgresql://server-name-or-IPaddress:port;`
	`DatabaseName=database-name`
	`[;connectOption=value ...]`

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

After you complete all of the fields in the **Database Connection** panel, you might use the **Test** button to test the database connection .

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. The default value for minimum connections is 8.

Field	Entry
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. 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.

Field	Entry
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.

   Note:
   You cannot change the **Name** of an existing database pool. When you edit the **Name** of an existing database pool and save the changes, Optimize does not update the existing database pool, but creates a new one.

   To associate logical servers to connection pools, use the **Map DB Pools** tab in webMethods Central Configurator. For more information, see "Managing webMethods Optimize Environments" on page 21.

Installing a JDBC Driver for MySQL

If you want to configure your Optimize environment to use a MySQL Community Edition database, you must install a JDBC Driver for MySQL that is compatible with MySQL Community Edition 5.7.

To install a JDBC Driver for MySQL

1. Download the JDBC Driver for MySQL.

   Note:
   Optimize supports only MySQL Community Edition 5.7.

2. Copy the driver to the *Software AG_directory* \common\lib\etx directory.

Managing webMethods Optimize Environments

This section 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. For more information, see "Exporting an Environment Configuration" on page 100 and "Importing an Environment Configuration" on page 101.

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 Software AG Universal Messaging or Broker Server (deprecated)
  - 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
- Software AG Universal Messaging or Broker Server (deprecated)
- Web Service Data Collector
- My webMethods Server

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

- Analytic Engine
- Software AG Universal Messaging or Broker Server (deprecated)
- 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 (deprecated) 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 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 provides links to the topics where you can find more information about the subcomponent settings.

To define	See
Default settings	“About Global Configuration Settings” on page 26
Analytic Engine settings	“Defining Analytic Engine Settings” on page 39
Adabas SOA Gateway Resource Module settings	“Defining Adabas SOA Gateway Resource Module Settings” on page 69
Broker Resource Module settings	“Defining Broker Resource Module Settings” on page 71
Analytic Engine Cluster configuration	“Defining Analytic Engine Cluster Settings” on page 34
Collector settings (for Infrastructure Data Collector)	“Defining Collector Settings” on page 72
Com-plete Resource Module settings	“Defining Com-plete Resource Module Settings” on page 75
Database settings	“Defining Database Settings” on page 40
Data maintenance settings	“Defining Data Maintenance Settings” on page 41
E-mail settings	“Defining E-Mail Settings” on page 51
EntireX Resource Module settings	“Defining EntireX Resource Module Settings” on page 77
ETS Resource Module settings	“Defining ETS Resource Module Settings” on page 78
Infrastructure Data Collector settings	“Defining Logical Server Subcomponents for the Infrastructure Data Collector ” on page 68
IS Resource Module settings	“Defining Integration Server Resource Module Settings” on page 81
JMSEventAction settings	“Defining JMSEventAction Settings” on page 45
JNDI configuration	“Defining JNDI Configuration Settings” on page 46
Journal logging	“Defining Journal-Logging Settings” on page 28
Monitor behavior settings	“Defining Monitor Behavior Settings” on page 53
Process Tracker settings	“Defining Process Tracker Settings” on page 54
SNMP alert settings	“Defining SNMP Alert Settings” on page 56
Station configuration	“Defining Station Configuration Settings” on page 58
Web service data collector settings	“Defining Logical Server Subcomponents for the Web Service Data Collector” on page 62
Web service action settings	“Defining WSAction Settings” on page 59

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 "About Global Configuration Settings" on page 26.

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. For more information, see "Defining Host Servers for an Environment" on page 92.

About 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.

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

Important:
webMethods Broker has been deprecated.

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.

The following table describes the JNDI Configuration settings.

Field	Description
Broker Name	If you choose “Software AG Universal Messaging” (the default value) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select "Broker (deprecated)" 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 (deprecated)” or “Universal Messaging”. If you choose “Universal Messaging”, then the **Broker Name** field should be empty. If you choose “Broker (deprecated)”, 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.
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.

Field	Description
	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 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*. The **Max File Size** field requires a specific format of the input that must include both the number and the unit of measure (KB, MB, or GB). For example, to set the **Max File Size** to *10 MB*, type `10MB`.

- **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 options for the **Period** field that are described in the following table.

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*

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 that are 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 single Terracotta Server or 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 website and product documentation. For information about installing Terracotta in a webMethods environment, refer to *Installing Software AG 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 34. 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.

Adding a Terracotta License

The Terracotta license is a file called terracotta-license.key. It contains the license information for all of your Terracotta components. You add this file to Analytic Engine by placing it into the *Software AG_directory* \common\conf directory of the machine where Analytic Engine runs.

For more information about installing the Terracotta license file on a webMethods product, see *Using Terracotta with webMethods Products*.

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. For more information, see “Adding Logical Servers to an Environment” on page 22.

2. Review Analytic Engine subcomponent 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. For more information, see “Defining Analytic Engine Cluster Settings” on page 34.

3. Define the appropriate hardware hosts for your Analytic Engine cluster as appropriate on the Define Hosts tab. For more information, see “Defining Host Servers for an Environment” on page 92.

4. Map your Analytic Engine logical servers to the desired host server machines as appropriate on the Map Servers tab. For more information, see “Mapping Logical Servers” on page 93.

5. Map endpoints as appropriate for your system configuration. For more information, see “Mapping Endpoints” on page 94. 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. For more information, see “Mapping Database Pools” on page 95.

7. Validate your configuration as described in “Validating an Environment Configuration” on page 96.

8. Deploy your configuration to the host systems as described in “Deploying an Environment” on page 97.

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 the data is 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, see the Terracotta website 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.

You must consider the following areas when you customize your Terracotta installation in an Analytic Engine clustered environment, based on system configuration considerations and desired performance characteristics:

- JVM arguments in start up script
- Persistence mode setting
- Mandatory property updates
- Optional ulimit settings to resolve log errors
- Failover tuning for guaranteed consistency

For detailed information about customizing Terracotta, see the Terracotta website and associated documentation.

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 <tc-properties></tc-properties> section of the Terracotta tc-config.xml file in order for a Analytic Engine clustering to work correctly with a TSA.

```
<property name=”search.query.wait.for.txns” value=”false”/>
```

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.

Configure the Failover Tuning for Guaranteed Consistency

The failover tuning for guaranteed consistency in Terracotta is used for clustered environments. You can set mirror groups when you have clusters that consist of up to two data centers. To configure the failover tuning for guaranteed consistency, you must set the `failover-priority` in Terracotta to `CONSISTENCY`. For more information, see the Terracotta website and associated documentation.

Cleaning Up Terracotta Persistent Caches

In a clustered environment, if Analytic Engine is configured to use Terracotta persistent caches, and you accidentally modify the Analytic Engine database while the Analytic Engine is shut down, then the caches become inconsistent with the database. As a result, Analytic Engine fails during startup.

To enable the Analytic Engine to start up properly, you must use the flushCaches tool to clear any inconsistent Terracotta persistent caches. The tool is available in the *Software AG_directory* \optimize\analysis\bin directory.

Important:
Before you run the flushCaches tool, verify that all Analytic Engine nodes are shut down.

To clean up Terracotta persistent caches

1. In your Software AG installation folder, navigate to *Software AG_directory*\optimize\analysis\bin and open a command prompt.
2. Run flushCaches.bat/sh.
3. To clear the required caches, perform one or more of the followig operations by entering the corresponding command in the console:
   - To list all known caches in a file, enter the following command:

     On a Windows environment:

     ```
     flushCaches -dump [fileName]
     ```

     On a Unix environment:

     ```
     ./flushCaches.sh -dump [fileName]
     ```

     where you must provide a new file name using the `fileName` parameter.

- To clear caches that are listed in the specified file, enter the following command:

  On a Windows environment:

  ```
  flushCaches -file [fileName]
  ```

  On a Unix environment:

  ```
  ./flushCaches.sh -file [fileName]
  ```

  where you must provide a file that contains caches to be cleared using the `fileName` parameter. Before you use this command, you must verify that all required caches are listed in the file. You can edit the file by adding and/or removing cache names.

- To clear all clustered caches, specified in the sag.opt.clusterable.caches.xml file available in your *Software AG_directory* \optimize\analysis\conf\Caching directory, enter the following command:

  On a Windows environment:

  ```
  flushCaches -all
  ```

  On a Unix environment:

  ```
  ./flushCaches.sh -all
  ```

  Note:
  If you must clear caches that are not clustered or listed in the sag.opt.clusterable.caches.xml file, you must add their names in a file and clear them using the `flushCaches -file [fileName]` command instead.

4. Restart the Analytic Engine.

When you restart the Analytic Engine, the cleared caches are reloaded from the Analytic Engine database, thus ensuring that the Terracotta persistent caches are in a consistent state.

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 several subcomponents for which you must review or configure settings.

- "Defining Analytic Engine Settings" on page 39
- "Defining Database Settings" on page 40
- "Defining Data Maintenance Settings" on page 41
- "Defining Event Publication Settings" on page 42
- "Defining JMSEventAction Settings" on page 45
- "Defining JNDI Configuration Settings" on page 46

Defining Analytic 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 Analytic 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 Analytic 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 **Analytic Engine Settings**.

5. In the **Analytic Engine Settings for Analytic Engine** area, define the Analytic Engine settings that you want the logical server to use.

The following table provides information about the Analytic Engine settings.

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.

Field	Description
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.
Publish Event Processing Metrics	Indicates whether or not Optimize will publish internal events for the Event Publication metrics. The check box is enabled by default.
Publish Persistence Metrics	Indicates whether or not Optimize will publish internal events for the Persistence metrics. The check box is enabled by default.

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.

The following table provides information about the data maintenance settings.

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.

Field	Description
	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 several Optimize related areas: KPI readings, KPI statistics, processes tracking, and rules. 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.

For information about configuring Optimize for subscription and publication of events in an EDA environment, see *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.

The following table provides more information about the events that are published when you select the corresponding check box.

Field	Description
Publish Events for KPI Readings	Publishes EDA events related to Optimize KPI readings of type *MeasurementResult*.
Publish Events for KPI Statistics	Publishes EDA events related to Optimize KPI statistics of type *MeasurementResultStatistics*.
Publish Events for Process Tracking	Publishes EDA events related to Optimize process tracking of the following types: ■ *ProcessStageStarted* ■ *ProcessStageCompleted* ■ *ProcessStageBreached* ■ *ProcessStepInstanceUnexpectedChange* ■ *ProcessInstanceDuration* ■ *ProcessStepInstanceDuration* ■ *ProcessStageDefinitionsChange*
Publish Events for Rules	Publishes EDA events in case of a rule violation in the Analytic Engine of type *BAMRuleStateChange*.

6. If applicable, use the respective filter options field to include or exclude KPI names, specific process events or rules events for publication.

The following table provides more information about the filter options.

Filter Options	Description
KPI Filter Options	Use the **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.
Process Filter Options	Use the **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.
Rules Filter Options	Use the **Include** and **Exclude** radio buttons and the **Filtered Rule Names** box to include or exclude specific rules events for publication. Type the desired rule names in the **Filtered Rule Names** box and then select the appropriate radio button depending on whether you want to include or exclude the specified rules.

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

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.

Defining In-Process Event Publication

If you want your Optimize system to use in-process event publication, instead of Universal Messaging, you must enable the in-process event publication for the Optimize Analytic Engine and disable event publishing through Universal Messaging for the Infrastructure Data Collector.

Enabling In-Process Event Publication in Analytic Engine

To enable in-process event publication for Analytic Engine

1. In Command Central: **Home > Instances > All.**
2. Select **Optimize Analytic Engine** from the list of instances.
3. Open the **Event Routing** page.

4. Open the **Configuration** tab.

5. In the drop-down list, select **Service Groups.**

6. Click **Default**.

7. Click **Edit**.

8. In the **Service Name** drop-down list, select **In-Process**.

9. Click **Save**.

Disabling Event Publication for Infrastructure Data Collector

To disable event publication for Infrastructure Data Collector

1. Stop the Infrastructure Data Collector if it is running.

2. Open the *Software AG_directory* `/profiles/InfraDC/configuration/com.softwareag.platform.config.propsloader/com.softwareag.eda.nerv.properties` configuration file in a text editor.

3. Delete the value of the `com.softwareag.eda.nerv.default.jms.provider` property.

4. Save the changes.

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

Important:
webMethods Broker has been deprecated.

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, uncheck 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. The rest of the settings are applicable based on your SSL needs.

The following table describes the **JNDI Configuration for Analytic Engine** settings.

Field	Description
Broker Name	If you choose “Software AG Universal Messaging” (the default value) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select "Broker (deprecated)" 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 (deprecated)" or “Universal Messaging”. If you choose “Universal Messaging”, then **Broker Name** field should be empty. If you choose “Broker (deprecated)”, 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.
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.

Field	Description
	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

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, clear 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 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*. The **Max File Size** field requires a specific format of the input that must include both the number and the unit of measure (KB, MB, or GB). For example, to set the **Max File Size** to *10 MB*, type `10MB`.

- **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 options for the **Period** field.

The following table provides information about the 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 that are 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 **Mail Settings forAnalytic Engine**.

The following table provides information about the **Mail Settings for Analytic Engine**.

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 seconds 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. 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 options in the **Default Days** list box that are described in the following table.

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.

Field Option	Description
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 **Process Tracker Settings for Analytic Engine** area, define the Process Tracker settings.

The following table provides information about the Process Tracker settings.

Field	Description
Input Events	Specifies the type of events that Process Tracker consumes. Set to: ■ **MAPI** (default) to define that Process Tracker consumes MAPI events. The Monitor API (MAPI) is used to deliver Optimize events and provide enhanced notifications for event delivery problems. For more information about the MAPI, see *Administering webMethods Optimize*. ■ **EDA** to define that Process Tracker consumes EDA events. EDA events are part of Software AG's implementation of an Event Driven Architecture system. In an EDA environment applications can emit and consume EDA events that occur within the suite at run time. For more information about EDA events, see *webMethods Event Processing Help*.
Process Instance Cache Size	Specifies the size of the in-memory structure where Process Tracker keeps track of all running process instances. The default value is `5000000` process instances. **Note:** If the specified size is reached, Process Tracker needs to load additional process instances data from the database, which might slow down its performance. Software AG recommends that you specify a higher value for a big number of long-running process instances.
Process Model Refresh Interval	Specifies the interval (in milliseconds) at which Process Tracker refreshes the process model definition from the database. The default value is `300000` milliseconds.
Staging Queue Maximum Size	Specifies the maximum size of the staging queue. The default value is `1000000` control operations. **Note:** If you expect large numbers of operations to be processed, Software AG recommends that you specify a higher value for this field.

Field	Description
Staging Queue Processing Batch Size	Specifies the number of objects to be passed for processing from the staging queue in a single transaction. The default value is `1000000` objects. **Note:** If your database is large, Software AG recommends that you specify a value smaller than the value of the **Staging Queue Maximum Size** field.
Staging Queue Settling Period	Specifies the period (in milliseconds) in which Process Tracker allows for incoming control operations to be gathered in the staging queue before they are processed. The default value is `10000` milliseconds.

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 and specify the settings.

The following table provides information about the SNMP alert settings.

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. Complete the fields in the station settings area.

The following table provides more information about the station settings.

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>
```

The following table provides information about the XML properties.

For this property	Substitute
Test Action	A unique name to identify the action.
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 that are listed in the following table.

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 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.

List of Parameters	Description
`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.

- "Defining DataCollector Settings for the Web Service Data Collector " on page 63
- "Defining JNDI Configuration Settings" on page 63
- "Defining Journal-Logging Settings" on page 65

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.

The following table provides information about the Data Collector settings.

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).
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

Important:
webMethods Broker has been deprecated.

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. Uncheck 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.

The following table provides information about the JNDI Configuration settings.

Field	Description
Broker Name	If you choose “Software AG Universal Messaging” (the default value) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select "Broker (deprecated)" 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 (deprecated)” or “Universal Messaging”. If you choose “Universal Messaging”, then **Broker Name** field should be empty. If you choose “Broker (deprecated)”, 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.

Field	Description
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 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 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*. The **Max File Size** field requires a specific format of the input that must include both the number and the unit of measure (KB, MB, or GB). For example, to set the **Max File Size** to *10 MB*, type `10MB`.

   - **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 options for the **Period** field, as described in the following table.

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 that are 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.

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 Apama Resource Module Settings

To define Apama 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 **Apama Resource Module Settings**.

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

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

The following table provides information about the Apama Resource Module settings.

Field	Description
Status Poll Interval	The amount of time that passes (in seconds) before the system starts to poll the Apama Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	The interval (in seconds) between retries of polling the Apama Resource Module.

Field	Description
	The default value is 30 seconds.
Status Poll Retry Count	The number of times you want the system to retry polling the Apama Resource Module. The default value is 3 retries.
Auto Discovery Interval	The interval at which Apama auto-discovery runs. The default value is 30 seconds.
Auto Discovery Flag	Indicates whether Apama auto-discovery is enabled. By default, auto-discovery is disabled.

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.

The following table provides information about the Broker Resource Module settings.

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 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.

The following table describes the configurable information for each component.

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.
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.

Field	Description
	■ 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.
Load Adabas Definitions	Indicates whether to load Adabas metadata.
Load Adabas SOA Gateway Definitions	Indicates whether to load Adabas SOA Gateway metadata.
Load Apama Definitions	indicates whether to load Apama 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 Digital Event Services Definitions	Indicates whether to load Digital Event Services metadata.
Load EntireX Definitions	Indicates whether to load EntireX metadata.
Load Event Routing Framework Definitions	Indicates whether to load Event Routing Framework metadata.
Load Integration Server Definitions	Indicates whether to load Integration Server metadata.
Load JMX Definitions	Indicates whether to load JMX metadata.
Load MashZone NextGenDefinitions	Indicates whether to load MashZone NextGen 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.

Field	Description
Load Natural Definitions	Indicates whether to load Natural metadata.
Load Presto Server Definitions	Indicates whether to load Presto version 9.9 metadata.
Load SNMP Definitions	Indicates whether to load SNMP metadata.
Load Terracotta Definitions	Indicates whether to load Terracotta metadata.
Load Universal Messaging Definitions	Indicates whether to load Universal Messaging metadata.
Load Universal Messaging Cluster Definitions	Indicates whether to load Universal Messaging Cluster metadata.

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

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 following table provides information about the possible values for the **Name** property.

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 Digital Event Services Resource Module Settings

To define Digital Event Services 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 with which you want to work.

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

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

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

5. Define the Digital Event Services Resource Module settings that you want the logical server to use.

The following table provides more information about the Digital Event Services Resource Module settings.

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

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 following table provides information about the possible values of the **Name** property.

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 following table provides information about the possible values for the **Hostname** property.

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 following table provides information about the possible values for the **Trace Level** property.

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.

Value	Description
`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.
`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 Event Routing Resource Module Settings

To define Event Routing 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 with which you want to work.

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

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

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

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

The following table provides more information about the Event Routing Resource Module settings.

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

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 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 following table provides information about the Integration Server (IS) Resource Module settings.

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. 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 JNDI Configuration Settings

Important:
webMethods Broker has been deprecated.

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 clearing 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. Clear 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.

The following table describes the JNDI Configuration settings.

Field	Description
Broker Name	If you choose “Software AG Universal Messaging” (the default value) as the JMS server in the **Naming Factory Type** field, this field should be empty. If you select "Broker (deprecated)" 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 (deprecated)” or “Universal Messaging”. If you choose “Universal Messaging”, then **Broker Name** field should be empty. If you choose “Broker (deprecated)”, 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.

Field	Description
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.

Note:
Refer to "Defining JNDI Configuration Settings" on page 63 in the section on Configuring Default Settings for Logical Servers for more information about specific JNDI Configuration settings.

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 MashZone NextGen Resource Module Settings

To define MashZone NextGen 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 **MashZone NextGen Resource Module Settings**.

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

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

The following table provides information about the MashZone NextGen Resource Module settings.

Field	Description
Status Poll Interval	This is the amount of time that passes (in seconds) before the system starts to poll the MashZone NextGen Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	This is the interval (in seconds) between retries of polling the MashZone NextGenResource Module. The default value is 30 seconds.
Status Poll Retry Count	The number of times you want the system to retry polling the MashZone NextGen 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 MashZone NextGen auto-discovery runs. The default value is 30 minutes.
Auto Discovery Flag	Indicates whether MashZone NextGen 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 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 following table provides information about the MWS Resource Module settings.

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.

Field	Description
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 Presto Resource Module Settings

To define Presto 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 **Presto Resource Module Settings**.

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

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

The following table provides information about the Presto Resource Module settings.

Field	Description
Status Poll Interval	This is the amount of time that passes (in seconds) before the system starts to poll the Presto Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	This is the interval (in seconds) between retries of polling the Presto Resource Module. The default value is 30 seconds.

Field	Description
Status Poll Retry Count	The number of times you want the system to retry polling the Presto 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 Presto auto-discovery runs. The default value is 6 minutes.
Auto Discovery Flag	Indicates whether Presto 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 Terracotta Resource Module Settings

To define Terracotta 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 **TC Resource Module Settings**.

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

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

The following table provides more information about the TC Resource Module settings.

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

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 Cluster Resource Module Settings

To define Universal Messaging Cluster 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 Cluster Resource Module Settings**.

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

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

The following table provides information about the UM Cluster Resource Module settings.

Field	Description
Status Poll Interval	This is the amount of time that passes (in seconds) before the system starts to poll the UM Cluster Resource Module. The default value is 30 seconds.
Status Poll Retry Interval	This is the interval (in seconds) between retries of polling the UM Cluster 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 Cluster Resource Module. The default value is 3 retries.
Max Wait Time For Entire Cluster Namespace	The maximum time you want the system to wait for the entire cluster namespace within the UM Cluster Resource Module. The default value is 30 seconds.
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 Cluster auto-discovery runs. The default value is 6 minutes.
Auto Discovery Flag	Indicates whether UM Cluster 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 following table provides information about the UM Resource Module settings.

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.

Field	Description
	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.

Configuring Broker Server to Work with Analytic Engine

Important:
webMethods Broker has been deprecated.

You must configure Broker Server to work with Analytic Engine by setting the `max-recv-events` and `max-send-events-size` parameters in the Broker Server configuration file, `awbroker.cfg`. For more information about the `awbroker.cfg` file and the configurable properties, see *Administering webMethods Broker*.

To configure Broker Server to work with Analytic Engine

1. Stop Broker Server if it is running.

2. Open the Broker Server `awbroker.cfg` configuration file in a text editor.

   The Broker Server configuration file is located in the *webMethods Broker_directory* `\data` directory on the machine where Broker Server is installed.

3. Add the following parameters:

```
max-recv-events=1050
max-send-events-size=1048576
```

4. Save and close the file.

5. Start Broker Server.

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 **Define Environments** 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.

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 and the hosts to which each logical server is mapped for deployment. If a logical server is not mapped, a red circle icon (O) 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 **Select Host For:** dialog box displays a list of hosts that are available for mapping to the logical server.

   Alternatively, if there is only one host available for mapping, you can click the **Map All** button to map all logical servers to that host.

5. Select a host from the **Select Host For:** list.

6. Click **Ok**.

7. Repeat steps 4-6 until each logical server is mapped to a 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**.

2. In the list of environments, click the **Edit** icon ✎ of the environment that you want to define or edit.

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

   The **Map Endpoints** page lists the logical servers in the environment, the subcomponents of each logical server, and the default values 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 (deprecated)" 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 wmjmsnaming. If you choose "Software AG 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 protocol and ports in the Incoming Connections columns.

   If you want to configure the My webMethods Server URL with root context, type the root context path in the field next to the protocol and port fields for MWS. For example, if My webMethods Server is running on *localhost* and you want to set the My webMethods Server URL to *http://localhost:8585/mymws*, type *http*, *8585*, and *mymws* in the fields.

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

5. If needed, edit the protocol and ports in the Outgoing Connections columns.

   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 135.

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 14, 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, name the four database connection pools for Optimize: `Analysis`, `MwS` (My webMethods Server), `ProcessAudit`, and `ProcessTracker`.

The following table demonstrates how to assign the database components to the four database pools.

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 Configurator 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 Configurator.

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 Configurator 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 98.

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 40.
- The analysis database pool is configured with a DB user that has the required permissions to execute DDL statements. For more information, see "Mapping Database Pools" on page 95.

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 98.

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 Configurator 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 135.

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 Configurator to use SSL and the webMethods Password Administratorutility, see "Security" on page 135.

Exporting and Importing an Environment Configuration

webMethods Central Configurator 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**.

   Your browser initiates a download.

3. Save the exported environment file.

   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 Configurator imports the environment and displays the environment name in the list of environments on the **Define Environments** page.

Using Command Central to Manage Optimize

Using Command Central to Manage Analytic Engine

You can use the Command Central web user interface and command line interface to configure and manage Analytic Engine.

You can use Command Central to perform the following operations:

- Configure settings for an Analytic Engine instance. You must configure the Analytic Engine instance to change its status from NOT_READY to ONLINE.

  Important:
  Before you start the Analytic Engine instance, you must configure:
  - At least one JDBC connection pool in the **Databases** configuration type.
  - All endpoints in the **Endpoints** configuration type.
  - All database pool mappings in the **Functional Aliases** configuration type.
- Start and stop an Analytic Engine instance.

- Monitor run-time statuses for an Analytic Engine instance.

  For information about monitoring run-time statuses, see "Lifecycle Actions for Optimize Analytic Engine" on page 132.

- Configure settings for the Event Routing component used by Analytic Engine. Event Routing is installed with pre-configured default values for a default service group. You can edit the fields to specify values other than the default ones. For more information about the fields and values to specify when configuring Event Routing settings for an Analytic Engine instance, see *Communicating Between Software AG Products Using Event Routing*.

  To make the Optimize Event Routing component visible in Command Central, first you must deploy an Optimize environment using the My webMethods Server user interface and then click  to refresh the inventory in Command Central.

- View and download Analytic Engine logs.

Configuring Optimize Analytic Engine

To configure an Analytic Engine instance

1. In Command Central, navigate to **Home > Instances > All**.
2. Click the name of the Analytic Engine instance that you want to configure.
3. Click the **Configuration** tab.
4. Select a configuration type from the drop-down list and perform the appropriate configuration task, as described in the following table.

To	Click
Add new data	
Edit data	Click the name of the component that you want to update and click **Edit**. **Important:** You must restart the Optimize Analytic Engine instance to apply the changes.
Delete data	
Test whether data is added or edited successfully	**Test**

Note:
Some tasks might not be available for the selected configuration type.

Configuring a Database Pool

To configure a database pool

1. Navigate to the **Databases** configuration type and click  .

2. In **Connection Basics**, complete the fields that are described in the following table.

Field	Description
Alias	Name of the connection pool. Use only characters that are valid for a file name in your operating system.
Description	Description of the connection pool. You can describe the purpose of the database connection pool and the location of the database schema.

3. In **Connection Details**, complete the fields that are described in the following table.

Option	Description

RDBMS	Database type.
	`SQL Server`
	`Oracle`
	`DB2` (for Linux, UNIX, and Windows)
	`MySQL`
	`MySQL Community Edition`
	`PostgreSQL`

URL	URL to the database schema that contains one or more database component sets.
Username	Name of the database user.
Password	Password of the database user.

4. Optional. Change the default **Connection Pool Configuration** settings.

   The settings under **Connection Pool Configuration** are identical to the settings in the **Pool Settings** panel in the **Database Pool Configuration** page in My webMethods.

For more information about **Pool Settings**, see "Configuring a Database Pool" on page 15.

5. Click **Save**.

   The connection pool is listed in the **Databases** table. The JDBC URL is displayed in the middle column of table.

Configuring Endpoints

To configure the Endpoint connections

1. Navigate to the **Endpoints** configuration type.

   The Endpoints table lists the four endpoints that you must configure in order to start the Analytic Engine instance.

2. Click an endpoint and click **Edit**.

3. Configure the endpoint connection settings.

The following table provides information about the endpoint connection settings.

Field	Description
Alias	Default name of the endpoint. You cannot edit this field.
Description	Description of the endpoint connection.
Protocol	Protocol type that the endpoint connection uses. The protocol types that are available in the drop-down list depend on the selected endpoint.

Protocol	Available in Endpoint
wmjmsnaming	JMS-Provider. Select this protocol if Analytic Engine uses Broker (deprecated) as a JMS provider.
nsp	JMS-Provider, if Analytic Engine uses Universal Messaging as a JMS provider. This is the default value of the protocol type for this endpoint.
nsps	JMS-Provider, if Analytic Engine uses Universal Messaging as a JMS provider.
nhp	JMS-Provider, if Analytic Engine uses Universal Messaging as a JMS provider.

Field	Description	
	nhps	JMS-Provider, if Analytic Engine uses Universal Messaging as a JMS provider.
	http	■ Configuration-Agent ■ MWS ■ WS-Registry
	https	■ Configuration-Agent ■ MWS ■ WS-Registry
Host	The host of the endpoint to which the Analytic Engine instance connects, for example *localhost*.	
Port	The port of the endpoint to which the Analytic Engine instance connects.	
Root context	The root context path of the My webMethods Server URL. For example, if you want to set the My webMethods Server URL to *protocol://host:port/mymws*, type *mymws*.	

4. Click **Save**.

   The endpoint connection settings are saved and listed in the Endpoints table.

   Important:
   Command Central does not validate the endpoint settings that you define.

Configuring Functional Aliases

To map Analytic Engine components to database pools

1. Navigate to the **Functional Aliases** configuration type and click the component name.
2. Click **Edit**.
3. Select the database pool that you want to map to the component.

   The **Pool** drop-down list contains all JDBC connection pools that you defined in the **Databases** configuration type.

4. Click **Save**.

Configuring JNDI Settings

Important:
webMethods Broker has been deprecated.

To configure JNDI settings

1. Navigate to the **JNDI** configuration type.

   The JNDI configuration settings are divided into General settings and Security settings. Only the General settings are required for the JNDI configuration.

2. Click **Edit**.

3. Configure the General settings.

The following table provides information about the General settings.

Field	Description
Naming Factory Type	Select **Universal Messaging** or **Broker (deprecated)**. The default value is **Universal Messaging**.
Broker Name	When **Naming Factory Type** is **Universal Messaging**, you cannot edit this field.
JMS Cluster URL	Type the URL if the Analytic Engine uses a Universal Messaging cluster as a JMS provider.

4. If necessary, configure the Security settings.

   a. Select the **Enable SSL** check-box.

   b. Populate the Security settings.

   For information about the Security settings that you can configure, see "Defining JNDI Configuration Settings" on page 46.

5. Click **Save**.

Configuring Analytic Engine Cluster Settings

To configure the Terracotta Server Array settings

1. Navigate to the **Clustering** configuration type.

2. Click **Edit**.

3. Enter the URL of the Terracotta server in the empty URL row. The format for this URL is `<host>:<port>`.

   Important:
   Add only one URL of the specified format in the row.

4. To add another Terracotta server and form a Terracotta Server Array, click  .

   A new row is added under the row that contains the URL of the first Terracotta server.

5. Enter the URL in the new row.

6. Click **Save**.

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

   For more information about Analytic Engine clustering, see "Analytic Engine Clustering" on page 31.

Configuring Properties

To configure settings for the logical server subcomponents of the Analytic Engine

1. Navigate to the **Properties** configuration type.

2. Click the settings link of the subcomponent, and click **Edit**.

3. Set the appropriate values in the fields.

The following table provides links to the topics where you can find more information about the subcomponent settings.

Subcomponent Settings	For more information about the subcomponent settings that you can configure, see
Analysis Engine Settings	"Defining Analytic Engine Settings" on page 39
Data Maintenance Settings	"Defining Data Maintenance Settings" on page 41
Database Settings	"Defining Database Settings" on page 40
Event Publication Settings	"Defining Event Publication Settings" on page 42
JMSEventAction Settings	"Defining JMSEventAction Settings" on page 45
Monitor Behavior Settings	"Defining Monitor Behavior Settings" on page 53

Subcomponent Settings	For more information about the subcomponent settings that you can configure, see
Process Tracker Settings	"Defining Process Tracker Settings" on page 54
SNMPAlert Settings	"Defining SNMP Alert Settings" on page 56
Station Settings	"Defining Station Configuration Settings" on page 58
WSAction Settings	"Defining WSAction Settings" on page 59

4. Click **Save**.

Configuring Email Settings

To configure email settings for the Analytic Engine

1. Navigate to the **Email** configuration type.
2. Click **Mail Settings** and click **Edit**.
3. Set the appropriate values in the fields.

   For more information about the fields that you can configure, see "Defining E-Mail Settings" on page 51.

   The field names in Command Central and the corresponding field names in My webMethods are listed in the following table.

Field Name in Command Central	Field Name in My webMethods
Mail server URL	Mail Server
Sender address	Default Sender
Receiver address	Admin Address
Mail encoding	Default Mail Encoding
Username	Server User
Password	Server Password

4. Click **Save**.

Configuring Logging

To configure Journal Logging settings for the Analytic Engine instance

1. Navigate to the **Logging** configuration type and click **Edit**.

2. Define the loggers in the provided XML snippet.

The following table describes the appender parameters.

Parameter	Description
`threshold`	Threshold level for the target. Threshold levels indicate the level of information that is logged to the target. If you do not specify the threshold level of a journal logger, the logger inherits the assigned level from the root logger. The threshold levels listed in order of increasing importance are: ■ *None*, or off ■ *Trace* ■ *Debug* ■ *Info* ■ *Warn* ■ *Error* ■ *Fatal* The journal logger logs messages that have the threshold level that you specify, as well as messages that have the threshold level that is of higher importance. For example, if the threshold value is Warn, the logger logs Warn, Error, and Fatal messages.
`target`	The standard output for the logs. ■ *System.Err* ■ *System.Out* The parameter is available for the Console appender.
`file`	Name of the file to which you want to log the data. The parameter is available for the TimeOrSize appender.
`datePattern`	Date/time characters that are appended to each log file name. ■ *yyyy-mm-dd*. Create a new log file at midnight. ■ *yyyy-mm-dd-hh:mm*. Create a new log file every minute. ■ *yyyy-mm-dd-hh*. Create a new file log every hour. *hh* uses a 24-hour clock. ■ *yyyy-mm-dd-{AM|PM}*. Create log files at noon and midnight.

Parameter	Description
	■ *yyyy-ww*. Create a new log file each Sunday at midnight. ■ *yyyy-mm*. Create a new log on the last day of each month at midnight. The parameter is available for the TimeOrSize appender.
`syncOnQueueFull`	■ *true*. Synchronize logs when the maxFileSize is reached. ■ *false*. Do not synchronize logs when the maxFileSize is reached. The parameter is available for the Async appender.
`queueSize`	The maximum queue size. The parameter is available for the Async appender.
`maxLogLimit`	Number of log files that are kept in the rollover destination. When the maximum number of log files is reached, the oldest log file is removed and a new one is allocated. The parameter is available for the TimeOrSize appender.
`maxFileSize`	Restrict the amount of disk space (in megabytes) that is used for logging. The parameter is available for the TimeOrSize appender.
`type`	Destination where you want to store the rolled over log files. The possible values are: ■ *File* ■ *Directory* The parameter is available for the TimeOrSize appender.
`hostName`	The default value is *localhost*. The parameter is available for the TimeOrSize appender when the value of the type parameter is Directory.
`logDirectory`	Name of the directory where the rolled over files are stored. Type `.` if you want the logs to be stored in the default directory. The parameter is available for the TimeOrSize appender when the value of the type parameter is Directory.
`dateFormatString`	Date format that is used for the rolled over log files. The parameter is available for the TimeOrSize appender when the value of the type parameter is Directory.
`timeFormatString`	Time format that is used for the rolled over log files. The parameter is available for the TimeOrSize appender when the value of the type parameter is Directory.

Note:
You can edit the default appender parameters, or add appenders to the xml.

3. Click **Save**.

Securing Optimize Analytic Engine with SSL

To secure Optimize Analytic Engine with SSL (Secure Sockets Layer) you must set up the appropriate SSL KeyStore and TrustStore in the GlueSSLProperties.xml file.

For more information about the procedures you must follow to configure SSL for Analytic Engine, see "Security" on page 135 and "Securing Optimize Engines with SSL" in *Administering webMethods Optimize*.

Using Command Central to Manage Web Service Data Collector

You use the Command Central web user interface and command line interface to manage the Web Service Data Collector.

You can use Command Central to manage the following operations for the Web Service Data Collector:

- Configure settings for a Web Service Data Collector instance. You must configure the Web Service Data Collector instance to change its status from NOT_READY to ONLINE.
- Start, stop, and restart a Web Service Data Collector instance.

  For detailed information about changing the status of a product instance, see *Software AG Command Central Help*.
- Monitor run-time statuses for a Web Service Data Collector instance.

  For information about monitoring run-time statuses, see "Lifecycle Actions for Optimize Web Service Data Collector" on page 133.
- View and download Web Service Data Collector logs.

Configuring Optimize Web Service Data Collector

To configure a Web Service Data Collector instance

1. In Command Central, navigate to **Home > Instances > All**.
2. Click the Optimize Web Service Data Collector instance that you want to configure.
3. Click the **Configuration** tab.
4. Select a configuration type from the drop-down list and perform the appropriate configuration task, as described in the following table.

To	Click
Add new data	
Edit data	Click the name of the component that you want to update and click **Edit**. **Important:** You must restart the Optimize Web Service Data Collector instance to apply the changes.
Delete data	
Test whether data is added or edited successfully	**Test**

Note:
Some tasks might not be available for the selected configuration type.

Configuring Endpoints

To configure the Endpoint connections

1. Navigate to the **Endpoints** configuration type.

   The Endpoints table lists the three endpoints that you must configure in order to start the Web Service Data Collector instance.

2. Click an endpoint and click **Edit**.

3. Configure the endpoint connection settings.

The following table provides information about the endpoint connection settings.

Field	Description
Alias	Default name of the endpoint. You cannot edit this field.
Description	Description of the endpoint connection.
Protocol	Protocol type that the endpoint connection uses. The protocol types that are available in the drop-down list depend on the selected endpoint.

Protocol	Available in Endpoint
wmjmsnaming	JMS-Provider.

Field	Description	
		Select this protocol if Web Service Data Collector uses Broker (deprecated) as a JMS provider.
	nsp	JMS-Provider, if Web Service Data Collector uses Universal Messaging as a JMS provider. This is the default value of the protocol type for this endpoint.
	nsps	JMS-Provider, if Web Service Data Collector uses Universal Messaging as a JMS provider.
	nhp	JMS-Provider, if Web Service Data Collector uses Universal Messaging as a JMS provider.
	nhps	JMS-Provider, if Web Service Data Collector uses Universal Messaging as a JMS provider.
	http	■ Configuration-Agent ■ MWS ■ WS-Registry
	https	■ Configuration-Agent ■ MWS ■ WS-Registry
Host	The host of the endpoint to which the Web Service Data Collector instance connects, for example *localhost*.	
Port	The port of the endpoint to which the Web Service Data Collector instance connects.	
Root context	The root context path of the My webMethods Server URL. For example, if you want to set the My webMethods Server URL to *protocol://host:port/mymws*, type *mymws*.	

4. Click **Save**.

The endpoint connection settings are saved and listed in the Endpoints table.

Important:
Command Central does not validate the endpoint settings that you define.

Configuring JNDI Settings

Important:

webMethods Broker has been deprecated.

To configure JNDI settings

1. Navigate to the **JNDI** configuration type.

   The JNDI configuration settings are divided into General settings and Security settings. Only the General settings are required for the JNDI configuration.

2. Click **Edit**.

3. Configure the General settings.

The following table provides information about the General settings.

Field	Description
Naming Factory Type	Select **Universal Messaging** or **Broker (deprecated)**. The default value is **Universal Messaging**.
Broker Name	When **Naming Factory Type** is **Universal Messaging**, you cannot edit this field.
JMS Cluster URL	Type the URL if the Web Service Data Collector uses a Universal Messaging cluster as a JMS provider.

4. If necessary, configure the Security settings.

   a. Select the **Enable SSL** check-box.

   b. Populate the Security settings.

   For information about the Security settings that you can configure, see "Defining JNDI Configuration Settings" on page 46.

5. Click **Apply**.

Configuring Properties

To configure data collector settings for Web Service Data Collector

1. Navigate to the **Properties** configuration type.

2. Click**Data Collector Settings**, and click **Edit**.

3. Set the appropriate values in the fields.

For more information about the settings that you can configure, see "Defining DataCollector Settings for the Web Service Data Collector " on page 63.

4. Click **Save**.

Configuring Logging

To configure Journal Logging settings for the Web Service Data Collector instance

1. Navigate to the **Logging** configuration type and click **Edit**.
2. Define the loggers in the provided XML snippet.

   You can edit the default appender parameters, or add new appenders to the xml. For more information about the configurable properties, see "Configuring Logging" on page 108.

3. Click **Save**.

Securing Optimize Web Service Data Collector with SSL

To secure Optimize Web Service Data Collector with SSL (Secure Sockets Layer) you must set up the appropriate SSL KeyStore and TrustStore in the GlueSSLProperties.xml file.

For more information about the procedures you must follow to configure SSL for Web Service Data Collector, see "Security" on page 135 and "Securing Optimize Engines with SSL" in *Administering webMethods Optimize*.

Using Command Central to Manage Infrastructure Data Collector

You use the Command Central web user interface and command line interface to manage the Infrastructure Data Collector.

You can use Command Central to manage the following operations for the Infrastructure Data Collector:

- Configure settings for a Infrastructure Data Collector instance. You must configure the Infrastructure Data Collector instance to change its status from NOT_READY to ONLINE.
- Change the default settings of the Infrastructure Data Collector OSGi component.
- Start, stop, and restart a Infrastructure Data Collector instance.

  For detailed information about changing the status of a product instance, see *Software AG Command Central Help*.

- Monitor run-time statuses for a Web Service Data Collector instance.

  For information about monitoring run-time statuses, see "Lifecycle Actions for Optimize Web Service Data Collector" on page 133.

- View and download Infrastructure Data Collector logs.

Configuring Optimize Infrastructure Data Collector

To configure an Infrastructure Data Collector instance

1. In Command Central, navigate to **Home > Instances > All**.
2. Click the name of the Infrastructure Data Collector instance that you want to configure.
3. Click the **Infrastructure DC** tab.
4. Click the **Configuration** tab.
5. Select a configuration type from the drop-down list and perform the appropriate configuration task, as described in the following table.

To	Click
Add new data	
Edit data	Click the name of the component that you want to update and click **Edit**. **Important:** You must restart the Optimize Infrastructure Data Collector instance to apply the changes.
Delete data	
Test whether data is added or edited successfully	**Test**

Note:
Some tasks might not be available for the selected configuration type.

Configuring Collector Settings

To configure Collector settings

1. Navigate to the **Collector Settings** configuration type.
2. Click **Edit**.

3. Configure the Collector Settings.

   For more information about the fields, see "Defining Collector Settings" on page 72.

4. Select the metadata definitions that you want the collector to load.

5. Click **Save**.

Configuring Endpoints

To configure the Endpoint connections

1. Navigate to the **Endpoints** configuration type.

   The Endpoints table lists the endpoints that you must configure to start the Infrastructure Data Collector instance.

2. Click an endpoint and click **Edit**.

3. Configure the endpoint connection settings.

The following table provides information about the endpoint connection settings.

Field	Description
Alias	Default name of the endpoint. You cannot edit this field.
Description	Description of the endpoint connection.
Protocol	Protocol type that the endpoint connection uses. ■ **wmjmsnaming** Select this protocol if Infrastructure Data Collector uses Broker (deprecated) as a JMS provider. ■ **NSP** This is the default value of the protocol type for this endpoint. ■ **NSPS** ■ **NHP** ■ **NHPS**
Host	The host of the endpoint to which the Infrastructure Data Collector instance connects, for example *localhost*.
Port	The port of the endpoint to which the Infrastructure Data Collector instance connects.

4. Click **Save**.

The endpoint connection settings are saved and listed in the Endpoints table.

Important:
Command Central does not validate the endpoint settings that you define.

Configuring JNDI Settings

Important:
webMethods Broker has been deprecated.

To configure JNDI settings

1. Navigate to the **JNDI** configuration type.

   The JNDI configuration settings are divided into General settings and Security settings. Only the General settings are required for the JNDI configuration.

2. Click **Edit**.

3. Configure the General settings.

The following table provides information about the General settings.

Field	Description
Naming Factory Type	Select **Universal Messaging** or **Broker (deprecated)**. The default value is **Universal Messaging**.
Broker Name	When **Naming Factory Type** is **Universal Messaging**, you cannot edit this field.
JMS Cluster URL	Type the URL if the Infrastructure Data Collector uses a Universal Messaging cluster as a JMS provider.

4. If necessary, configure the Security settings.

   a. Select the **Enable SSL** check-box.

   b. Populate the Security settings.

   For information about the Security settings that you can configure, see "Defining JNDI Configuration Settings" on page 82.

5. Click **Apply**.

Configuring Resource Module Settings

To configure the resource module settings of the Infrastructure Data Collector

1. Navigate to the **Resource Module Settings** configuration type.
2. Click the settings link of the resource module, and click **Edit**.
3. Set the appropriate values in the fields.

The following table provides links to the topics where you can find more information about the resource module settings.

Resource module settings	For more information about the resource module settings that you can configure, see
Adabas Resource Module Settings	"Defining Adabas SOA Gateway Resource Module Settings" on page 69.
Apama Resource Module Settings	"Defining Apama Resource Module Settings" on page 70.
Broker Resource Module Settings	"Defining Broker Resource Module Settings" on page 71.
Complete Resource Module Settings	"Defining Com-plete Resource Module Settings" on page 75.
Digital Event Services Resource Module Settings	"Defining Digital Event Services Resource Module Settings" on page 76.
ETS Resource Module Settings	"Defining ETS Resource Module Settings" on page 78.
EntireX Resource Module Settings	"Defining EntireX Resource Module Settings" on page 77.
Event Routing Resource Module Settings	"Defining Event Routing Resource Module Settings" on page 80.
IS Resource Module Settings	"Defining Integration Server Resource Module Settings" on page 81.
MWS Resource Module Settings	"Defining My webMethods Server Resource Module Settings" on page 86.
MashZone NG Resource Module Settings	"Defining MashZone NextGen Resource Module Settings" on page 84.
Presto Resource Module Settings	"Defining Presto Resource Module Settings" on page 87.
TC Resource Module Settings	"Defining Terracotta Resource Module Settings" on page 88.
UM Cluster Resource Module Settings	"Defining Universal Messaging Cluster Resource Module Settings" on page 89.

Resource module settings	For more information about the resource module settings that you can configure, see
UM Resource Module Settings	"Defining Universal Messaging (UM) Resource Module Settings" on page 91.

4. Click **Save**.

Configuring the Infrastructure Data Collector OSGi Component

To configure the Infrastructure Data Collector OSGi Component

1. In Command Central, navigate to **Home > Instances > All**.
2. Click the name of the Infrastructure Data Collector instance, for example "InfraDC".
3. Click the **Configuration** tab.
4. Select a configuration type and click **Edit**.

> **Note:**
> If you select the **Ports** configuration type, you must click the **Alias** link before you click **Edit**.

5. Configure the OSGi component settings.

The following table provides information about the OSGi component settings.

OSGi component configuration type	For more information about the OSGi component settings that you can configure, see
JVM Options	"Configuring Additional JVM Parameters " in *Administering webMethods Optimize*.
Java System Properties	"Configuring the Java Service Wrapper" in *Administering webMethods Optimize*.
Memory	"Configuring JVM Memory Allocation" in *Administering webMethods Optimize*.
Ports	"Configure Ports" on page 120.

6. Click **Apply**.

Configure Ports

Command Central uses the ports specified in the OSGi component to monitor Infrastructure Data Collector.

To configure the ports for the Infrastructure Data Collector OSGi component

1. Navigate to the **Ports** configuration type and click the **Alias** link.
2. Click **Edit**.
3. In Connection Basics, configure the fields for the corresponding port type.
   - The following table describes the configurations for HTTP and HTTPS ports.

Field	Description
Enabled	Whether the port is enabled.
Port Number	The port number that you want to use. Type a number that is not already in use.
Alias	The port name that you want to use. Use an alias that is unique for the instance or component and can be included in a user-friendly URL. Use nly ASCII characters, numbers, underscore (_), dot (.), and a hyphen (-).
Keep Alive Timeout	When to close the connection if the server has not received a request from the client within this timeout value (in milliseconds); or when to close the connection if the client has explicitly placed a close request with the server.
Spare Threads Min	The starting number of request processing spare threads.
Redirect Port	The port to use when redirecting SSL connection requests.
Spare Threads Max	The maximum number of request processing spare threads.
Accept Count	The maximum number of simultaneous connection requests allowed in the connection queue.
Connection Timeout	The connection timeout (in milliseconds). This attribute is not set by default on HTTPS ports.
HTTP Header Size Max	The maximum incoming URL length in characters.
Upload Timeout Disable	Indicates if using a longer connection timeout is allowed when waiting for the servlet container to update.
Lookups Enable	Indicates if DNS lookups are allowed to get the actual host name of a remote client.

   - The following table describes the configurations for JMX and SSH ports.

Field	Description
Enabled	Whether the port is enabled.
Port Number	The port number that you want to use. Type a number that is not already in use. ■ For JMX, select the port for monitoring, managing, and implementing the Java process. ■ For SSH, select the port for remote shell services or execution processes.
Alias	The port name that you want to use. Use an alias that is unique for the instance or component and can be included in a user-friendly URL. Use nly ASCII characters, numbers, underscore (_), dot (.), and a hyphen (-).
JAAS Realm	Specifies the realm name that authenticates the Java Authentication and Authorization (JAAS) service.

- The following table describes the configurations for JDWP port.

Field	Description
Port Number	The port number that you want to use. Type a number that is not already in use.
Alias	The port name that you want to use. Use an alias that is unique for the instance or component and can be included in a user-friendly URL. Use nly ASCII characters, numbers, underscore (_), dot (.), and a hyphen (-).
Suspend	Suspends the runtime until debugger connects.

Note:
The JDWP port is only used when the profile is started in debug mode.

4. In Thread Pool Configuration, for HTTP and HTTPS ports, complete the fields that are described in the following table.

Field	Description
Enabled	Whether the listener uses this pool exclusively for dispatching requests. The existing thread pool is a global thread pool. If there is a very high load on this resource, there may be a delay until the global thread pool can process the request. When you enable the private threat pool option, the requests that come into this port do not compete with other server functions for threads. When you view the port details, the server reports the total number of private thread pool threads currently in use for the port.

Field	Description
Threadpool Min	The minimum number of threads for this private thread pool. The default value is 1.
Threadpool Max	The maximum number of threads for this private thread pool. The default value is 5.
Threadpool Priority	The Java thread priority. The default value is 5. **Important:** Use this setting with extreme care because it affects server performance and throughput.

5. Click **Save**.

Using the Command Line to Manage Optimize

Commands that Optimize Supports

Optimize supports the Command Central Command Line Interface commands listed in the following table. The table lists where you can find general information about each command. Additionally, the table lists where you can learn more about arguments and options that Optimize supports or details about the actions Optimize takes when you execute an `exec` command.

Commands	For more information, see
`sagcc create configuration data`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration Types that Optimize Web Service Data Collector Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc delete configuration data`	For general information about the command, see *Software AG Command Central Help*.
`sagcc get configuration data`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration Types that Optimize Web Service Data Collector

Commands	For more information, see
	Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc update configuration data`	For general information about the command, see *Software AG Command Central Help*.
`sagcc get configuration instances`	For general information about the command, see *Software AG Command Central Help*.
`sagcc list configuration instances`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration Types that Optimize Web Service Data Collector Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc get configuration types`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration Types that Optimize Web Service Data Collector Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc list configuration types`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration Types that Optimize Web Service Data Collector Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc exec configuration validation create`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration

Commands	For more information, see
	Types that Optimize Web Service Data Collector Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc exec configuration validation delete`	For general information about the command, see *Software AG Command Central Help*.
`sagcc exec configuration validation update`	For general information about the command, see *Software AG Command Central Help*. For Optimize-specific information about using this command, see "Configuration Types that Optimize Analytic Engine Supports" on page 131, "Configuration Types that Optimize Web Service Data Collector Supports" on page 132, and "Configuration Types that Optimize Infrastructure Data Collector Supports" on page 132.
`sagcc exec administration product migration migrate`	This command is supported by Optimize Infrastructure Data Collector. For general information about the command, see *Software AG Command Central Help*. For information about the Infrastructure Data Collector migration utility, see *Upgrading Software AG Products*.

Examples When Executing on Command Central

Examples for Analytic Engine

- To list all configuration types for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc list configuration types optimizeAnalysis-analysis
--server http://localhost:8092/spm -p secret
```

- To list all configuration instances for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc list configuration instances optimizeAnalysis-analysis
--server http://localhost:8092/spm -p secret
```

- To test the creation of a JDBC configuration instance "COMMON-JDBC-mssql" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret", and configuration definition in "test.xml" file:

```
sagcc exec configuration validation optimizeAnalysis-analysis create
```

```
COMMON-JDBC --server http://localhost:8092/spm -p secret
-i C:\inputs\test.xml
```

- To create a JDBC configuration instance "COMMON-JDBC-mssql" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret", and configuration definition in "test.xml" file:

  ```
  sagcc create configuration data optimizeAnalysis-analysis COMMON-JDBC
  --server http://localhost:8092/spm -p secret -i C:\inputs\test.xml
  ```

  The format of the "test.xml" file that creates the "COMMON-JDBC-mssql" configuration instance is the following:

  ```
  <JDBCSettings>
              <Pool alias="mssql">
                          <Name>mssql</Name>
                          <Description>Microsoft SQL DB</Description>
                          <MinSize>8</MinSize>
                          <MaxSize>60</MaxSize>
                          <MaxIdleTime>20</MaxIdleTime>
                          <RampUpDelay>500</RampUpDelay>
                          <Retries>8</Retries>
                          <RetriesBackoff>500</RetriesBackoff>
                          <MaxStatementsInCache>0</MaxStatementsInCache>
                          <DatabaseServer type="SQLSERVER">
                                      <Host>localhost</Host>
                                      <PortNumber>1433</PortNumber>
                                      <Database>optimize</Database>
                                      <URL>jdbc:wm:sqlserver://localhost:1433;
                                      DatabaseName=optimize</URL>
                                      <User>Administrator</User>
                                      <Password>1482227624814_4</Password>
                          </DatabaseServer>
              </Pool>
  </JDBCSettings>
  ```

- To get the configuration instance details for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret" and include the output instance details in an xml file:

  ```
  sagcc get configuration instances optimizeAnalysis-analysis
  COMMON-JDBC-default --server http://localhost:8092/spm -p secret --format xml
  ```

- To test the update of the configuration data of a JDBC configuration instance "COMMON-JDBC-mssql" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret", and configuration definition in "test.xml" file:

  ```
  sagcc exec configuration validation optimizeAnalysis-analysis udpate
  COMMON-JDBC-mssql --server http://localhost:8092/spm -p secret
  -i C:\inputs\test.xml
  ```

- To update the configuration data of a JDBC configuration instance "COMMON-JDBC-mssql" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret":

  ```
  sagcc update configuration data optimizeAnalysis-analysis
  ```

```
COMMON-JDBC-mssql --server http://localhost:8092/spm -p secret
-i C:\inputs\test.xml
```

- To get the configuration data of a JDBC configuration instance "COMMON-JDBC-mssql" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc get configuration data optimizeAnalysis-analysis COMMON-JDBC-mssql
--server http://localhost:8092/spm
```

- To delete a JDBC configuration instance "COMMON-JDBC-mssql" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm":

```
sagcc delete configuration data optimizeAnalysis-analysis COMMON-JDBC
--server http://localhost:8092/spm -p secret
```

- To update the configuration data of an Endpoints configuration instance "COMMON-COMPONENT-ENDPOINTS-Configuration-Agent" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "test1.xml" file:

```
sagcc update configuration data optimizeAnalysis-analysis
COMMON-COMPONENT-ENDPOINTS-Configuration-Agent
--server http://localhost:8092/spm -p secret -i C:\inputs\test1.xml
```

- To update the configuration data of a Databases configuration instance "COMMON-DBFUNCTION-AnalysisEngine" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "test2.xml" file:

```
sagcc update configuration data optimizeAnalysis-analysis
COMMON-DBFUNCTION-AnalysisEngine
--server http://localhost:8092/spm -p secret -i C:\inputs\test2.xml
```

- To update the configuration data of a configuration instance "COMMON-SMTP-MailSettings" for Analytic Engine with ID "optimizeAnalysis-analysis" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "test3.xml" file:

```
sagcc update configuration data optimizeAnalysis-analysis
COMMON-SMTP-MailSettings --server http://localhost:8092/spm
-p secret -i C:\inputs\test3.xml
```

The format of the "test3.xml" file that updates the "COMMON-SMTP-MailSettings" configuration instance is the following:

```
<EmailSettings>
            <Host>smtp</Host>
            <PortNumber>6670</PortNumber>
            <User>admin</User>
            <Password>mailServerPasswordHandle</Password>
            <Charset>UTF-8</Charset>
            <ExtendedProperties>
               <Property name="name">Mail Settings</Property>
               <Property name="AuthenticationRequired">false</Property>
               <Property name="SenderDomain">softwareag.com</Property>
```

```
                <Property name="DefaultSender">optimize@localhost</Property>
                <Property name="AdminAddress">admin@localhost</Property>
                <Property name="Templates">
Business monitor violation=./templates/BusinessMonitorViolation.template
SNMPAlert=./templates/SNMPAlert.template
DefaultTemplate=./templates/DefaultEmailAlert.template.vm
                </Property>
                <Property name="SocketTimeout">60000</Property>
            </ExtendedProperties>
</EmailSettings>
```

Examples for Web Service Data Collector

- To get the configuration data of the endpoints configuration instance "COMMON-COMPONENT-ENDPOINTS-Configuration-Agent" for Web Service Data Collector with ID "optimizeWSDataCollector-dataCollector" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc get configuration data optimizeWSDataCollector-dataCollector
COMMON-COMPONENT-ENDPOINTS-Configuration-Agent
--server http://localhost:8092/spm -p secret
```

- To update the configuration data of the endpoints configuration instance "COMMON-COMPONENT-ENDPOINTS-Configuration-Agent" for Web Service Data Collector with ID "optimizeWSDataCollector-dataCollector" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "port_data.xml" file:

```
sagcc update configuration data optimizeWSDataCollector-dataCollector
COMMON-COMPONENT-ENDPOINTS-Configuration-Agent
--server http://localhost:8092/spm -p secret -i c:\inputs\port_data.xml
```

- To get the configuration data of a JNDI configuration instance "COMMON-JNDI-JNDIConfiguration" for Web Service Data Collector with ID "optimizeWSDataCollector-dataCollector" that is running on server "http://localhost:8092/spm":

```
sagcc get configuration data optimizeWSDataCollector-dataCollector
COMMON-JNDI-JNDIConfiguration --server http://localhost:8092/spm -p secret
```

- To update the configuration data of a JNDI configuration instance "COMMON-JNDI-JNDIConfiguration" for Web Service Data Collector that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "port_data.xml" file:

```
sagcc update configuration data optimizeWSDataCollector-dataCollector
COMMON-JNDI-JNDIConfiguration --server http://localhost:8092/spm
-p secret -i c:\inputs\port_data.xml
```

- To get the configuration data of a Logging configuration instance "COMMON-LOG" for Web Service Data Collector with ID "optimizeWSDataCollector-dataCollector" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc get configuration data optimizeWSDataCollector-dataCollector
COMMON-LOG --server http://localhost:8092/spm -p secret
```

- To update the configuration data of a Logging configuration instance "COMMON-LOG" for Web Service Data Collector with ID "optimizeWSDataCollector-dataCollector" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data fin the "port_data.xml" file:

```
sagcc update configuration data optimizeWSDataCollector-dataCollector
COMMON-LOG --server http://localhost:8092/spm -p secret
-i c:\inputs\port_data.xml
```

Examples for Infrastructure Data Collector

- To list all configuration types for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc list configuration types OSGI-InfraDC-ENGINE
--server http://localhost:8092/spm -p secret
```

- To list all configuration instances for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc list configuration instances OSGI-InfraDC-ENGINE
--server http://localhost:8092/spm -p secret
```

- To test the update of the configuration data of a JNDI configuration instance "COMMON-JNDI-InfraDC-JNDIConfiguration" for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret", and configuration definition in "jndi_test.xml" file:

```
sagcc exec configuration validation OSGI-InfraDC-ENGINE udpate
COMMON-JNDI-InfraDC-JNDIConfiguration --server http://localhost:8092/spm
-p secret -i C:\inputs\jndi_test.xml
```

- To update the configuration data of a JNDI configuration instance "COMMON-JNDI-InfraDC-JNDIConfiguration" for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret", and configuration definition in "jndi_test.xml" file:

```
sagcc update configuration data OSGI-InfraDC-ENGINE
COMMON-JNDI-InfraDC-JNDIConfiguration --server http://localhost:8092/spm
-p secret -i C:\inputs\jndi_test.xml
```

  The format of the "jndi_test.xml" file that updates the "COMMON-JNDI-InfraDC-JNDIConfiguration" configuration instance is the following:

```
<JNDISettings>
    <JNDI alias="JNDIConfiguration-InfraDC">
    <ExtendedProperties>
       <Property name="ConnectionUri"/>
       <Property name="NamingFactoryImpl">com.pcbsys.nirvana.nSpace.
NirvanaContextFactory</Property>
       <Property name="UseSSL">true</Property>
       <Property name="UseEncryption">true</Property>
       <Property name="KeyStoreFile">C:\SoftwareAG\certificates\
client.jks</Property>
```

```
       <Property name="KeyStoreType">JKS</Property>
       <Property name="DN"/>
       <Property name="TrustStoreFile">C:\SoftwareAG\certificates\
truststore.jks</Property>
       <Property name="TrustStoreType">JKS</Property>
       <Property name="StorePassword">brokerKeyAndTrustStorePasswordHandle
</Property>
       <Property name="JmsUrlOverride"/>
    </ExtendedProperties>
    </JNDI>
</JNDISettings>
```

- To get the configuration data of a JNDI configuration instance "COMMON-JNDI-InfraDC-JNDIConfiguration" for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret":

```
sagcc get configuration data
OSGI-InfraDC-ENGINE COMMON-JNDI-InfraDC-JNDIConfiguration
--server http://localhost:8092/spm
```

- To delete a JNDI configuration instance "COMMON-JNDI-InfraDC-JNDIConfiguration" for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm":

```
sagcc delete configuration data
OSGI-InfraDC-ENGINE COMMON-JNDI-InfraDC-JNDIConfiguration
--server http://localhost:8092/spm -p secret
```

- To update the configuration data of an Endpoints configuration instance "COMMON-COMPONENT-ENDPOINTS-JMS-Provider" for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "test1.xml" file:

```
sagcc update configuration data OSGI-InfraDC-ENGINE
COMMON-COMPONENT-ENDPOINTS-JMS-Provider
--server http://localhost:8092/spm -p secret -i C:\inputs\test.xml
```

- To update the configuration data of a Collector Settings configuration instance "IDC-SETTING-CollectorSettings" for Infrastructure Data Collector with ID "OSGI-InfraDC-ENGINE" that is running on server "http://localhost:8092/spm" with administrator password "secret" with the configuration data in the "cs_test.xml" file:

```
sagcc update configuration data OSGI-InfraDC-ENGINE
IDC-SETTING-CollectorSettings
--server http://localhost:8092/spm -p secret -i C:\inputs\cs_test.xml
```

  The format of the "cs_test.xml" file that updates the "IDC-SETTING-CollectorSettings" configuration instance is the following:

```
<properties>
    <name>Collector Settings</name>
    <dcName>InfraDCxx</dcName>
    <pollingInterval>2</pollingInterval>
    <adabasLoadFlag>false</adabasLoadFlag>
    <adabasSOAGatewayLoadFlag>true</adabasSOAGatewayLoadFlag>
```

```
    <apamaLoadFlag>false</apamaLoadFlag>
    <applinxLoadFlag>true</applinxLoadFlag>
    <brokerLoadFlag>false</brokerLoadFlag>
    <completeDCLoadFlag>true</completeDCLoadFlag>
    <entireXLoadFlag>false</entireXLoadFlag>
    <eventRoutingLoadFlag>false</eventRoutingLoadFlag>
    <integrationServerLoadFlag>true</integrationServerLoadFlag>
    <mwsLoadFlag>true</mwsLoadFlag>
    <mzngLoadFlag>true</mzngLoadFlag>
    <naturalAjaxLoadFlag>true</naturalAjaxLoadFlag>
    <naturalLoadFlag>true</naturalLoadFlag>
    <prestoLoadFlag>true</prestoLoadFlag>
    <snmpLoadFlag>true</snmpLoadFlag>
    <tcLoadFlag>true</tcLoadFlag>
    <umClusterLoadFlag>false</umClusterLoadFlag>
    <umLoadFlag>false</umLoadFlag>
    <desLoadFlag>true</desLoadFlag>
    <logLevel>TRACE</logLevel>
</properties>
```

- To view the command line help for the migration utility for Infrastructure Data Collector with ID "InfrastructureDC":

```
sagcc list administration product local InfrastructureDC migration help
```

- To migrate Infrastructure Data Collector with ID "InfrastructureDC" from a source directory "C:/SoftwareAG" to a destination directory "C:/SoftwareAG_" on a target node with alias "local":

```
sagcc exec administration product local InfrastructureDC migration
migrate srcDir=C:/SoftwareAG destDir=C:/SoftwareAG_
```

Configuration Types that Optimize Analytic Engine Supports

The Optimize Analytic Engine run-time component supports creating instances of the configuration types that are listed in the following table.

Configuration Type	Use to configure
COMMON-CLUSTER	Cluster configuration and to specify cluster settings such as the Terracotta Server Array URLs. **Important:** Any changes to the cluster configuration take effect only after you restart the node.
COMMON-COMPONENT-ENDPOINTS	Endpoint connections between Analytic Engine and its logical servers.
COMMON-DBFUNCTION	Database functional aliases for Analytic Engine.
COMMON-JDBC	JDBC connection pools for Analytic Engine.
COMMON-JNDI	JNDI settings for Analytic Engine.

Configuration Type	Use to configure
COMMON-LOG	Log levels for log categories and log file locations.
COMMON- SMTP	Settings for sending e-mail messages.
COMMON-SYSPROPS	Server configuration parameters.

Configuration Types that Optimize Web Service Data Collector Supports

The Optimize Web Service Data Collector run-time component supports creating instances of the configuration types that are listed in the following table.

Configuration Type	Use to configure
COMMON-COMPONENT-ENDPOINTS	Endpoint connections between Web Service Data Collector and its logical servers.
COMMON-JNDI	JNDI settings for Optimize Web Service Data Collector.
COMMON-LOG	Log levels for log categories and log file locations.
COMMON-SYSPROPS	Server configuration parameters.

Configuration Types that Optimize Infrastructure Data Collector Supports

The Optimize Infrastructure Data Collector run-time component supports creating instances of the configuration types that are listed in the following table.

Configuration Type	Use to configure
COMMON-COMPONENT-ENDPOINTS	Endpoint connections between Infrastructure Data Collector and its logical servers.
COMMON-JNDI	JNDI settings for Infrastructure Data Collector.
IDC-RESOURCES	Infrastructure Data Collector Resource Module Properties Configuration.
IDC-SETTINGS	Infrastructure Data Collector Settings.

Lifecycle Actions for Optimize Analytic Engine

The following table lists the actions that Analytic Engine supports with the `sagcc exec lifecycle` command and the operation taken against Analytic Engine when an action is executed.

Action	Description
Start	Starts an instance that was stopped or not started.

Action	Description
Stop	Stops an instance that was started earlier.
Restart	Restarts an instance that was running or stopped earlier.

Run-time Monitoring Statuses for Optimize Analytic Engine

The following table lists the run-time statuses that the Analytic Engine run-time component can return in response to the `sagcc get monitoring runtimestatus` and `sagcc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Description
STARTING	The Analytic Engine server is starting.
ONLINE	The Analytic Engine server is running.
NOT_READY	The Analytic Engine server is started, but it is not configured in the My webMethods Server administration user interface. If the Analytic Engine server is not configured, it cannot accept client requests.
STOPPING	The Analytic Engine server is stopping.
STOPPED	The Analytic Engine server has stopped.

Lifecycle Actions for Optimize Web Service Data Collector

The following table lists the actions that the Web Service Data Collector supports with the `sagcc exec lifecycle` command and the operation taken against the Web Service Data Collector when an action is executed.

Action	Description
Start	Starts an instance that was stopped or not started.
Stop	Stops an instance that was started earlier.
Restart	Restarts an instance that was running or stopped earlier.

Run-time Monitoring Statuses for Optimize Web Service Data Collector

The following table lists the run-time statuses that the Web Service Data Collector run-time component can return in response to the `sagcc get monitoring runtimestatus` and `sagcc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Description
STARTING	The Web Service Data Collector server is starting.
ONLINE	The Web Service Data Collector server is running.
NOT_READY	The Web Service Data Collector server is started, but it is not configured in the My webMethods Server administration user interface. If the Web Service Data Collector server is not configured, it cannot accept client requests.
STOPPING	The Web Service Data Collector server is stopping.
STOPPED	The Web Service Data Collector server has stopped.

Lifecycle Actions for Optimize Infrastructure Data Collector

The following table lists the actions that Infrastructure Data Collector supports with the `sagcc exec lifecycle` command and the operation taken against the Infrastructure Data Collector when an action is executed.

Action	Description
Start	Starts an instance that was stopped or not started.
Stop	Stops an instance that was started earlier.
Restart	Restarts an instance that was running or stopped earlier.

Run-time Monitoring Statuses for Optimize Infrastructure Data Collector

The following table lists the run-time statuses that the Infrastructure Data Collector run-time component can return in response to the `sagcc get monitoring runtimestatus` and `sagcc get monitoring state` commands, along with the meaning of each run-time status.

Run-time Status	Description
STARTING	The Infrastructure Data Collector server is starting.
ONLINE	The Infrastructure Data Collector server is running.
NOT_READY	The Infrastructure Data Collector server is started, but it is not configured in the My webMethods Server administration user interface. If the Infrastructure Data Collector server is not configured, it cannot accept client requests.
STOPPING	The Infrastructure Data Collector server is stopping.
STOPPED	The Infrastructure Data Collector server has stopped.

Security

The following topics provide information about how to use Central Configurator 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 Configurator

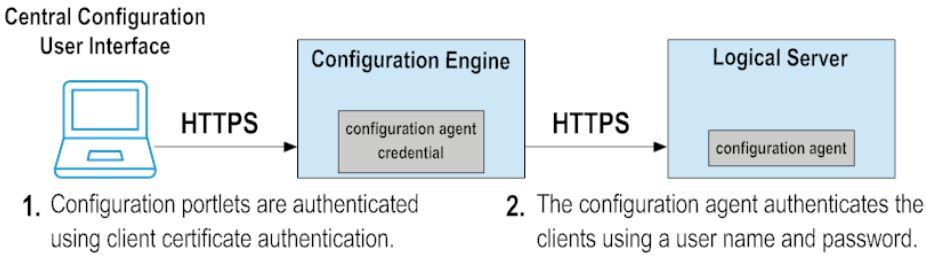
Central Configurator supports the use of SSL to secure and encrypt communications between the configuration agents and the configuration engine as well as between the configuration system's own internal communications. All that is needed to enable SSL is to obtain and install certificates in the webMethods 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 Configurator administrator files. Software AG also recommends that you change the master encryption password. All of these tasks are explained in this chapter.

Central Configurator 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 Configurator UI components and the Central Configurator Web application to provide client certificate authentication. All other authentication between Central Configurator and the BAM logical servers' configuration agents is done through simple user name and password authentication.

Figure 3. Central Configurator and SSL implementation



For instructions about installing certificates for SSL, see "Installing Certificates" on page 135.

Installing Certificates

You can add the authentication credentials for Central Configurator portlets by obtaining and installing certificates on the configuration engine and configuration agents. For more information about installing KeyStore and TrustStore certificates, see *Administering My webMethods Server*.

Securing the Central Configurator in My webMethods Server

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 Configurator 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. The SSL key and trust stores are 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 store 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

1. In Windows, open the *Software AG_directory*\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:

```
# SSL Properties
set.JAVA_KEYSTORE=Software AG_directory\MWS\server\default\config
    \security\localhost.p12
set.JAVA_KEYSTORETYPE=pkcs12
set.JAVA_KEYSTORE_PASSWORD=encrypted_password
set.JAVA_TRUSTSTORE=Software AG_directory\MWS\server\default\config
    \security\sagdemoca.jks
set.JAVA_TRUSTSTORETYPE=jks
set.JAVA_TRUSTSTORE_PASSWORD=encrypted_password
```

3. Save your changes to the serverproperties.xml file and close it.

4. Open the *Software AG_directory*\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 *Software AG_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.

10. Open the *Software AG_directory* \optimize\<*component*>\conf\glue\GlueSSLProperties.xml file in an appropriate text editor. Note that <*component*> should be replaced with the Optimize component for which SSL is being configured.

11. 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:

```
<entry key="keyStore">./security/ssl/glueKeyStore.jks</entry>
<entry key="keyStoreType">jsk</entry>
<entry key="keyStorePasswordHandle">keyStoreHandle</entry>

<entry key="trustStore">./security/ssl/glueTrustStore.jks</entry>
<entry key="trustStoreType">jsk</entry>
<entry key="trustStorePasswordHandle">trustStoreHandle</entry>
```

Note:
Paths must be relative to *Software AG_directory* \optimize\<*component*>\conf. For further information on adding a password-handle to the Optimize Password Manager, see " webMethods Password Administrator Utility" on page 137.

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.

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 Configurator 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 webMethods Server as illustrated in the above section.

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.dsand` `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 Configurator Administrator Files

The Central Configurator 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

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 Software AG Products*. See *Installing Software AG Products* for up-to-date, detailed installation information.

Before you begin a default Optimize installation and configuration, make sure you have the information about the components that is listed in the following table.

Component	Default Server:Primary Port
Analytic Engine	<*configuration agent*>:15000
Software AG Universal Messaging or Broker (deprecated)	<*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 is also helpful to have on hand a copy of *Installing Software AG Products* and *Administering webMethods Optimize*.

Database Information

A typical implementation of Optimize involves placing the five Optimize database component sets — Analysis, Process Tracker, Process Audit Log, and My webMethods Server — in five database schemas. For more information, see “Installing Optimize Database Component Sets" in *Administering webMethods Optimize* before you begin the installation.