WebSphere Process Server for z/OS

IBM

Common Event Infrastructure

Version 7.0.0

Contents

Common Event Infrastructure

Common Event Infrastructure is an embeddable technology intended to provide basic event management services to applications that require those services.

This event infrastructure serves as an integration point for consolidation and persistence of raw events from multiple, heterogeneous sources, and distribution of those events to event consumers. Events are represented using the Common Base Event model, a standard, XML-based format defining the structure of an event. For more information, see the Common Base Event model sub-topic.

By using this common infrastructure, diverse products that are not tightly coupled with one another can integrate their management of events, providing an end-to-end view of enterprise resources and correlating events across domain boundaries. For example, events generated by a network monitoring application can be correlated with events generated by a security application. Such correlation can be difficult to achieve when each product uses its own approach to event management.

Common Event Infrastructure provides facilities for generation, propagation, persistence, and consumption of events, but it does not define the events themselves. Instead, application developers and administrators define event types, event groups, filtering, and correlation.

Common Event Infrastructure components

Common Event Infrastructure consists of the following major components:

Common Base Event
>The Common Base Event component supports the creation of events and access to their property data. Event sources use the Common Base Event APIs to create new events conforming to the Common Base Event model; event consumers use the APIs to read property data from received events. In addition, applications can convert events to and from XML text format, supporting interchange with other tools. The Common Base Event component is part of the Eclipse Test and Performance Tools Platform (TPTP).

Emitter
>The emitter component supports the sending of events. After an event source creates an event and populates it with data, the event source submits the event to an emitter. The emitter optionally performs automatic content completion and then validates the event to ensure that it conforms to the Common Base Event specification. It also compares the event to configurable filter criteria. If the event is valid and passes the filter criteria, the emitter sends the event to the event service. An emitter can send events to the event service either synchronously (using Enterprise JavaBeans™ calls) or asynchronously (using a Java™ Message Service queue).

Event service
>The event service is the conduit between event sources and event consumers. The event service receives events submitted to emitters by event sources. It stores events in a persistent data store, and then

distributes them asynchronously to subscribed event consumers. In addition, the event service supports synchronous queries of historical events from the persistent store.

Event catalog

The event catalog is a repository of event metadata. Applications use the event catalog to retrieve information about classes of events and their permitted content.

In addition, an application or solution using Common Event Infrastructure might also include the following components (which are not part of the infrastructure itself):

Event source

An event source is any application that uses an emitter to send events to the event service.

Event consumer

An event consumer is any application that receives events from the event service.

Event catalog application

An event catalog application is any application that stores or retrieves event metadata in the event catalog. This might be a management or development tool; it might also be an event source or event consumer.

The following diagram shows the general flow of events from event source to event consumer using Common Event Infrastructure.
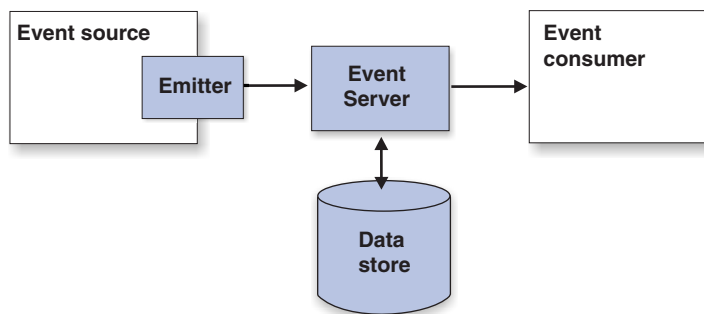
Common Base Event model

The Common Base Event model is a standard that defines a common representation of events that is intended for use by enterprise management and business applications. This standard, developed by the IBM® Autonomic Computing Architecture Board, supports encoding of logging, tracing, management, and business events using a common XML-based format, making it possible to correlate different types of events that originate from different applications. The Common Base Event model is part of the IBM Autonomic Computing Toolkit; for more information, see http://www.ibm.com/autonomic.

Common Event Infrastructure currently supports version 1.0.1 of the specification.

The basic concept behind the Common Base Event model is the *situation*. A situation can be anything that happens anywhere in the computing infrastructure, such as a server shutdown, a disk-drive failure, or a failed user login. The

Common Base Event model defines a set of standard situation types that accommodate most of the situations that might arise (for example, StartSituation and CreateSituation).

An *event* is a structured notification that reports information related to a situation. An event reports three kinds of information:
- The situation itself (what has happened)
- The identity of the affected component (for example, the server that has shut down)
- The identity of the component that is reporting the situation (which might be the same as the affected component)

The Common Base Event specification defines an event as an XML element containing properties that provide all three kinds of information. These properties are encoded as attributes and subelements of the root element, CommonBaseEvent.

The Common Base Event format is extensible. In addition to the standard event properties, an event can also contain extended data elements, which are application-specific elements that can contain any kind of information relevant to the situation. The *extensionName* attribute labels an event with an optional classification name (an event class), which indicates to applications what sort of extended data elements to expect. The event catalog stores event definitions that describe these event classes and their allowed content.

For complete details on the Common Base Event format, see the specification document and XSD schema included in the IBM Autonomic Computing Toolkit.

Configuring Common Event Infrastructure

You can configure Common Event Infrastructure resources, or change existing resources, using the server AdminTask object

About this task

Common Event Infrastructure (CEI) can be installed with a default configuration that is fully functional on a stand-alone server configuration. You would perform this task only when you create a stand-alone server. In all other cases, you to use the administrative console to configure CEI — such as when you are installing it in a network deployment environment or in a cluster — to ensure that the configuration is appropriate on your system.

You can also use the wsadmin command to configure CEI, or you can use the command to alter an existing CEI configuration. In either case, you would change the configuration of CEI by using the server AdminTask object to run administrative commands.

After changing CEI configuration, you must restart the server or cluster.

Common Event Infrastructure components

Common Event Infrastructure components are installed as a set of applications, services, and resources on the server.

When you configure Common Event Infrastructure, a number of components are created and deployed on your server.

Common Event Infrastructure service
> A service installed into the server, that enables applications and clients to use Common Event Infrastructure. You can view the configuration of the Common Event Infrastructure service in the administrative console, as follows:
> - For a server, select **Servers > Application Servers >** *server_name* **> Business Integration > Common Event Infrastructure > Common Event Infrastructure Service**.
> - For a cluster, select **Servers > Clusters >** *cluster_name* **> Business Integration > Common Event Infrastructure > Common Event Infrastructure Service**.
>
> If the check box labeled "Enable the event infrastructure server" is selected, then the service is installed and running or it will start after you restart your server or cluster. If it is cleared, then the service is not installed or will be uninstalled after you restart your server or cluster

Event service settings
> A set of properties used by the event service that enable event distribution and persistence using the data store. Typically, no configuration is necessary for this resource, but you might need to create additional event service settings if you want to set up multiple event services in the same cell. To view the event service settings, click **Service integration > Event service > Event service settings**.

Event messaging configuration

The resources that support asynchronous event transmission to the event service using the Java Messaging Service (JMS). The default messaging configuration uses the server embedded messaging. You can optionally configure an external JMS provider for event messaging.

Event database

The event database is used to persistently store events received by the event service. The Derby database is included as part of the server, but is not recommended for use in production environments. Instead, you can configure an external event database on the following products: DB2®, Oracle, SQLServer, and Informix®.

Event filter plug-in

A filter plug-in is used to filter events at the source using XPath event selectors. To configure the filter properties, click **Service Integration > Common Event Infrastructure > Event Emitter Factories > Event Filter Settings**.

Emitter factory

An emitter factory is an object used by event sources to create emitters; an emitter is used to send events to the event service. The properties of an emitter factory affect the behavior of any emitter that is created using that emitter factory. To view the available emitter factories, click **Service Integration > Common Event Infrastructure > Event Emitter Factories**.

Event service transmission

An event service transmission is an object defining properties that determine how emitters access the event service synchronously using EJB calls; these properties are used by emitter factories when creating new emitters. You can view or change the available event service transmissions from the emitter factory settings.

JMS transmission

A JMS transmission is an object that defines properties that determine how emitters access the event service asynchronously using a JMS queue; these properties are used by emitter factories when creating new emitters. You can view or change the available JMS transmissions from the emitter factory settings.

Event group

An event group is a logical collection of events used to categorize events according to their content. When querying events from the event service or subscribing to event distribution, an event consumer can specify an event group to retrieve only the events in that group. Event groups can also be used to specify which events should be stored in the persistent data store. To view the available event groups in the administrative console, click **Service integration > Common Event Infrastructure > Event service > Event services >** *event_service* **> Event groups**.

Configuring the Common Event Infrastructure using the administrative console

ConfigureCommon Event Infrastructure by using the server administrative console.

About this task

Open the Common Event Infrastructure Server panel of administrative console:

If you are configuring a server, select **Servers > Server Types > WebSphere application servers >** *server_name* **> Business Integration > Common Event Infrastructure > Common Event Infrastructure Server**.

If you are configuring a cluster, click **Servers > Clusters > WebSphere application server clusters >** *cluster_name* **> Business Integration > Common Event Infrastructure > Common Event Infrastructure Server**.

Procedure

1. Enable the deployment of the Common Event Infrastructure enterprise application by selecting the check box labeled **Enable the event infrastructure server**. If the server has already been configured, then you can enable or disable it by selecting or clearing the check box. If the enable check box is cleared then Common Event Infrastructure has not been configured, or has had a previous configuration disabled but the server has not been restarted. An information message shows you whether this deployment target has Common Event Infrastructure configured. If the server has already been configured, you can change the data source settings for the event database, the message store, or both.

   Note: If you select the check box to enable the Common Event Infrastructure server and the server has not yet been configured, then the parameters shown is used to configure it unless you change them.
   - If you are performing the configuration the first time, then the event data source tables are created on the common database. If there is already a Common Event Infrastructure server configuration, then you need to create a database.
   - The messaging service is created under a unique schema under the common database.

   When the server/cluster on which Common Event Infrastructure has been configured is restarted, then the new changes take effect.

2. Configure (or change the current settings for an existing configuration of) the event database by using one of the following methods to populate the fields with the appropriate settings.
   - Click **Edit** for a database configuration panel with a more extensive list of options than the ones listed on the panel.
   - Use the fields on the panel to enter the information:
   a. **Database name** – the name of the database you use to store events.
   b. **Create Tables** – select this check box if you want to create the database tables on the event database.

      Note: If you are configuring Common Event Infrastructure to use a database on another server, then you are not be able to create the tables using this control. Instead, you will have to use the database scripts that will be generated after you complete the rest of this configuration. In this case, you can click **Edit** to show the data source detail panel, which tells you the location of the database creation scripts.
   c. **Username** and **Password** – for authenticating into the event database.
   d. **Server** – name of the server where the event database is located.
   e. **Provider** – choose a provider for your database from the menu.

      Note: The **Schema** field is activated only if the database is created using DB2 on an iSeries® or z/OS® platform. In all other cases, the schema field is disabled.

Important: If the tables exist on the target database, then the configuration can fail.

3. Select whether the Common Event Infrastructure bus is to be**Local** on the server, or **Remote** and reside on another server. If you choose remote, then select the remote location from the menu or click **New** to create a new remote bus.

4. Configure Common Event Infrastructure support for messaging.

   • Click Edit for a database configuration panel with a more extensive list of options than the ones listed on the panel.

   • Use the fields on the panel to enter the information:

   a. **Database name** – enter the name of the database you use to store messages.

   b. **Schema** – enter a name for the schema, or accept the default name given.

   c. **Username** and **Password** – for authenticating into the messaging database.

   d. **Server** – name of the server where the messaging database is located.

   e. **Provider** – choose a provider for your database from the menu.

5. Click **OK** or **Apply**.

6. Restart your server or cluster.

Results

All the major parts of Common Event Infrastructure are now configured and running on your server or cluster. The configuration includes the event data store, the messaging engine, and the event application. This single panel can be used in place of many commands and steps you would otherwise use to configure Common Event Infrastructure.

What to do next

After you have restarted your server or cluster, you will be able to store service component events that are emitted from your applications. You can now change the runtime properties of the Common Event Infrastructure server by selecting the **Common Event Infrastructure Destination** panel. You can choose whether to start the Common Event Infrastructure server at startup, and specify the emitter factory JNDI name where the events are sent.

Deploying the Common Event Infrastructure application

Before you can use Common Event Infrastructure, you must first deploy the event service and associated resources in the server runtime environment.

About this task

The Common Event Infrastructure enterprise application includes the runtime components of the event service and the default messaging configuration used for asynchronous event submission.

To deploy the event service:

Procedure

From the wsadmin tool, run the **deployEventService** administrative command in batch or interactive mode. The parameters of the **deployEventService** administrative command are as follows:

nodeName
> The name of the node where the event service to be deployed. This parameter is optional; if you do not specify a node name, the default is the current node. If you specify a node name, then you must also specify the server name using the **serverName** parameter. This parameter is not valid if you are deploying the event service in a cluster.

serverName
> The name of the server where the event service to be deployed. This parameter is required only if you specify a node; it is not valid if you are deploying the event service in a cluster.

clusterName
> The name of the cluster where the event service to be deployed. This parameter is optional and must not be specified if you are deploying at the node or server scope.

enable
> Indicates whether the event service to be started automatically when the server starts. The default value is `true`.

Results

After the administrative command completes, the Common Event Infrastructure event service and default messaging configuration are deployed at the specified scope.

What to do next

If WebSphere® security is enabled, you must also configure the JMS authentication alias and password using the **setEventServiceJmsAuthAlias** administrative command.

If you are deploying the event service in a cluster, you must also manually configure the event database.

Deploying Common Event Infrastructure in a cluster

There are several ways you can deploy Common Event Infrastructure resources in a cluster environment.

Deploying Common Event Infrastructure in an existing cluster
You can deploy the event service application in an existing cluster.

About this task

Deploying the event service application in a cluster is essentially the same as deploying the application on a stand-alone server. However, in a cluster environment, no default event database is configured.

To deploy and configure Common Event Infrastructure in a cluster environment:

Procedure
1. Run the **deployEventService** administrative command as you would for a stand-alone server, but specifying the name of the cluster. Use the clusterName parameter to specify the cluster.

2. On the deployment manager system, run the database configuration administrative command. Specify the cluster name using the clusterName parameter. This command generates the database configuration script.
3. Copy the generated database configuration script to the database system.
4. Run the database configuration script on the database system to create the event database.
5. On the deployment manager system, run the **enableEventService** command to enable the event service. Use the clusterName parameter to specify the name of the cluster.

Creating a cluster by converting an existing Common Event Infrastructure server

You can create a cluster by converting an existing stand-alone server that is already configured with Common Event Infrastructure.

Before you begin

Before you can convert the existing server, make sure that it is fully configured for Common Event Infrastructure. The configuration includes deploying the event service application and configuring the event database.

About this task

To create the cluster:

Procedure

1. Follow the typical WebSphere process for converting a stand-alone server into the first member of a new cluster. When the server is converted, the following steps take place:
   - Common Event Infrastructure resources available at the scope of the server are moved to the new cluster scope.

     Default database: If the existing server is configured with the default Derby database, the database resources are not moved to the cluster scope. Instead, these resources are removed. The default database configuration is not supported in a cluster. In this situation, the event service in the cluster is disabled by default.
   - The deployed event service application target list is modified to remove the converted server and add the new cluster.
2. Optional: If the converted server was configured with the default Derby database, you must configure a new event database for the cluster and then enable the event service:
   a. On the deployment manager system, run the database configuration administrative command. Specify the cluster name using the clusterName parameter. This command generates the database configuration script.
   b. Copy the generated database configuration script to the database system.
   c. Run the database configuration script on the database system to create the event database.
   d. On the deployment manager system, run the **enableEventService** command to enable the event service. Use the clusterName parameter to specify the name of the cluster.

Creating a cluster by using an existing Common Event Infrastructure server as a template

You can create a cluster by specifying an existing Common Event Infrastructure server as a template.

Before you begin

Before you can create a cluster using this method, you must have an existing server that is fully configured for Common Event Infrastructure. The configuration includes deploying the event service application and configuring the event database.

About this task

To create the cluster:

Procedure

1. Follow the typical WebSphere process for creating new cluster, using the existing Common Event Infrastructure server as a template for the first cluster member. When the first member is created, the following steps take place:

   • Common Event Infrastructure resources available at the scope of the existing server are copied to the new cluster scope.

      Default database: If the existing server is configured with the default Derby database, the database resources are not copied to the cluster scope. The default database configuration is not supported in a cluster. In this situation, the event service in the cluster is disabled by default.

   • The deployed event service application target list is modified to include the new cluster.

2. Optional: If the existing server was configured with the default Derby database, you must configure a new event database for the cluster and then enable the event service:

   a. On the deployment manager system, run the database configuration administrative command. Specify the cluster name using the clusterName parameter. This command generates the database configuration script.

   b. Copy the generated database configuration script to the database system.

   c. Run the database configuration script on the database system to create the event database.

   d. On the deployment manager system, run the **enableEventService** command to enable the event service. Use the clusterName parameter to specify the name of the cluster.

Configuring event messaging

You can modify the messaging configuration used for JMS transmission of events to the event service.

About this task

You will create the messaging infrastructure for Common Event Infrastructure when you use the administrative console panel to configure Common Event Infrastructure on a server. Generally, the messaging configuration will use the default messaging provider and create a single JMS queue for asynchronous

transmission of events to the event service. You can, if necessary, modify this messaging configuration.

Configuring additional JMS queues

If you are using the default event messaging configuration, you can add additional JMS queues for transmission of events to the event service.

About this task

To configure an additional JMS queues using the default messaging configuration, you can set up multiple JMS queues that are routed to the service integration bus queue destination. The Common Event Infrastructure service integration bus queue destination depends upon the scope at which the event service is deployed:

Scope	Service integration bus queue destination
Server	*node.server*.CommonEventInfrastructureQueueDestination
Cluster	*cluster*.CommonEventInfrastructureQueueDestination

For more information about service integration bus configuration, refer to the documentation.

Configuring event messaging using an external JMS provider

If you do not want to use the default embedded messaging configuration for event transmission, you can configure asynchronous message transport to use an external Java Messaging Service (JMS) provider.

Before you begin

Before you can configure event messaging using an external JMS provider, you must first create a JMS queue and connection factory using the appropriate interfaces for your JMS provider. You must also create a listener port or activation specification.

About this task

To configure event messaging using an external JMS provider:

Procedure

From the wsadmin tool, run the **deployEventServiceMdb** administrative command in batch or interactive mode. The parameters of the **deployEventServiceMdb** command are as follows:

applicationName
　　The application name of the event service message-driven bean to be deployed. This parameter is required.

nodeName
　　The name of the node where the event service message-driven bean is to be deployed. If you specify a node name, you must also specify a server name. The node name is an optional parameter; the default value is the current node. Do not specify this parameter if you are deploying the application in a cluster.

serverName
　　The name of the server where the event service message-driven bean is to be

deployed. This parameter is required if you are deploying the application at server scope; otherwise it is optional. Do not specify a server name if you are deploying the application in a cluster.

clusterName

The name of the cluster where the event service message-driven bean is to be deployed. Specify this parameter only if you are deploying the application in a cluster.

listenerPort

The name of the listener port used by the event service message-driven bean to publish events. The specified listener port must exist. You must specify either a listener port or an activation specification, but not both.

activationSpec

The JNDI name of the activation specification used by the event service message-driven bean to publish events. The specified activation specification must exist. You must specify either a listener port or an activation specification, but not both.

qcfJndiName

The JNDI name of the JMS queue connection factory to be used by the event service message-driven bean. This parameter is required if you specify an activation specification; otherwise it is optional. If you specify a queue connection factory and a listener port, the queue connection factory must match the one configured for the listener port.

Results

The **deployEventServiceMdb** administrative command deploys the message-driven bean for the event service, configured for the specified listener port or activation specification. It also creates an emitter factory and JMS transmission using the external JMS configuration. Applications can use either the default emitter factory (which is configured to use the default messaging configuration) or the new emitter factory (which uses the external JMS provider).

What to do next

If you want to set up more than one JMS queue to the event service, you can run this command multiple times, specifying different enterprise application names and JMS queues. Each time you run the script, it deploys an additional message-driven bean and configures new resources to use the specified JMS queue.

Configuring the JMS authentication alias

If WebSphere security is enabled and you want to use asynchronous JMS messaging to submit events to the event service, you must configure the JMS authentication alias.

About this task

To configure the JMS authentication alias:

Procedure

From the wsadmin tool, run the **setEventServiceJmsAuthAlias** administrative command in batch or interactive mode. The parameters of the **setEventServiceJmsAuthAlias** command are as follows:

userName
>The name of the user to be used for the JMS authentication alias. This parameter is required.

password
>The password of the user to be used for the JMS authentication alias. This parameter is required.

nodeName
>The name of the node where you want to update or create the JMS authentication alias. If you specify a node name, you must also specify a server name. Do not specify a node name if you are configuring the authentication alias in a cluster.

serverName
>The name of the server where you want to update or create the JMS authentication alias. This parameter is required only if you specify a node; it is not valid if you are configuring the authentication alias in a cluster.

clusterName
>The name of the cluster where you want to update or create the JMS authentication alias. Specify this parameter only if you are configuring the authentication alias in a cluster; if you specify a cluster name, do not specify a node or server name.

Results

The JMS authentication alias used by the event service objects is updated at the specified scope; if the authentication does not exist, it is created using the specified values.

Populating the event database

The event database is required to support persistence of events. If you ran the createDB.sh script, you have already created and populated the event database. If you did not use the createDB.sh script, run the generated DDL scripts to populate the event database.

Before you begin

Before you start, you must have already created the event database; see Creating the DB2 databases and storage groups using DButility.sh, SPUFI, or DSNTEP2.

About this task

Populate the event database using the DDL scripts that were generated by the administrative console.

Procedure
1. Copy, to your working directory, the following generated DDL script files from `/WebSphere/V6R1M0/DeploymentManager/profiles/default/databases/event/` *cluster_name*`/dbscripts/db2zos/ddl` (where *cluster_name* is the name of the cluster to which the network deployment cell belongs):
   - catalogSeed.ddl
   - cr_db.ddl
   - cr_db_catalog.ddl
   - cr_tbl.ddl

- cr_tbl_catalog.ddl
- ins_metadata.ddl

2. Assign the appropriate permissions to the copy of each file:

   `chmod 755 catalogSeed.ddl`

3. Edit the values in the file to suit your needs. Change the database names and the storage group names to meet your naming requirements. The names that you specify in the file must match the values that you entered in the response file that provided input to the configuration script.

   Note: The files are provided in ASCII format. If the tools that you use to view, edit, and run the scripts require the scripts to be in EBCDIC format, use the iconv command to convert the file to EBCDIC. For example:

   `iconv -t IBM-1047 -f ISO8859-1 catalogSeed.ddl >`
   `catalogSeed_EBCDIC.ddl`

   If you the convert the file from ASCII format to EBCDIC but need to run the file in ASCII format, use iconv to convert the file back to ASCII. For example:

   `iconv -t ISO8859-1 -f IBM-1047 catalogSeed_EBCDIC.ddl >`
   `catalogSeed.ddl`

4. Run the customized scripts using the tool of your choice. For example, DBUtility.sh or SPUFI.

Results

The event database is now populated.

Upgrading a DB2 for z/OS event database from a previous version

If you have an existing DB2 event database from Version 5.1 of Common Event Infrastructure on a z/OS system, you must upgrade it to the current version.

About this task

To upgrade a DB2 event database on a z/OS system:

Procedure

1. Make a backup copy of the existing event database.
2. Go to the *profile_root*/bin directory.
3. Run the DB2 for z/OS upgrade script:
   - `eventUpgradeDB2ZOS.sh runUpgrade=[true|false] dbUser=user`
     `[dbName=name] [dbPassword=pw]`
     `[scriptDir=dir] storageGroup=group`
     `bufferPool4K=4kbufpool bufferPool8k=8kbufpool`
     `bufferPool16K=16kbufpool`

   The typical required parameters are as follows:

   runUpgrade
   Indicates whether you want the upgrade script to automatically run the generated DDL scripts to complete the database upgrade. This parameter is required. Specify false if you want to manually upgrade the database at a later time or on a different system.

z/OS systems: This parameter is ignored on a native z/OS system. Automatically running the generated DDL scripts is supported only on a client system.

dbUser
Specifies the DB2 user ID to use. This parameter is required.

dbName
Specifies the DB2 database name. The default name for the event database is event. This parameter is required if you specified runUpgrade=true.

dbPassword
Specifies the password for the specified DB2 user ID. This parameter is optional; if you do not specify a password, DB2 prompts you to type it.

scriptDir
Specifies the directory you want to contain the generated DDL scripts. This parameter is optional; if you do not specify a directory, the scripts are stored in the .\eventDBUpgrade\db2zos directory.

storageGroup
Specifies the name of the storage group. This parameter is required.

bufferPool4K
Specifies the name of the 4K buffer pool. This parameter is required.

bufferPool8K
Specifies the name of the 8K buffer pool. This parameter is required.

bufferPool16K
Specifies the name of the 16K buffer pool. This parameter is required.

To see a complete list of parameters and usage information, run the **eventUpgradeDB2ZOS** script with no parameters.

Results

The upgrade script generates the required DDL scripts for upgrading the event database. If you specified runUpgrade=true on a client system, the DDL scripts are automatically run, completing the upgrade.

What to do next

You must now manually run the generated DDL scripts using the SQL Processor Using File Input (SPUFI) facility. This step completes the database upgrade.

Removing Common Event Infrastructure configuration

If you need to remove the configuration for Common Event Infrastructure, in preparation for uninstalling the server, you must first remove the deployed enterprise applications and the database configuration.

About this task

Common Event Infrastructure is installed in the base installation of server, but will not be active unless it is already configured. This topic tells you only how to remove a previously configured instance of Common Event Infrastructure. To remove the configuration for Common Event Infrastructure, follow these steps:

Removing the event database

To remove the event database, you can use the appropriate administrative command for the type of database.

About this task

To remove the event database:

Procedure

1. Start the wsadmin tool.
2. Use the AdminTask object to run the appropriate administrative command for your event database:

Database type	Command
"removeEventServiceDerbyDB command" on page 80	
DB2 on Linux®, UNIX®, and Windows® systems	
DB2 on z/OS systems	"removeEventServiceDB2ZOSDB command" on page 79
DB2 on iSeries systems	
Informix	
Oracle	

Database-specific notes:

- On a z/OS system, the administrative command only removes the JDBC data source. To remove the database, you must use SPUFI to run the database removal script generated during database creation. By default, this script is located in the configuration file system in the: `/WebSphere/V7R0/DeploymentManager/profiles/default/databases/event/`*`cluster_name`*`/dbscripts/db2zos` directory.
- On an iSeries system, the administrative command only removes the JDBC data source. To remove the database on the iSeries system, drop the collection that was created for the database.

The required parameters (for example, user IDs and passwords) vary by database type. For a complete list of parameters and usage information, refer to the help for the administrative command.

Removing the Common Event Infrastructure application

To manually remove the event service enterprise application and resources from the server, you can use the **removeEventService** administrative command.

About this task

To remove the event service enterprise application:

Procedure

1. Start the wsadmin tool.

2. Use the AdminTask object to run the **removeEventService** administrative command in batch or interactive mode. The parameters of the **removeEventService** administrative command are as follows:

nodeName
The name of the node where the event service is deployed. This parameter is optional; if you do not specify a node name, the default is the current node. If you specify a node name, then you must also specify the server name using the **serverName** parameter. This parameter is not valid if you are removing the event service from a cluster.

serverName
The name of the server where the event service is deployed. This parameter is required only if you specify a node; it is not valid if you are removing the event service from a cluster.

clusterName
The name of the cluster where the event service is deployed. This parameter is optional and must not be specified if you are removing the event service from a server.

Removing event messaging from the Common Event Infrastructure server

To remove the event service messaging configuration for an external JMS provider, you can use the removeEventServiceMdb administrative command.

About this task

This command removes the message-driven bean deployed for the JMS configuration. To remove the event service message-driven bean:

Procedure

1. Start the wsadmin tool.
2. Use the AdminTask object to run the **removeEventServiceMdb** administrative command in batch or interactive mode. The parameters of the **removeEventServiceMdb** administrative command are as follows:

applicationName
The application name of the deployed event service message-driven bean.

nodeName
The name of the node where the event service message-driven bean is deployed. If you specify a node name, you must also specify a server name. The node name is an optional parameter; the default value is the current node. Do not specify this parameter if you are removing the application from a cluster.

serverName
The name of the server where the event service message-driven bean is deployed. This parameter is required if you are removing the application from a server; otherwise it is optional. Do not specify a server name if you are removing the application from a cluster.

clusterName
The name of the cluster where the event service message-driven bean is deployed. Specify this parameter only if you are removing the application from a cluster.

Administering Common Event Infrastructure

These topics describe several administrative tasks you can use to control the operation of the Common Event Infrastructure components at run time.

Administering the event service using the administrative console

You can use the Web-based administrative console to administer the event service.

Enabling and disabling the event service using the administrative console

You can enable and disable the event service by modifying the properties of the event service in the server administrative console.

About this task

If the event service is enabled, it automatically starts when the server starts.

To enable or disable the event service from the administrative console:

Procedure

1. Select one of the following methods to go to the appropriate administrative console panel:
   - Open the Common Event Infrastructure Server panel of administrative console
     - For servers, select **Servers** → **Server Types** → **WebSphere application servers** → *server_name* → **Business Integration** → **Common Event Infrastructure** → **Common Event Infrastructure Server**.
     - For clusters, select **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* → **Business Integration** → **Common Event Infrastructure** → **Common Event Infrastructure Server**.
   - Alternatively, you can open the Container Services to perform this task:
     - For a servers, click **Servers** → **Server Types** → **WebSphere application servers** → *server_name* → **Container Services** → **Common Event Infrastructure Service**.
     - For clusters, click **Servers** → **Clusters** → **WebSphere application server clusters** → *cluster_name* → **Cluster members** → *server* → **Container Services** → **Common Event Infrastructure Service**.
2. Select or deselect the **Enable service at server startup** property. If the check box is selected, the Common Event Infrastructure service starts when the server starts.
3. Save the configuration changes.
4. In a network deployment environment only, synchronize the node.
5. Restart the servers or cluster.

Creating an event emitter factory using the administrative console

An emitter factory is used by event sources to create emitters.

About this task

The properties of an emitter factory affect the behavior of any emitter that is created using that emitter factory. You can use the default emitter factory or create additional emitter factories for your event sources to use. You might also want to create an additional emitter factory to specify a different transaction mode or event transmission. To view the event emitter that was created after you configured Common Event Infrastructure (CEI) on the Common Event Infrastructure Destination panel:

1.
    - For a single server, select **Servers > Server Types > WebSphere application servers > *server_name***.
    - For a cluster, select **Servers > Clusters > WebSphere application server clusters*cluster_name***.
2. On the **Configuration** tab, select **Business Integration > Common Event Infrastructure > Common Event Infrastructure Destination**.
3. You can choose an existing event emitter factory from the menu of JNDI names, or specify one in the text field.

You must resolve JNDI names to a remote server if the CEI server not local. See the WebSphere Application Server documentation for more information about naming within an ND environment. To create an emitter factory:

Procedure

1. In the server administrative console, click **Service integration** > **Common Event Infrastructure** > **Event emitter factories** > **New**.
2. Specify the properties of the new emitter factory. Refer to the online help for the Emitter Factory Settings page for detailed information about these properties.
3. Save the configuration changes.
4. In a network deployment environment only, synchronize the node.
5. Restart the server.

Results

Event sources can now use the configured emitter factory to create emitters.

Creating an event group using the administrative console

An event group defines a logical collection of events based on the content of their property data. An event group can be used when querying events from the event service, and it can optionally be associated with a JMS destination for asynchronous event distribution.

About this task

To create an event group:

Procedure

1. Optional: Set up one or more JMS destinations for the event group. An event group can be associated with one JMS topic, and one or more JMS queues. Refer to the documentation for your JMS provider for information about how to create JMS destinations and connection factories and bind them into a JNDI namespace.

Security: If WebSphere security is enabled, the configuration for the JMS destination must specify an authentication alias.

2. Create the new event group. In the server administrative console, click **Service Integration** > **Common Event Infrastructure** > **Event Service** > **Event Services** > *event_service* > **Event Groups** > **New**.

3. Specify the properties of the new event group, including the event selector and optional JMS destinations.

4. Save the configuration changes.

5. In a network deployment environment only, synchronize the node.

6. Restart the server.

Results

Event consumers can now specify the event group when querying events. If event distribution is enabled in the event service settings, events belonging to the event group are also published to any JMS destinations specified in the event group. Event consumers can then receive events asynchronously by subscribing to the appropriate destinations.

Creating an event filter using the administrative console

An event filter defines properties used by the default filter plug-in, which is used by emitters to filter events at the source.

About this task

An event filter can be specified as part of the configuration of an emitter factory. Any emitter created using that emitter factory then uses the specified filter to determine which event should be sent to the event service.

To create an event filter:

Procedure

1. In the server administrative console, click **Service Integration** > **Common Event Infrastructure** > **Event Emitter Factories** > *emitter_factory* > **Event Filters** > **New**.

2. Specify the properties of the new event filter. Refer to the online help for the Event Filter Settings page for detailed information about these properties.

3. Save the configuration changes.

4. In a network deployment environment only, synchronize the node.

5. Restart the server.

Results

Event emitters can now use the configured filter to determine which events to send to the event service.

Administering the event service using scripting

You can use scripting interfaces to administer the event service.

Enabling the event service using scripting

You can enable the event service by running an AdminTask administrative command using the wsadmin tool.

About this task

If the event service is enabled, it automatically starts when the server starts.

To enable the event service using the wsadmin tool:

Procedure

1. Start the wsadmin tool.
2. Use the AdminTask object to run the **enableEventService** administrative command:
   - Using Jacl:
     ```
     # enable event service at server scope
     $AdminTask enableEventService { -nodeName node1 -serverName server1 }

     # enable event service at cluster scope
     $AdminTask enableEventService  { -clusterName cluster1 }
     ```
   - Using Jython:
     ```
     # enable event service at server scope
     AdminTask.enableEventService('[ -nodeName node1 -serverName server1 ]')

     # enable event service at cluster scope
     AdminTask.enableEventService('[ -clusterName cluster1 ]')
     ```
   The parameters of the **enableEventService** command are as follows:

   nodeName
   > The name of the node where the event service should be enabled. This is an optional parameter; the default value is the current node. If you specify a node name, you must also specify a server name. Do not specify a node if you are enabling the event service in a cluster.

   serverName
   > The name of the server where the event service should be enabled. This parameter is required if you specify a node name. Do not specify a server name if you are enabling the event service in a cluster.

   clusterName
   > The name of the cluster where the event service should be enabled. This parameter is required if you are enabling the event service in a cluster.

3. Restart the server.

Disabling the event service using scripting

You can disable the event service by running an AdminTask administrative command using the wsadmin tool.

About this task

If the event service is disabled, it does not automatically start when the server starts.

To disable the event service using the wsadmin tool:

Procedure

1. Start the wsadmin tool.
2. Use the AdminTask object to run the **disableEventService** administrative command:
   - Using Jacl:
     ```
     # disable event service at server scope
     $AdminTask disableEventService { -nodeName node1 -serverName server1 }

     # disable event service at cluster scope
     $AdminTask disableEventService  { -clusterName cluster1 }
     ```
   - Using Jython:
     ```
     # disable event service at server scope
     AdminTask.disableEventService('[ -nodeName node1 -serverName server1 ]')

     # disable event service at cluster scope
     AdminTask.disableEventService('[ -clusterName cluster1 ]')
     ```
   The parameters of the **disableEventService** command are as follows:

   nodeName
   > The name of the node where the event service should be disabled. This is an optional parameter; the default value is the current node. If you specify a node name, you must also specify a server name. Do not specify a node if you are disabling the event service in a cluster.

   serverName
   > The name of the server where the event service should be disabled. This parameter is required if you specify a node name. Do not specify a server name if you are disabling the event service in a cluster.

   clusterName
   > The name of the cluster where the event service should be disabled. This parameter is required if you are disabling the event service in a cluster.

3. Restart the server.

Creating an emitter factory with scripting

You can create an emitter factory using a Jacl or Jython script.

Before you begin

Before starting this task, the wsadmin tool must be running. See the WebSphere Application Server documentation for more information.

About this task

Perform the following steps to configure a new emitter factory.

Procedure

1. Identify the Common Event Infrastructure provider ID.
   - Using Jacl:
     ```
     set providerid [$AdminConfig getid \
     /Cell:mycell/Node:mynode/Server:myserver/ \
     EventInfrastructureProvider:/]
     ```
   - Using Jython:
     ```
     providerid =
     AdminConfig.getid('/Cell:mycell/Node:mynode/Server:myserver/EventInfrastructureProvider:/')
     print providerid
     ```

Example output:

```
EventInfrastructureProvider(cells/mycell/nodes/mynode/servers/myserver|resources-cei.xml#
  EventInfrastructureProvider_1)
```

2. Set the required attributes.

- Using Jacl:

```
set Name [list name "EmitterName"]
set JndiName [list jndiName "Put JNDI name for new emitter factory here"]
set Description [list description "Put description here"]
set Category [list category "Put category here"]

# set TransactionMode to true to send each event in a new transaction
set TransactionMode [list preferredTransactionMode "false"]

# set SynchronizationMode to true to use synchronous event transmission
# as the preferred synchronization mode
set SynchronizationMode [list preferredSynchronizationMode "true"]

# leave blank if synchronous transmission is not supported
set SyncJNDIName [list synchronousTransmissionProfileJNDIName \
  "Put JNDI name of synchronous transmission profile here "]

# leave blank if asynchronous transmission is not supported
set AsyncJNDIName [list asynchronousTransmissionProfileJNDIName \
  "Put JNDI name of asynchronous transmission profile here "]

set FilteringEnabled [list filteringEnabled "false"]

# leave blank if filtering is not enabled
set FilterJNDIName [list filterFactoryJNDIName \
  "Put JNDI name of event filter here"]

# custom properties include compatibility mode
set CompatibilityMode [list [list name compatibilityMode] \
  [list description ""] \
  [list required false] \
  [list type java.lang.Boolean] \
  [list value "false"] ] \
set resProp [list [list resourceProperties [list \
  $CompatibilityMode ]]]
```

- Using Jython:

```
Name = ['name', 'EmitterName']
JndiName = ['jndiName', 'Put JNDI name for new emitter factory here']
Description = ['description', 'Put description here']
Category = ['category', 'Put category here']

# set TransactionMode to true to send each event in a new transaction
TransactionMode = ['preferredTransactionMode', 'false']

# set SynchronizationMode to true to use synchronous event transmission
# as the preferred synchronization mode
SynchronizationMode = ['preferredSynchronizationMode', 'true']

# leave blank if synchronous transmission is not supported
SyncJNDIName = ['synchronousTransmissionProfileJNDIName', \
                'Put JNDI name of synchronous transmission profile here']

# leave blank if asynchronous transmission is not supported
AsyncJNDIName = ['asynchronousTransmissionProfileJNDIName', \
                'Put JNDI name of asynchronous transmission profile here']

FilteringEnabled = ['filteringEnabled', 'false']

# leave blank if filtering is not enabled
FilterJNDIName = ['filterFactoryJNDIName', 'Put JNDI name of event filter here']
```

```
# custom properties include compatibility mode
compatibilityName = ['name','compatibilityMode']
compatibilityDescription = ['description','']
compatibilityRequired = ['required', 'false']
compatibilityType = ['type', 'java.lang.Boolean']
compatibilityValue = ['value', 'false']
CompatibilityMode = [compatibilityName, compatibilityDescription, compatibilityRequired, \
                     compatibilityType, compatibilityValue]
customProperties = ['propertySet', [['resourceProperties', [CompatibilityMode]]]]
```

3. Set the properties for the new emitter factory.

- Using Jacl:

```
set properties "[list $Name $JndiName $Description $Category \
    $TransactionMode $SynchronizationMode $AsyncJNDIName \
    $SyncJNDIName $FilteringEnabled $FilterJNDIName \
                  [list propertySet $resProp]]"
```

- Using Jython:

```
properties = [Name,JndiName,Description,Category,TransactionMode,
SynchronizationMode,AsyncJNDIName,SyncJNDIName,FilteringEnabled,
FilterJNDIName, customProperties]
print properties
```

Example output:

```
[['name', 'EmitterName'], ['jndiName', 'Put JNDI name for new
emitter factory here'], ['description', 'Put description here'],
['category', 'Put category here'], ['preferredTransactionMode',
'false'], ['preferredSynchronizationMode', 'true'],
['asynchronousTransmissionProfileJNDIName', 'Put JNDI name of
asynchronous transmission profile here '],
['synchronousTransmissionProfileJNDIName', 'Put JNDI name of
synchronous transmission profile here '], ['filteringEnabled',
'false'], ['filterFactoryJNDIName', 'Put JNDI name of event
filter here'], ['propertySet', [['resourceProperties', [[['name',
'compatibilityMode'], ['description', ''], ['required', 'false'],
['type', 'java.lang.Boolean'], ['value', 'false']]]]]]]]
```

4. Create the emitter factory.

- Using Jacl:

```
set emitterProf [$AdminConfig create EmitterFactoryProfile \
$providerid $properties]
```

- Using Jython:

```
print AdminConfig.create('EmitterFactoryProfile', providerid, properties)
```

Example output:

```
EmitterName(cells/mycell/nodes/mynode/servers/myserver|resources-cei.xml#EmitterFactoryProfile_1)
```

5. Save the configuration changes.
6. In a network deployment environment only, synchronize the node.
7. Restart the server.

Creating an event group with scripting

You can create an event group using a Jacl or Jython script.

Before you begin

Before starting this task, the wsadmin tool must be running. See the WebSphere
Application Server documentation for more information.

About this task

Perform the following steps to configure a new event group.

Procedure

1. Identify the Common Event Infrastructure provider ID.

   - Using Jacl:

```
set providerid [$AdminConfig getid \
/Cell:mycell/Node:mynode/Server:myserver/ \
EventInfrastructureProvider:/]
```

   - Using Jython:

```
providerid =
AdminConfig.getid
 ('/Cell:mycell/Node:mynode/Server:myserver/EventInfrastructureProvider:/')
 print providerid
```

   Example output:

```
EventInfrastructureProvider(cells/mycell/nodes/mynode/servers/myserver|
 resources-cei.xml#EventInfrastructureProvider_1)
```

2. Obtain the event group list.

   - Using Jacl:

```
set eventGroupProfileId [lindex [$AdminConfig list
EventGroupProfileList $providerid] 0]
```

   - Using Jython:

```
eventGroupProfileId = AdminConfig.list('EventGroupProfileList',providerid)
```

3. Set the required attributes.

   - Using Jacl:

```
set name [ list eventGroupName "EventGroupName" ]

# escape all '[' characters in the event selector string.
set selectorString [ list eventSelectorString "Set event selector here"]

# leave blank if events should not be published using JMS
set JNDIName [ list topicJNDIName "Set topic JNDI name here"]

# leave blank if events should not be published using JMS
set connectionFactoryJNDIName [ list topicConnectionFactoryJNDIName \
  "Set topic connection factory JNDI name here" ]

set persistEventsFlag [ list persistEvents "true" ]

# custom properties include compatibility mode
set CompatibilityMode [list [list name compatibilityMode] \
  [list description ""] \
  [list required false] \
  [list type java.lang.Boolean] \
  [list value "false"] ] \
set resProp [list [list resourceProperties [list \
  $CompatibilityMode ]]]
```

   - Using Jython:

```
Name = ['eventGroupName', 'EventGroupName' ]
SelectorString = ['eventSelectorString', 'Set event selector here']

# leave blank if events should not be published using JMS
JNDIName = [ 'topicJNDIName', 'Set topic JNDI name here']

# leave blank if events should not be published using JMS
ConnectionFactoryJNDIName = ['topicConnectionFactoryJNDIName', \
  'Set topic connection factory JNDI name here']
```

```
PersistEventsFlag = ['persistEvents', 'true']

# custom properties include compatibility mode
compatibilityName = ['name','compatibilityMode']
compatibilityDescription = ['description','']
compatibilityRequired = ['required', 'false']
compatibilityType = ['type', 'java.lang.Boolean']
compatibilityValue = ['value', 'false']
CompatibilityMode = [compatibilityName,
                     compatibilityDescription,
                     compatibilityRequired, \
                     compatibilityType,
                     compatibilityValue]
customProperties = ['propertySet', [['resourceProperties',
                    [CompatibilityMode]]]]
```

4. Set the properties for the new event group.

- Using Jacl:

```
set properties [ list $name $selectorString $JNDIName \
$connectionFactoryJNDIName $persistEventsFlag [list propertySet \
$resProp]]
```

- Using Jython:

```
properties = [Name,SelectorString,JNDIName,ConnectionFactoryJNDIName, \
              PersistEventsFlag,customProperties]
```

Example output:

```
[['eventGroupName', 'EventGroupName'], ['eventSelectorString',
'Set event selector here'], ['topicJNDIName', 'Set topic JNDI
name here'], ['topicConnectionFactoryJNDIName', 'Set topic
connection factory JNDI name here'], ['persistEvents','true'],
['propertySet', [['resourceProperties', [[['name',
'compatibilityMode'], ['description', ''], ['required', 'false'],
['type', 'java.lang.Boolean'], ['value', 'false']]]]]]]
```

5. Create the event group.

- Using Jacl:

```
set result [ $AdminConfig create EventGroupProfile
 $eventGroupProfileId $properties ]
```

- Using Jython:

```
print AdminConfig.create('EventGroupProfile',
 eventGroupProfileId, properties)
```

Example output:

```
(cells/mycell/nodes/mynode/servers/myserver|resources-cei.xml#EventGroupProfile_1)
```

6. Save the configuration changes.

7. In a network deployment environment only, synchronize the node.

8. Restart the server.

Creating an event filter with scripting

You can create an event filter using a Jacl or Jython script.

Before you begin

Before starting this task, the wsadmin tool must be running. See the WebSphere Application Server documentation for more information.

About this task

Perform the following steps to configure a new event filter.

Procedure

1. Identify the Common Event Infrastructure provider ID.

   - Using Jacl:

     ```
     set providerid [$AdminConfig getid \
     /Cell:mycell/Node:mynode/Server:myserver/ \
     EventInfrastructureProvider:/]
     ```

   - Using Jython:

     ```
     providerid =
     AdminConfig.getid('/Cell:mycell/Node:mynode/Server:myserver/EventInfrastructureProvider:/')
     print providerid
     ```

     Example output:

     ```
     EventInfrastructureProvider(cells/mycell/nodes/mynode/servers/myserver|resources-cei.xml#
     EventInfrastructureProvider_1)
     ```

2. Set the required attributes.

   - Using Jacl:

     ```
     set Name [list name "EventFilterName"]
     set JndiName [list jndiName "Put JNDI name for new event filter here"]
     set Description [list description "Set description of event filter here"]
     set Category [list category "Set category for event filter here"]

     # escape all '[' characters in the configuration string
     # (for example, "CommonBaseEvent\[@severity=50\]"
     set filterConfigurationString [list filterConfigurationString \
       "Set filter configuration string here"]
     ```

   - Using Jython:

     ```
     Name = ['name', 'EventFilterName']
     JndiName = ['jndiName',  'Put JNDI name for new event filter here']
     Description = ['description', 'Set description of event filter here']
     Category = ['category', 'Set category for event filter here']
     FilterConfigurationString = ['filterConfigurationString', \
       'Set filter configuration string here']
     ```

3. Set the properties for the new event filter.

   - Using Jacl:

     ```
     set properties [list $name $jndiName $description $category \
     $filterConfigurationString]
     ```

   - Using Jython:

     ```
     properties = [Name,JndiName,Description,Category,FilterConfigurationString]
     print properties
     ```

     Example output:

     ```
     [['name', 'EventFilterName'], ['jndiName', 'Put JNDI name for new
     event filter here'], ['description', 'Set description of event
     filter here'], ['category', 'Set category for event filter
     here'], ['filterConfigurationString', 'Set filter configuration
     string here']]
     ```

4. Create the event filter.

   - Using Jacl:

     ```
     set filterProf [$AdminConfig create FilterFactoryProfile \
     $providerid $properties]
     ```

   - Using Jython:

     ```
     print AdminConfig.create('FilterFactoryProfile', providerid, properties)
     ```

     Example output:

     ```
     EventFilterName(cells/mycell/nodes/mynode/servers/myserver|resources-cei.xml#FilterFactoryProfile_1)
     ```

5. Save the configuration changes.

6. In a network deployment environment only, synchronize the node.

7. Restart the server.

Logging and tracing Common Event Infrastructure components

You can enable logging and tracing in order to debug problems with an application using Common Event Infrastructure.

About this task

Common Event Infrastructure components use the JSR47 Java logging framework, which is available in Common Event Infrastructure server and client environments. For more information about using the logging framework, refer to the WebSphere Application Server Network Deployment, version 6.1 troubleshooting documentation.

The following table indicates the logger names used by the Common Event Infrastructure components.

Table 1. Logger names

Component	Logger name
Root logger name	com.ibm.events
Event catalog	com.ibm.events.catalog
Event service subcomponents	com.ibm.events.access com.ibm.events.bus com.ibm.events.distribution com.ibm.events.server
Default data store plug-in	com.ibm.events.datastore
Event emitter	com.ibm.events.emitter
Notification helper	com.ibm.events.notification
Configuration	com.ibm.events.configuration com.ibm.events.admintask
Database configuration	com.ibm.events.install.db
Migration	com.ibm.events.migration
Miscellaneous utilities	com.ibm.events.util

Maintaining the event database

If you are using an external event database, you should periodically run the provided scripts for database maintenance.

Updating DB2 event database statistics

To enable a DB2 database to optimize queries and find free space, you can update the database statistics by running the **runstats** script.

About this task

Updating DB2 database statistics is recommended on a regular basis, and especially under any of the following circumstances:

- Events have been deleted from the database, using either the event deletion interfaces of the event service or the rapid event purge utility of the default data store plug-in

- A large number of events have been inserted into the database
- Tables have been reorganized using the **reorg** script
- Indexes have been added or removed from a table

The **runstats** script is located in the *profile_root*/event/*node_name*/*server_name* or *cluster_name*/dbscripts/db2 directory.

Procedure

To update the database statistics, run runstats command:
The parameters are as follows:

db_user
> The database user ID to use. This parameter is required.

db_password
> The database password. This parameter is optional; if you do not specify the password on the command line, the DB2 database prompts you for it.

Reorganizing DB2 event database tables

After events are purged or deleted from a DB2 event database , you can reorganize the database tables using the **reorg** script.

About this task

The **reorg** script is located in the *profile_root*/event/*node_name*/*server_name* or *cluster_name*/dbscripts/db2 directory.

Procedure

To reorganize the event database tables, run the reorg command:
The parameters are as follows:

db_alias
> The database alias. The event database must be catalogued on the DB2 client; if you are running the script on the DB2 server, the database is already catalogued.

db_user
> The database user ID to use. This parameter is required.

db_password
> The database password. This parameter is optional; if you do not specify the password on the command line, the DB2 database prompts you for it.

What to do next

After running the **reorg** script, you should update the database statistics using the **runstats** script. For more information, see "Updating DB2 event database statistics" on page 29.

Purging events from the event database

You can run the provided scripts to purge large numbers of events from the event database.

About this task

The default data store plug-in provides a set of utilities you can run periodically to purge large numbers of old events from the event database. These utilities are different from the eventpurge event service command, which deletes events matching specified criteria.

The database purge capability uses the concept of *buckets*. A bucket is a set of tables used to store events in the event database. The default data store plug-in uses two buckets:

- The active bucket is the bucket containing the most recent events; new events are stored in the active bucket. The active bucket cannot be purged using the database purge utility.
- The inactive bucket contains older events. Events stored in the inactive bucket can be queried, deleted, or modified, but typically no new events are stored in the inactive bucket. The inactive bucket can be purged by the database purge utility.

Each event is stored in only one bucket. From the perspective of an event consumer, there is no distinction between the active and inactive buckets; a consumer can query, modify, or delete a specific event without knowing which bucket the event is stored in. The advantage of this approach is that the inactive bucket can be purged using database-specific interfaces without affecting the active bucket; typical event traffic can continue even while the purge operation is taking place.

After the inactive bucket is purged, you can then swap the buckets so that the active bucket becomes inactive and the inactive bucket becomes active. You can swap the buckets only when the inactive bucket is empty.

Although new events are stored only in the active bucket, under some circumstances events might be stored in the inactive bucket immediately after the buckets are swapped. The data store plug-in checks periodically to determine which bucket is currently marked as active, but until the next check takes place, some events might continue to be stored in the inactive bucket. Similarly, events sent as part of a batch are all stored in the same bucket, even if that bucket becomes inactive while the batch is still being processed.

If you want to use the fast purge capability, it is your responsibility to determine how frequently to swap buckets or purge the inactive bucket, depending upon event traffic, storage space, archival requirements, or other considerations.

Viewing or changing the event database active bucket status

The active bucket status indicates which bucket is currently available and which is currently not available.

Procedure

To view or change the active bucket status, use the eventbucket command:

```
eventbucket [-status] [-change]
```

This command has the following options:

-status
Use this option to see information about the current bucket configuration,

including the active bucket setting and the bucket check interval (the frequency with which the data store plug-in checks to determine which bucket is active).

-change
Use this option to swap the active and inactive buckets. The inactive bucket must be empty before you can use this option.

Purging the inactive bucket for a DB2 event database

The database purge utility for a DB2 event database is implemented using the DB2 load utility.

About this task

To purge the inactive bucket:

Procedure

1. Run the eventbucket command to identify the inactive bucket (bucket 0 or bucket 1).
2. Upload the appropriate utility control file. These files are generated during database configuration and are located in the *profile_path*/event/dbscripts/ db2zos directory. Upload one of the following files:
   - If bucket 0 is inactive, upload the fastpurge00.ctl file.
   - If bucket 1 is inactive, upload the fastpurge01.ctl file.

   Upload format: The control file must be uploaded with a fixed record format and a logical record length of 80.
3. Go to the ISPF DB2I Primary Option Menu and select the **Utilities** option.
4. Specify the following information:

Field	Value
Function	EDITJCL
Utility	LOAD
Statement Data Set	The name of the data set containing the uploaded control file
LISTDEF	NO
Template	NO

5. Press Enter to continue to the next panel.
6. In the recdsn entry field, type the name of the data set containing the uploaded control file.
7. Press Enter. The JCL script to purge the inactive bucket is generated.
8. Press Enter to clear the output messages.
9. Edit the generated JCL script as needed.
10. Submit the JCL script.

Results

The generated JCL script can be reused. After you create the scripts for purging both buckets (0 and 1), you do not need to repeat this entire procedure.

Changing the event database bucket check interval

The bucket check interval specifies how frequently the data store plug-in checks to determine which bucket is active. This value is specified as a custom property in the data store settings.

About this task

The default bucket check interval is 5 minutes (300 seconds). A shorter interval reduces the likelihood that events will be stored in the inactive bucket after swapping, but might decrease performance.

To change the bucket check interval:

Procedure

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service* > **Event data store**.
2. Modify the value of the BucketCheckInterval property to specify the bucket check interval in seconds.

Results

The change takes effect the next time the server is started.

Securing access to Common Event Infrastructure functions

You can use WebSphere method-level declarative security to secure access to Common Event Infrastructure functions.

Common Event Infrastructure defines six security roles, each one associated with a related group of functions. These security roles control access to both programming interfaces and commands.

The following table describes the security roles and the types of users associated with each role.

Table 2. Security roles and user types

Security role	User types
eventCreator	Event sources that need to submit events to an emitter using synchronous EJB calls. This role provides access to the following interfaces: • Emitter.sendEvent() • Emitter.sendEvents() • **eventemit** command The eventCreator role restricts access to event submission only if the emitter is configured to use synchronous EJB calls for event transmission. If the emitter uses asynchronous JMS messaging for event transmission, you must use JMS security to restrict access to the destination used to submit events.
eventUpdater	Event consumers that need to update events stored in the event database. This role provides access to the following interfaces: • EventAccess.updateEvents() • EventAccess.eventExists() • EventAccess.queryEventByGlobalInstanceId() • EventAccess.queryEventsByAssociation() • EventAccess.queryEventsByEventGroup() • **eventquery** command
eventConsumer	Event consumers that need to query events stored in the event database. This role provides access to the following interfaces: • EventAccess.eventExists() • EventAccess.queryEventByGlobalInstanceId() • EventAccess.queryEventsByAssociation() • EventAccess.queryEventsByEventGroup() • **eventquery** command

Table 2. Security roles and user types (continued)

Security role	User types
eventAdministrator	Event consumers that need to query, update, and delete events stored in the event database. This role provides access to the following interfaces: • EventAccess.purgeEvents() • EventAccess.eventExists() • EventAccess.queryEventByGlobalInstanceId() • EventAccess.queryEventsByAssociation() • EventAccess.queryEventsByEventGroup() • EventAccess.updateEvents() • Emitter.sendEvent() • Emitter.sendEvents() • **eventquery** command • **eventpurge** command • **eventemit** command • **eventbucket** command
catalogReader	Event catalog applications that need to retrieve event definitions from the event catalog. This role provides access to the following interfaces: • EventCatalog.getAncestors() • EventCatalog.getChildren() • EventCatalog.getDescendants() • EventCatalog.getEventDefinition() • EventCatalog.getEventDefinitions() • EventCatalog.getEventExtensionNamesForSourceCategory() • EventCatalog.getEventExtensionToSourceCategoryBindings() • EventCatalog.getParent() • EventCatalog.getRoot() • EventCatalog.getSourceCategoriesForEventExtension() • **eventcatalog** command (-listdefinitions option) • **eventcatalog** command (-listcategories option) • **eventcatalog** command (-exportdefinitions option)
catalogAdministrator	Event catalog applications that need to create, update, delete, or retrieve event definitions in the event catalog. This role provides access to all methods of the EventCatalog interface and all functions of the **eventcatalog** command. Because changes to the event catalog can result in generation of events, this role also provides access to event submission interfaces.

The event service message-driven bean runs using the server user identity. If you are using asynchronous JMS transmission to submit events to the event service, and you have enabled method-based security, you must map this user identity to the eventCreator role.

Security: If your event source is running with Java security enabled, and you want to generate your own globally unique identifiers (GUIDs), you must modify your policy file to enable correct processing. Add the following entries:

```
permission java.io.FilePermission "${java.io.tmpdir}${/}guid.lock",
  "read, write, delete";
permission java.net.SocketPermission "*", "resolve";
```

Troubleshooting Common Event Infrastructure

These topics provide troubleshooting information for the event service based on the task or activity you were doing when you encountered the problem.

Problems during startup

Troubleshoot problems with starting up your Common Event Infrastructure server.

Event service does not start (message CEIDS0058E)

The event service does not start and outputs message CEIDS0058E to the WebSphere log file.

Cause

The event service uses SQL statements qualified with the user name. This error indicates that the user name used by the event service to connect to the event database is not the same as the user ID that was used to create the database.

Remedy

The user ID used to connect to the event database must be the same one used to create the event database. To correct this problem:

1.
   - For a single server, select **Servers > Application servers > *server_name***.
   - For a cluster, select **Servers > Clusters > *cluster_name***.
2. From the **Configuration** tab, select **Business Integration > Common Event Infrastructure > Common Event Infrastructure Server**.
3. Change the specified user ID and password to match those used to create the database.
4. Save the configuration changes.
5. Restart the server.

Problems when sending events

Troubleshoot problems with sending events on your Common Event Infrastructure server.

Error when sending event (message CEIDS0060E)

My event source encounters an error when trying to send an event, and message CEIDS0060E appears in the WebSphere log file.

Cause

The event service uses metadata stored in the event database to map Common Base Event elements and attributes to database tables and columns. This information is read from the database the first time an application attempts to use the event service after startup.

The metadata tables are populated when the event database is created. This error occurs if the tables do not contain the required metadata at run time.

Remedy

To correct this problem, you need to re-create the required metadata. When the event database is created, the database configuration administrative command also generates a database script that can be used to repopulate the metadata at a later time. The name of this script depends on the database type:

Database type	Script name
Derby	ins_metadata.derby
DB2	ins_metadata.db2
Informix	ins_metadata.sql
Oracle	ins_metadata.ora
SQL Server	ins_metadata.mssql
DB2 UDB for iSeries	ins_metadata.db2
DB2 for z/OS	ins_metatdata.ddl

By default, the script, ins_metadata.ddl, is created in the *profile_root*/dbscripts/ *CEI_Databasename*/ddl directory. You can run this script at any time using the DB2 SQL processor.

After repopulating the metadata, restart the server.

Error when sending event (ServiceUnavailableException)

My event source application encounters an error when trying to send an event to the event server. The log file indicates a ServiceUnavailableException with the message "A communication failure occurred while attempting to obtain an initial context with the provider URL."

Cause

This problem indicates that the event source application cannot connect to the event server. This might be caused by either of the following conditions:

- The event server is not running.
- The event source application is not configured to use the correct JNDI provider URL.

Remedy

To correct this problem, follow these steps:

1. To check the status of the event server, go to the */WebSphere/V7R0/ DeploymentManager/default/bin/*bin directory and run the **serverStatus** command:

   serverStatus *servername*

2. If the event server is not running, use the **startServer** command to start it:

   startServer *servername*

3. Check the host name and Remote Method Invocation (RMI) port for the server containing the application that can not connect to the event server. Confirm that the same values are specified in the JNDI URL configured for the event

source application. If the CEI server is located on another server, then the JNDI needs to be resolved with that remote deployment target.

Error when sending event (NameNotFoundException)

My event source application encounters an error when trying to send an event to the event service. The log file indicates a NameNotFoundException with a message like "First component in name events/configuration/emitter/Default not found."

Cause

This problem indicates that the event service is not available and might be caused by either of the following conditions:

- The event service has not been deployed.
- The event service is disabled.

Remedy

To deploy the event service:

1. Start the wsadmin tool.
2. Use the AdminTask object to run the deployEventService administrative command.
3. Restart the server.

To enable the event service using the wsadmin tool:

1. Start the wsadmin tool.
2. Use the AdminTask object to run the enableEventService administrative command.
3. Restart the server.

To enable the event service using the administrative console:

1. Click **Applications** → **Application Types** → **WebSphere enterprise applications** → *server* → **Container Services** → **Common Event Infrastructure Service**.
2. Select the Enable service at server startup property.
3. Click **OK** to save your changes.
4. Restart the server.

Error when sending event (message CEIEM0025E)

My event source application encounters an error when trying to send an event to the event server. The log file indicates a DuplicateGlobalInstanceIdException.

Cause

This problem indicates that the emitter submitted the event, but the event service rejected it because another event already exists with the same global instance identifier. Each event must have a unique global instance identifier, specified by the globalInstanceId property.

Remedy

To correct this problem, do one of the following:

- Make sure your event source application generates a unique global instance identifier for each event.

- Leave the globalInstanceId property of your submitted events empty. The emitter then automatically generates a unique identifier for each event.

Error when sending event (message CEIEM0034E)

My event source encounters an error when trying to send an event to the event service. The log file indicates an EmitterException with the message "The JNDI lookup of a JMS queue failed because the JNDI name defined in the emitter profile is not bound in the JNDI."

Cause

This problem indicates that the JMS transmission configuration being used by the emitter specifies one or more JMS resources that are not defined in the JMS configuration.

Remedy

To correct this problem:

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event emitter factories** > *emitter_factory* > **JMS transmission settings**. Make sure you are viewing the JMS transmission for the emitter factory used by your event source application.
2. Check the values specified for the **Queue JNDI name** and **Queue connection factory JNDI name** properties. Make sure the specified JNDI names exist in the JNDI namespace and are valid JMS objects. If necessary, modify these properties or create the required JMS resources.

Event is not valid (message CEIEM0027E)

My event source is trying to send an event, but the emitter does not submit it to the event service and outputs message CEIEM0027E to the log file ("The emitter did not send the event to the event server because the Common Base Event is not valid").

Cause

This message indicates that one or more of the event properties contain data that does not conform to the Common Base Event specification. There are many ways in which event data might not be valid, including the following:

- The global instance identifier must be at least 32 characters but no more than 64 characters in length.
- The severity must be within the range of 0 - 70.

Remedy

To correct this problem:

1. Check the detailed exception message in the log file to determine which event property is not valid. For example, this messages indicates that the length of the global instance identifier (ABC) is not valid:

```
Exception: org.eclipse.hyades.logging.events.cbe.ValidationException
  : IWAT0206E The length of the identifier in the specified Common
    Base Event property is outside the valid range of 32 to 64
    characters.
  Property: CommonBaseEvent.globalInstanceId
  Value: ABC
```

2. Correct the event content at the source so it conforms to the Common Base Event specification.
3. Resubmit the event.

Synchronization mode not supported (message CEIEM0015E)

My event source is trying to send an event, but the emitter does not submit it to the event service and outputs message CEIEM0015E to the log file ("The emitter does not support the specified synchronization mode").

Cause

This problem indicates that the parameters passed by the event source when sending the event specify a synchronization mode that is not supported by the emitter. This can be caused be either of the following conditions:

- The event source is specifying a synchronization mode that is not valid. This is indicated by an IllegalArgumentException with the message "Synchronization mode *mode* is not valid."
- The event source is specifying a synchronization mode that the emitter is not configured to support. This is indicated by a SynchronizationModeNotSupportedException with the message "The emitter does not support the specified synchronization mode: *mode*."

Remedy

If the exception message indicates that your event source is specifying a synchronization mode that is not valid (IllegalArgumentException), check the method call that is trying to send the event. Make sure the method parameters specify one of the valid synchronization modes:

- SynchronizationMode.ASYNCHRONOUS
- SynchronizationMode.SYNCHRONOUS
- SynchronizationMode.DEFAULT

These constants are defined by the com.ibm.events.emitter.SynchronizationMode interface.

If the exception message indicates that the specified synchronization mode is not supported by the emitter (SynchronizationModeNotSupportedException), check the emitter factory configuration:

1. In the administrative console, click **Service Integration** > **Common Event Infrastructure** > **Event Emitter Factories** > *emitter_factory*. Make sure you are viewing the emitter factory used by the event source application.
2. Check the emitter factory settings to see which synchronization modes are supported:
   - If the **Support Event Service transmission** property is selected, synchronous mode is supported.
   - If the **Support JMS transmission** property is selected, asynchronous mode is supported.

   Querying transaction modes: An event source can programmatically query the supported transaction modes for a particular emitter by using the isSynchronizationModeSupported() method. Refer to the Javadoc API documentation for more information.

3. If the emitter does not support the synchronization mode you are trying to use, you must either change the emitter factory configuration or modify your event source to use a supported synchronization mode.

Transaction mode not supported (message CEIEM0016E)

My event source is trying to send an event, but the emitter does not submit it to the event service and outputs message CEIEM0016E to the log file ("The emitter does not support the specified transaction mode").

Cause

This problem indicates that the parameters passed by the event source when sending the event specify a transaction mode that is not supported by the emitter. This can be caused be either of the following conditions:

- The event source is specifying a transaction mode that is not valid.
- The event source is specifying a synchronization mode that is not supported by the emitter environment. Transactions are supported only in a Java EE container.

Remedy

To correct this problem, check the method call that is trying to send the event and make sure the method parameters specify the correct transaction mode:

- If the emitter is running in a Java EE container, make sure the method parameters specify one of the valid transaction modes:
  - TransactionMode.NEW
  - TransactionMode.SAME
  - TransactionMode.DEFAULT

  These constants are defined by the com.ibm.events.emitter.TransactionMode interface.
- If the emitter is not running in a Java EE container, make sure the method parameters specify TransactionMode.DEFAULT.

Problems when receiving or querying events

Troubleshoot problems involving receiving and querying events with your Common Event Infrastructure server.

Error when querying events (message CEIDS0060E)

My event consumer encounters an error when trying to query events from the event service, and message CEIDS0060E appears in the WebSphere log file.

Cause

The event service uses metadata stored in the event database to map Common Base Event elements and attributes to database tables and columns. This information is read from the database the first time an application attempts to use the event service after startup.

The metadata tables are populated when the event database is created. This error occurs if the tables do not contain the required metadata at run time.

Remedy

To correct this problem, you need to re-create the required metadata. When the event database is created, the database configuration administrative command also generates a database script that can be used to repopulate the metadata at a later time. The name of this script depends on the database type:

Database type	Script name
Derby	ins_metadata.derby
DB2	ins_metadata.db2
Informix	ins_metadata.sql
Oracle	ins_metadata.ora
SQL Server	ins_metadata.mssql
DB2 UDB for iSeries	ins_metadata.db2
DB2 for z/OS	ins_metatdata.ddl

By default, the script, ins_metadata.ddl, is created in the *profile_root*/dbscripts/ *CEI_Databasename*/ddl directory. You can run this script at any time using the DB2 SQL processor.

After repopulating the metadata, restart the server.

Events not being stored in the persistent data store

My event source application is successfully submitting events to the emitter, but when an event source queries the events, they are not in the persistent data store.

Cause

This problem indicates that the emitter is not sending events to the event service, or that the event service is not storing the events to the persistent data store. This can be caused be any of the following conditions:

- The persistent data store not enabled for the event service.
- The events do not belong to an event group that is configured for event persistence.
- The events are being filtered out by the emitter.

Remedy

To verify that the persistent data store is enabled for the event service:

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service*.
2. Make sure the **Enable event data store** check box is selected.
3. Click **OK** to save any changes.

To verify that the event group is configured for event persistence:

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service* > **Event groups** > *event_group*.
2. Make sure the **Persist events to event data store** check box is selected.
3. Click **OK** to save any changes.

Multiple event groups: An event might belong to multiple event groups. If any applicable event group is configured for persistence, and the data store is enabled, the event is stored in the data store.

To check the filter settings:

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event emitter factories** > *emitter_factory* > **Event filter**. (Make sure you are viewing the settings for the emitter factory your event source application is using.)

2. Check to see whether the filter configuration string excludes the events you are trying to send to consumers. If so, you can either modify the filter configuration string or modify the event data so the events are not filtered out.

3. Click **OK** to save any changes.

Events not being received by consumers (no error message)

My event source application is successfully submitting events to the emitter, but the events are not received by consumers using the JMS interface.

Cause

This problem can be caused be any of the following conditions:

- Event distribution is not enabled for the event service.
- The events are being filtered out by the emitter.
- The events are being filtered out by the notification helper.
- The event consumer is not specifying the correct event group.
- The JMS connection is not started.

Remedy

The remedy for this problem depends upon the underlying cause.

- **To verify that event distribution is enabled for the event service:**
  1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service*.
  2. If the **Enable event distribution** property is not selected, select the check box.
  3. Click **OK** to save any changes.
- **To check the event filter settings for the emitter:**
  1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event emitter factories** > *emitter_factory* > **Event filter**. (Make sure you are viewing the settings for the emitter factory your event source application is using.)
  2. Check to see whether the filter configuration string excludes the events you are trying to send to consumers. If so, you can either modify the filter configuration string or modify the event data so the events are not filtered out.
  3. Click **OK** to save any changes.
- **To check the event filter settings for the notification helper:**
  1. Check your event consumer application to see if an event selector is specified for the notification helper using the NotificationHelper.setEventSelector method.

2. If an event selector is specified, make sure it does not exclude the event you are trying to receive. (A null event selector passes all events.)

- **To check the event group specified by the event consumer:**
  1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service* > **Event groups**. The table shows a list of all event groups defined for the event service.
  2. Select the event group your event consumer subscribes to.
  3. Find the **Event selector string** property.
  4. Make sure the specified event selector matches the content of the event you are trying to receive. If it does not match, you might want to make one of the following changes:
     - Modify the event selector so the event is included in the event group.
     - Modify the event data so the event matches the event group.
     - Modify your event consumer to subscribe to a different event group that includes the event.

- **To start the JMS connection:**

  In your event consumer, use the QueueConnection.start() method or the TopicConnection.start() method before attempting to receive events.

Events not being received by consumers (NameNotFoundException)

My event source application is successfully submitting events to the emitter, but the events are not published to consumers using the JMS interface, and the log file indicates a NameNotFoundException.

Cause

This problem indicates that the event group configuration specifies one or more JMS resources that do not exist.

Remedy

To correct this problem:

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service* > **Event groups** > *event_group*.

   Multiple event groups: An event might belong to more than one event group.

2. Check the values of the **Topic JNDI name** and **Topic connection factory JNDI name** properties. Verify that the specified JMS resources exist. If necessary, use the configuration interface of your JMS provider to create the necessary resources.

Event group with extended data elements contains no events

I have defined an event group that specifies extended data element predicates, but queries on the event group do not return the expected events.

Cause

The event data might be valid XML but not conform with the Common Base Event specification. This can cause unexpected results without any error messages.

Consider an event with the following content:

```
<?xml version="1.0" encoding="ASCII"?>
<!-- Event that will match the XPath expression CommonBaseEvent
     [@globalInstanceId] -->
<CommonBaseEvent
      xmlns:xsi="http://www.w3.org/TR/xmlschema-1/"
      xmlns:="http://www.ibm.com/AC/commonbaseevent1_0_1"
      version="1.0.1"
      creationTime="2005-10-17T12:00:01Z"
      severity="10"
      priority="60"
   >
   <situation categoryName="RequestSituation">
       <situationType xsi:type="RequestSituation"
           reasoningScope="INTERNAL"
           successDisposition="Suceeded"
           situationQualifier="TEST"
       />
   </situation>
   <sourceComponentId
       component="component"
       subComponent="subcomponent"
       componentIdType="componentIdType"
       location="localhost"
       locationType="Hostname"
       componentType="sourceComponentType"
   />
   <extendedDataElement name="color" type="string">
       <values>red</values>
   </extendedDataElement>
</CommonBaseEvent>
```

This event contains a single extended data element with a single child element.

Now consider an event group definition configured with the following XPath event selector string:

```
CommonBaseEvent[extendedDataElements[@name='color' and @type='string'
      and @values='red']]
```

This event selector fails to match the event because the XML definition of the event contains a misspelling. In the event data, the `extendedDataElements` element is misspelled as `extendedDataElement`. Because this is well-formed XML, it does not cause an error; instead, it is treated as an any element, which is not searchable.

Remedy

Make sure the XML data for submitted events conforms to the Common Base Event specification.

Error when querying an event group (message CEIES0048E)

My event consumer application encounters an error when trying to query events from an event group. The log file indicates an EventGroupNotDefinedException and shows message CEIES0048E ("The event group is not defined in the event group list that the event server instance is using.")

Cause

This problem indicates that the event consumer application performed a query using the EventAccess bean, but the consumer specified an event group name that does not correspond to any existing event group.

Remedy

To correct this problem:

1. In the administrative console, click **Service integration** > **Common Event Infrastructure** > **Event service** > **Event services** > *event_service* > **Event groups**. The table shows a list of all event groups defined for the event service.

2. Make sure the event source specifies a defined event group name in the parameters of the query method call.

Miscellaneous problems

Troubleshoot miscellaneous problems with your Common Event Infrastructure server.

Event catalog pattern query fails on a Windows system

I am trying to do a pattern query for event definitions on a Windows system using the **eventcatalog** *command. For example: eventcatalog -listdefinitions -name EVENT% -pattern. I don't get the expected results.*

Cause

The percent character (%) is a reserved character in the Windows command-line interface and is not passed properly to the **eventcatalog** command.

Remedy

On Windows systems, you must escape the percent character character by typing %%:

```
eventcatalog -listdefinitions -name EVENT%% -pattern
```

Common Event Infrastructure commands

Commands used by Common Event Infrastructure for creating and removing event databases and services

Commands for creating event data sources on specific databases:

Generic Common Event Infrastructure commands used by the server to handle the event service:

Commands for removing event data sources on specific databases:

configEventServiceDB2DB command

Use the configEventServiceDB2DB command to configure the Common Event Infrastructure using a DB2 database.

Purpose

The configEventServiceDB2DB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to create the event service database and data sources for DB2 on a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Parameters

- createDB

The command generates the DDL database scripts and creates the database when this parameter is set to `true`. The command only generates the DDL database scripts when this parameter is set to `false`. To create the database, the current server must be already configured to run the database commands. The default value is `false` if not specified.

- overrideDataSource

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/*dbtype* if this parameter is not specified.

- nodeName

The name of the node that contains the server where the event service data source should be created. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be created. If this parameters is specified without the **nodeName** parameter, the command

will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- jdbcClassPath

The path to the JDBC driver. Specify only the path to the driver file; do not include the file name in the path. This parameter is required.

- dbNodeName

The DB2 node name (this must be 8 characters or less). This node must be already catalogued and configured to communicate with the DB2 server. This parameter must be set if the current server is configured as a DB2 client and the parameter **createDB** is set to `true`.

- dbHostName

The host name of the server where the database server is installed. This parameter is required.

- dbPort

DB2 instance port. The default value is 50 000 if not specified.

- dbName

The database name to be created. The default value is `event` if not specified.

- dbUser

DB2 user ID that has privileges to create and drop the databases. The default value is `db2inst1` if not specified.

- dbPassword

DB2 password. This parameter is required.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/db2 if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDB2DB {-createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -jdbcClassPath c:\sqllib\java
 -dbUser db2inst1
 -dbPassword dbpassword
 -dbHostName host_name
 -dbPort 50000 }
```

- Using Jython string:

```
AdminTask.configEventServiceDB2DB('[-createDB true
 -overrideDataSource true
 -nodeName nodename
```

```
-serverName servername
-jdbcClassPath c:\sqllib\java
-dbUser db2inst1
-dbPassword dbpassword
-dbHostName host_name
-dbPort 50000 ]')
```

- Using Jython list:

```
AdminTask.configEventServiceDB2DB(['-createDB', 'true',
'-overrideDataSource', 'true',
'-nodeName', 'nodename',
'-serverName', 'servername',
'-jdbcClassPath', 'c:\sqllib\java',
'-dbUser', 'db2inst1',
'-dbPassword', 'dbpassword ',
'-dbHostName', 'host_name',
'-dbPort', '50000 '])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDB2DB -interactive
```

- Using Jython string:

  Note: � **Vista** ▮ Windows 7 ▮ The product uses a Jython version that does not support Microsoft® Windows 2003, Windows Vista, or Windows 7 operating systems.

```
AdminTask.configEventServiceDB2DB('[-interactive]')
```

- Using Jython list:

```
AdminTask.configEventServiceDB2DB(['-interactive'])
```

configEventServiceDB2iSeriesDB command

Use the configEventServiceDB2iSeriesDB command to configure the Common Event Infrastructure using a DB2 for iSeries database.

Purpose

The configEventServiceDB2iSeriesDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use it to generate the DDL database scripts for use on remote database servers, create the event service database for DB2 iSeries on a local server, and creates data sources on a server or cluster. For more information about the AdminTask object, refer to the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: ▮ **Vista** ▮ Windows 7 ▮ The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- createDB

The command generates the DDL database scripts and creates the database when this parameter is set to true. The command only generates the DDL database scripts when this parameter is set to false. To create the database, the current server must be already configured to run the database commands. The default value is false if not specified.

- overrideDataSource

When this parameter is set to `true`, the command removes any existing event service data source at the specified scope before creating a new one. When this parameter is set to `false`, the command does not create an event service data source at the specified scope if another event service data source is found at the same scope. The default value is `false` if not specified.

- nodeName

The name of the node that contains the server where the event service data source should be created. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be created. If this parameters is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- toolboxJdbcClassPath

The path to the IBM Toolbox for Java DB2 JDBC driver. Specify only the path to the driver file; do not include the file name. You must specify either this parameter or the **jdbcClassPath** parameter.

Note: You must specify **toolboxJdbcClassPath** path if you are creating the database on an iSeries server.

- nativeJdbcClassPath

The path to the DB2 for iSeries native JDBC driver. Specify only the path to the driver file; do not include the file name in the path. You must specify either this parameter or the **toolboxJdbcClassPath** parameter.

Note: You must specify the **nativeJdbcClassPath** if you are using an iSeries server to create the database on a non-iSeries server. Specify the DB2 Universal Driver if you are creating the database on DB2 Universal Database™ for multiplatforms or DB2 Universal Database for z/OS. Specify a type 4 driver if you are creating the database on Informix, Oracle, or Microsoft SQL Server.

- dbHostName

The host name of the server where the DB2 for iSeries database server is installed. This parameter is required if you are using the IBM Toolbox for Java DB2 JDBC driver.

- dbName

The DB2 for iSeries database name. The default value is `*LOCAL` if not specified.

- collection

DB2 for iSeries library SQL collection. The maximum length for the collection name is 10 characters. The default value is an empty string if not specified.

- dbUser

DB2 user ID that has privileges to create and drop the databases. This parameter is required.

- dbPassword

Password of the database user ID. This parameter is required.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/ db2iseries if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDB2iSeriesDB {createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -dbUser db2user
 -dbPassword dbpassword
 -nativeJdbcClassPath /myDB2ClassPath
 -collection event}
```

- Using Jython string:

```
AdminTask.configEventServiceDB2iSeriesDB('[-createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -nativeJdbcClassPath /myDB2ClassPath
 -collection event]')
```

- Using Jython list:

```
AdminTask.configEventServiceDB2iSeriesDB(['-createDB', 'true',
 '-overrideDataSource', 'true',
 '-nodeName', 'nodename',
 '-serverName', 'servername',
 '-nativeJdbcClassPath', '/myDB2ClassPath',
 '-collection', 'event'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDB2iSeriesDB -interactive
```

- Using Jython string:

```
AdminTask.configEventServiceDB2iSeriesDB('[-interactive]')
```

- Using Jython list:

```
AdminTask.configEventServiceDB2iSeriesDB(['-interactive'])
```

configEventServiceDB2ZOSDB command

Use the configEventServiceDB2ZOSDB command to configure the Common Event Infrastructure using a DB2 for z/OS database.

Purpose

The configEventServiceDB2ZOSDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to create the event service database and data sources for DB2 z/OS on a server or

cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- createDB

The command generates the DDL database scripts and creates the database when this parameter is set to true. The command only generates the DDL database scripts when this parameter is set to false. To create the database, the current server must be already configured to run the database commands. The default value is false if not specified.

- overrideDataSource

When this parameter is set to true, the command removes any existing event service data source at the specified scope before creating a new one. When this parameter is set to false, the command does not create an event service data source at the specified scope if another event service data source is found at the same scope. The default value is false if not specified.

- nodeName

The name of the node that contains the server where the event service data source should be created. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be created. If this parameters is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- jdbcClassPath

The path to the JDBC driver. Specify only the path to the driver file; do not include the file name in the path. This parameter is required.

- dbHostName

The host name of the server where the database is installed. This parameter is required.

- dbPort

DB2 for z/OS instance port. The default value is 5027 if not specified.

- dbName

The DB2 database name. On the DB2 client, this is the name of the catalogued database. On the native z/OS server, it is the name of the database subsystem. The default value is event if not specified.

- dbDiskSizeInMB

Specify the disk size in MB for the event service database. This value must be at least 10 MB. The default value is 100 MB if not specified.

- dbUser

DB2 user ID that has privileges to create and drop the databases. This parameter is required.

- dbPassword

Password of the database user ID. This parameter is required.

- storageGroup

The storage group for the event database and the event catalog database. The storage group must already be created and active.

- bufferPool4K

The name of the 4K buffer pool. This buffer pool must be active before the database DDL scripts can be run.

- bufferPool8K

The name of the 8K buffer pool. This buffer pool must be active before the database DDL scripts can be run.

- bufferPool16K

The name of the 16K buffer pool. This buffer pool must be active before the database DDL scripts can be run.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/db2zos if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDB2ZOSDB {-createDB true -overrideDataSource true
 -nodeName nodename -serverName servername -jdbcClassPath c:\sqllib\java
 -dbUser db2user -dbPassword dbpassword -dbHostName host_name -dbPort 5027
 -storageGroup sysdeflt -bufferPool4K BP9 -bufferPool8K BP8K9
 -bufferPool16K BP16K9}
```

- Using Jython string:

```
AdminTask.configEventServiceDB2ZOSDB('[-createDB true -overrideDataSource true
 -nodeName nodename -serverName servername -jdbcClassPath c:\sqllib\java
 -dbUser db2user -dbPassword dbpassword -dbHostName host_name -dbPort 5027
 -storageGroup sysdeflt -bufferPool4K BP9 -bufferPool8K BP8K9
 -bufferPool16K BP16K9]')
```

- Using Jython list:

```
AdminTask.configEventServiceDB2ZOSDB(['-createDB', 'true', '-overrideDataSource',
 'true', '-nodeName', 'nodename', '-serverName', 'servername', '-jdbcClassPath',
 'c:\sqllib\java', '-dbUser', 'db2user', '-dbPassword', 'dbpassword',
 '-dbHostName', 'host_name', '-dbPort', '5027', '-storageGroup', 'sysdeflt',
 '-bufferPool4K', 'BP9', '-bufferPool8K', 'BP8K9', '-bufferPool16K', 'BP16K9'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDB2ZOSDB -interactive
```

- Using Jython string:

```
AdminTask.configEventServiceDB2ZOSDB('[-interactive]')
```

- Using Jython list:

configEventServiceDerbyDB command

Use the configEventServiceDerbyDB command to configure the Common Event Infrastructure using a Derby database.

Purpose

The configEventServiceDerbyDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to create the event service database and data sources for Derby on a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- createDB

The command generates the DDL database scripts and creates the database when this parameter is set to true. The command only generates the DDL database scripts when this parameter is set to false. To create the database, the current server must be already configured to run the database commands. The default value is false if not specified.

- overrideDataSource

When this parameter is set to true, the command removes any existing event service data source at the specified scope before creating a new one. When this parameter is set to false, the command does not create an event service data source at the specified scope if another event service data source is found at the same scope. The default value is false if not specified.

- nodeName

The name of the node that contains the server where the event service data source should be created. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be created. If this parameters is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbHostName

The host name of the network Derby database. To create the Derby network data source, specify this parameter and the **dbPort** parameter. To create the Derby local data source, do not specify this parameter and the **dbPort** parameter.

- dbPort

The port number of the network Derby database. To create the Derby network data source, specify this parameter and the **dbHostName** parameter. To create the Derby local data source, do not specify this parameter and the **dbHostName** parameter.

- dbName

The database name to be created. The default value is `event` if not specified.

- dbUser

The user ID used by the data source for the Derby database authentication. This parameter is optional when the WebSphere domain security is disabled. If you specify this parameter, you also must specify the **dbPassword** parameter. This parameter is required when the WebSphere domain security is enabled.

- dbPassword

The password used by the data source for the Derby database authentication. This parameter is optional when the WebSphere domain security is disabled. If you specify this parameter, you also must specify the **dbUser** parameter. This parameter is required when the WebSphere domain security is enabled.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/derby if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDerbyDB {-createDB true -overrideDataSource true
 -nodeName nodename -serverName servername}
```

- Using Jython string:

```
AdminTask.configEventServiceDerbyDB( '[-createDB true -overrideDataSource true
 -nodeName nodename -serverName servername]')
```

- Using Jython list:

```
AdminTask.configEventServiceDerbyDB(['-createDB', 'true', '-overrideDataSource',
 'true', '-nodeName', 'nodename', '-serverName', 'servername'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceDerbyDB -interactive
```

- Using Jython string:

```
AdminTask.configEventServiceDerbyDB('[-interactive]')
```

- Using Jython list:

```
AdminTask.configEventServiceDerbyDB(['-interactive'])
```

configEventServiceInformixDB command

Use the configEventServiceInformixDB command to configure the Common Event Infrastructure using an Informix database.

Purpose

The configEventServiceInformixDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to create the event service database and data sources for Informix on a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista**   Windows 7   The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- **createDB**

    The command generates the DDL database scripts and creates the database when this parameter is set to `true`. The command only generates the DDL database scripts when this parameter is set to `false`. To create the database, the current server must be already configured to run the database commands. The default value is `false` if not specified.

- **overrideDataSource**

    Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/*dbtype* if this parameter is not specified.

- **nodeName**

    The name of the node that contains the server where the event service data source should be created. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- **serverName**

    The name of the server where the event service data source should be created. If this parameters is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- **clusterName**

    The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- **jdbcClassPath**

    The path to the JDBC driver. Specify only the path to the driver file; do not include the file name in the path. This parameter is required.

- **dbInformixDir**

    The directory where the Informix database is installed. This parameter must be specified when the parameter **createDB** is set to `true`. This parameter is required.

- **dbHostName**

    The host name of the server where the database is installed. This parameter is required.

- dbServerName

Informix server name (for example, `ol_servername`). This parameter is required.

- dbPort

Informix instance port. The default value is 1526 if not specified.

- dbName

The database name to be created. The default value is `event` if not specified.

- dbUser

The Informix database schema user ID that will own the event service database tables. The WebSphere data source uses this user ID to authenticate the Informix database connection. This parameter is required.

- dbPassword

The password of the schema user ID that owns the event service Informix tables. The WebSphere data source uses this password to authenticate the Informix database connection. This parameter is required.

- ceiInstancePrefix

The command uses the event service instance name to group the database files in a directory with unique names. The default value is `ceiinst1` if not specified.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/informix if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceInformixDB {-createDB true
-overrideDataSource true -nodeName nodename -serverName servername
-jdbcClassPath "c:\program files\ibm\informix\jdbc\lib"
-dbInformixDir "c:\program files\ibm\informix" -dbUser informix
-dbPassword dbpassword -dbHostName host_name -dbPort 1526
-dbServerName ol_server }
```

- Using Jython string:

```
AdminTask.configEventServiceInformixDB('[-createDB true
    -overrideDataSource true
    -nodeName nodename
    -serverName servername
    -jdbcClassPath "c:\program files\ibm\informix\jdbc\lib"
    -dbInformixDir "c:\program files\ibm\informix"
    -dbUser informix
    -dbPassword dbpassword
    -dbHostName host_name
    -dbPort 1526
    -dbServerName ol_server]')
```

- Using Jython list:

```
AdminTask.configEventServiceInformixDB(['-createDB', 'true',
  '-overrideDataSource', 'true', '-nodeName', 'nodename',
  '-serverName', 'servername', '-jdbcClassPath',
  'c:\program files\ibm\informix\jdbc\lib', '-dbInformixDir',
```

```
                    'c:\program files\ibm\informix', '-dbUser', 'informix ',
                    '-dbPassword', 'dbpassword', '-dbHostName', 'host_name',
                    '-dbPort', '1526', '-dbServerName', 'ol_server'])
```

Interactive mode example usage:

- Using Jacl:

  ```
  $AdminTask configEventServiceInformixDB -interactive
  ```

- Using Jython string:

  ```
  AdminTask.configEventServiceInformixDB('[-interactive]')
  ```

- Using Jython list:

  ```
  AdminTask.configEventServiceInformixDB(['-interactive'])
  ```

configEventServiceOracleDB command

Use the configEventServiceOraclesDB command to configure the Common Event
Infrastructure using an Oracle database.

Purpose

The configEventServiceOracleDB command is a Common Event Infrastructure
administrative command is available for the AdminTask object. Use this command
to create the event service tables and data sources for Oracle on a server or cluster.
The command does not create the database; the Oracle SID must already exist. For
more information about the AdminTask object, see the WebSphere Application
Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not
support Microsoft Windows 2003, Windows Vista, or Windows 7 operating
systems.

Parameters

- createDB

  The command generates the DDL database scripts and creates the database
  when this parameter is set to `true`. The command only generates the DDL
  database scripts when this parameter is set to `false`. To create the database, the
  current server must be already configured to run the database commands. The
  default value is `false` if not specified.

- overrideDataSource

  Optional database script output directory. When this parameter is specified, the
  command generates the event service database scripts in the specified
  directory. If the specified directory does not contain a full path, the command
  creates the specified directory in *profile_root*/bin. The default database script
  output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/*dbtype* if
  this parameter is not specified.

- nodeName

  The name of the node that contains the server where the event service data
  source should be created. If this parameter is specified, then the **serverName**
  parameter must be set. You must not specify this parameter if the **clusterName**
  parameter is specified.

- serverName

  The name of the server where the event service data source should be created.
  If this parameters is specified without the **nodeName** parameter, the command

will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- jdbcClassPath

The path to the JDBC driver. Specify only the path to the driver file; do not include the file name in the path. This parameter is required.

- oracleHome

The ORACLE_HOME directory. This parameter must be set when the parameter **createDB** is set to `true`.

- dbHostName

The host name of the server where the Oracle database is installed. The default value is `localhost` if not specified.

- dbPort

Oracle instance port. The default value is 1521 if not specified.

- dbName

The Oracle system identifier (SID). The SID must already exist and must be available for the event service command to create the tables and populate the tables with data. The default value is `orcl` if not specified.

- dbUser

The Oracle schema user ID that will own the event service Oracle tables. The user ID will be created during the database creation; the WebSphere data source uses this user ID to authenticate the Oracle database connection. The default value is `ceiuser` if not specified.

- dbPassword

The password of the schema user ID. The password will be created during the database creation; the WebSphere data source uses this password to authenticate the Oracle database connection. This parameter is required.

- sysUser

Oracle sys user ID. This must be a user that has SYSDBA privileges. The default value is `sys` if not specified.

- sysPassword

The password for the user specified by the **sysUser** parameter. The default value is an empty string if not specified.

- ceiInstancePrefix

The command uses the event service instance name to group the database files in a directory with unique names. The default value is `ceiinst1` if not specified.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/oracle if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceOracleDB {-createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -jdbcClassPath c:\oracle\ora92\jdbc\lib
 -oracleHome c:\oracle\ora92
 -dbUser ceiuser
 -dbPassword ceipassword
 -dbHostName host_name
 -dbPort 1521
 -sysUser sys
 -sysPassword syspassword}
```

- Using Jython string:

```
AdminTask.configEventServiceOracleDB( '[-createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -jdbcClassPath c:\oracle\ora92\jdbc\lib
 -oracleHome c:\oracle\ora92
 -dbUser ceiuser
 -dbPassword ceipassword
 -dbHostName host_name
 -dbPort 1521
 -sysUser sys
 -sysPassword syspassword]' )
```

- Using Jython list:

```
AdminTask.configEventServiceOracleDB(['-createDB', 'true',
 '-overrideDataSource', 'true',
 '-nodeName', 'nodename',
 '-serverName', 'servername',
 '-jdbcClassPath', 'c:\oracle\ora92\jdbc\lib',
 '-oracleHome', 'c:\oracle\ora92',
 '-dbUser', 'ceiuser',
 '-dbPassword', 'ceipassword',
 '-dbHostName', 'host_name',
 '-dbPort', '1521',
 '-sysUser', 'sys',
 '-sysPassword', 'syspassword'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceOracleDB -interactive
```

- Using Jython string:

```
AdminTask.configEventServiceOracleDB('[-interactive]')
```

- Using Jython list:

```
AdminTask.configEventServiceOracleDB(['-interactive'])
```

configEventServiceSQLServerDB command

Use the configEventServiceSQLServerDB command to configure the Common
Event Infrastructure using a SQL Server database.

Purpose

The configEventServiceSQLServerDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to create the event service database and data sources for SQL Server on a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- createDB
  The command generates the DDL database scripts and creates the database when this parameter is set to `true`. The command only generates the DDL database scripts when this parameter is set to `false`. To create the database, the current server must be already configured to run the database commands. The default value is `false` if not specified.

- overrideDataSource
  Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/*dbtype* if this parameter is not specified.

- nodeName
  The name of the node that contains the server where the event service data source should be created. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
  The name of the server where the event service data source should be created. If this parameters is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
  The name of the cluster where the event service data source should be created. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbServerName
  The server name of the SQL Server database. This parameter must be set when the parameter **createDB** is set to `true`.

- dbHostName
  The host name of the server where the SQL Server database is running.

- dbPort
  SQL Server port. The default value is 1433 if not specified.

- dbName
  The database name to be created. The default value is `event` if not specified.

- dbUser

The SQL Server user ID that will own the event service tables. The default value is `ceiuser` if not specified.

- dbPassword

The password of the SQL Server user ID specified by the **dbUser** parameter. This parameter is required.

- saUser

User ID that has privileges to create and drop databases and users. This parameter is required when the **createDB** parameter is set to `true`. The default value is `sa` if not specified.

- saPassword

The sa password. You must not specify this parameter if the sa user ID does not have a password.

- ceiInstancePrefix

The command uses the event service instance name to group the database files in a directory with unique names. The default value is `ceiinst1` if not specified.

- outputScriptDir

Optional database script output directory. When this parameter is specified, the command generates the event service database scripts in the specified directory. If the specified directory does not contain a full path, the command creates the specified directory in *profile_root*/bin. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/ sqlserver if this parameter is not specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask configEventServiceSQLServerDB {-createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -dbUser ceiuser
 -dbPassword ceipassword
 -dbServerName sqlservername
 -dbHostName host_name
 -dbPort 1433
 -saUser sa
 -saPassword sapassword}
```

- Using Jython string:

```
AdminTask.configEventServiceSQLServerDB('[-createDB true
 -overrideDataSource true
 -nodeName nodename
 -serverName servername
 -dbUser ceiuser
 -dbPassword ceipassword
 -dbServerName sqlservername
 -dbHostName host_name
 -dbPort 1433
 -saUser sa
 -saPassword sapassword]')
```

- Using Jython list:

```
AdminTask.configEventServiceSQLServerDB(['-createDB', 'true',
 '-overrideDataSource', 'true',
 '-nodeName', 'nodename',
```

```
                      '-serverName', 'servername',
                      '-dbUser', 'ceiuser'
                      '-dbPassword', 'ceipassword',
                      '-dbServerName', 'sqlservername',
                      '-dbHostName', 'host_name',
                      '-dbPort', '1433',
                      '-saUser', 'sa',
                      '-saPassword', 'sapassword'])
```

Interactive mode example usage:

• Using Jacl:

```
$AdminTask configEventServiceSQLServerDB -interactive
```

• Using Jython string:

```
AdminTask.configEventServiceSQLServerDB('[-interactive]')
```

• Using Jython list:

```
AdminTask.configEventServiceSQLServerDB(['-interactive'])
```

deployEventService command

Use the deployEventService command to deploy the event service application onto your server.

Purpose

The deployEventService command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to deploy and configure the event service on a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: | **Vista** | | **Windows 7** | The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName
    The name of the node where the event service should be deployed. If this parameter is specified, then the serverName parameter must be specified. You must not specify this parameter if the clusterName parameter is specified.

- serverName
    The name of the server where the event service should be deployed. You must specify this parameter if the **nodeName** parameter is specified. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
    The name of the cluster where the event service should be deployed. You must not specify this parameter if the **nodeName** or **serverName** parameter are specified.

- enable
    Set this parameter to `true` if you want the event service to be started after the next restart of the server. The default value is `true`.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask deployEventService {-nodeName node_name
 -serverName server_name}
$AdminTask deployEventService {-clusterName cluster_name
 -enable false}
```

- Using Jython string:

```
AdminTask.deployEventService('[-nodeName node_name
 -serverName server_name]')

AdminTask.deployEventService('[-clusterName cluster_name
 -enable false]')
```

- Using Jython list:

```
AdminTask.deployEventService(['-nodeName', 'node_name',
 '-serverName', '-server_name'])
AdminTask.deployEventService(['-clusterName', 'cluster_name',
 '-enable', 'false'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask deployEventService {-interactive}
```

- Using Jython string:

```
AdminTask.deployEventService('[-interactive]')
```

- Using Jython list:

```
AdminTask.deployEventService(['-interactive'])
```

deployEventServiceMdb command

Use the deployEventServiceMdb command to deploy the event service message driven bean onto your server.

Purpose

The deployEventServiceMdb command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to deploy the event service MDB to a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: | **Vista** | | Windows 7 | The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName
   The name of the node where the event service MDB should be deployed. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
   The name of the server where the event service MDB should be deployed. You must specify this parameter if the **nodeName** parameter is specified. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service MDB should be deployed. You must not specify this parameter if the **nodeName** and **serverName** parameters are specified.

- applicationName

The name of the event service MDB application to be deployed on a server or cluster.

- listenerPort

The name of the listener port where the event service MDB should publish the events. The listener port must already be created. You must not specify this parameter if the **activationSpec** parameter is specified.

- activationSpec

The JNDI name of activation specification where the event service MDB should publish the events. The activation specification must already be created. You must not specify this parameter if the **listenerPort** parameter is specified.

- qcfJndiName

The JNDI name of the JMS queue connection factory object to be used by the event service MDB. You must specify this parameter if the **activationSpec** parameter is specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask deployEventServiceMdb {-applicationName appname -nodeName nodename
 -serverName servername -listenerPort lpname}$AdminTask deployEventServiceMdb
 {-applicationName appname -clusterName clustername -activationSpec asjndiname
 -qcfJndiName qcfjndiname}
```

- Using Jython string:

```
AdminTask.deployEventServiceMdb('[-applicationName appname -nodeName nodename
 -serverName servername -listenerPort lpname]')AdminTask.deployEventServiceMdb
 ('[-applicationName appname -clusterName clustername -activationSpec asjndiname
 -qcfJndiName qcfjndiname]')
```

- Using Jython list:

```
AdminTask.deployEventServiceMdb(['-applicationName', 'appname', '-nodeName',
 'nodename', '-serverName', '-servername', ' -listenerPort', 'lpname'])
AdminTask.deployEventServiceMdb(['-applicationName', 'appname',
 '-clusterName', 'clustername', '-activationSpec', 'asjndiname',
 '-qcfJndiName', 'qcfjndiname'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask deployEventServiceMdb {-interactive}
```

- Using Jython string:

```
AdminTask.deployEventServiceMdb('[-interactive]')
```

- Using Jython list:

```
AdminTask.deployEventServiceMdb(['-interactive'])
```

---

setEventServiceJmsAuthAlias command

Use the setEventServiceJmsAuthAlias command to set or update the JMS authentication alias associated with the event service on your server.

Purpose

The setEventServiceJmsAuthAlias command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to update the authentication alias used by the event service JMS objects on a server or cluster. If the JMS authentication alias does not exist, it is created. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** Windows 7 The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName
The name of the node where the event service JMS authentication alias should be updated. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
The name of the server where the event service JMS authentication alias should be updated. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
The name of the cluster where the event service JMS authentication alias should be updated. You must not specify this parameter if the **nodeName** and **serverName** parameters are specified.

- userName
The name of the user to be used in the update of the event service JMS authentication alias on a server or cluster.

Important: You must specify a valid user ID; this field cannot be empty.

- password
The password of the user to be used in the update of the event service JMS authentication alias on a server or cluster.

Important: You must specify a valid password; this field cannot be empty.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask setEventServiceJmsAuthAlias{-nodeName nodename
 -serverName servername username -password pwd}
$AdminTask setEventServiceJmsAuthAlias {-clusterName clustername
 -userName username -password pwd}
```

- Using Jython string:

```
AdminTask.setEventServiceJmsAuthAlias('[-nodeName nodename
 -serverName servername -userName username -password pwd]')
AdminTask.setEventServiceJmsAuthAlias('[-clusterName clustername
 -userName username -password pwd]')
```

- Using Jython list:

```
AdminTask.setEventServiceJmsAuthAlias(['-nodeName', 'nodename',
 '-serverName', '-servername', '-userName', 'username', '-password', 'pwd'])
AdminTask.setEventServiceJmsAuthAlias(['-clusterName',
 'clustername', '-userName', 'username', '-password', 'pwd'])
```

Interactive mode example usage:

- Using Jacl:

  ```
  $AdminTask setEventServiceJmsAuthAlias {-interactive}
  ```

- Using Jython string:

  ```
  AdminTask.setEventServiceJmsAuthAlias( '[-interactive]')
  ```

- Using Jython list:

  ```
  AdminTask.setEventServiceJmsAuthAlias(['-interactive'])
  ```

enableEventService command

Use the enableEventService command to enable the event service on your server.

Purpose

The enableEventService command is a Common Event Infrastructure
administrative command is available for the AdminTask object. Use this command
to enable the event service to be started after the next restart of the server(s)
designated by the nodeName, serverName, or clusterName parameters. For more
information about the AdminTask object, see the WebSphere Application Server
Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not
support Microsoft Windows 2003, Windows Vista, or Windows 7 operating
systems.

DITA

Parameters

- nodeName

  The name of the node where the event service should be enabled. If this
  parameter is specified, then the **serverName** parameter must be specified. You
  must not specify this parameter if the **clusterName** parameter is specified.

- serverName

  The name of the server where the event service should be enabled. You must
  specify this parameter if the **nodeName** parameter is specified. You must not
  specify this parameter if the **clusterName** parameter is specified.

- clusterName

  The name of the cluster where the event service should be enabled. You must
  not specify this parameter if the **nodeName** and **serverName** parameters are
  specified.

Sample

Batch mode example usage:

- Using Jacl:

  ```
  $AdminTask enableEventService {-nodeName nodename
   -serverName servername}

  $AdminTask enableEventService {-clusterName clustername}
  ```

- Using Jython string:

```
AdminTask.enableEventService('[-nodeName nodename
 -serverName servername]')

AdminTask.enableEventService('[-clusterName clustername]')
```

- Using Jython list:

```
AdminTask.enableEventService(['-nodeName', 'nodename',
 '-serverName', '-servername'])

AdminTask.enableEventService(['-clusterName', 'clustername'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask enableEventService {-interactive}
```

- Using Jython string:

```
AdminTask.enableEventService('[-interactive]')
```

- Using Jython list:

```
AdminTask.enableEventService(['-interactive'])
```

disableEventService command

Use the disableEventService command to disable the event service on your server.

Purpose

The disableEventService command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to disable the event service from starting after the next restart of the server(s) designated by the nodeName, serverName, or clusterName parameters. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista**   The product uses a Jython version that does not support Microsoft Windows 2003 or Windows Vista operating systems.

Parameters

- nodeName
> The name of the node where the event service should be disabled. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
> The name of the server where the event service should be disabled. You must specify this parameter if the **nodeName** parameter is specified. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
> The name of the cluster where the event service should be disabled. You must not specify this parameter if the **nodeName** and **serverName** parameters are specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask disableEventService {-nodeName nodename
 -serverName servername}

$AdminTask disableEventService {-clusterName clustername}
```
- Using Jython string:
```
AdminTask.disableEventService('[-nodeName nodename
 -serverName servername]')

AdminTask.disableEventService('[-clusterName clustername]')
```
- Using Jython list:
```
AdminTask.disableEventService(['-nodeName', 'nodename',
 '-serverName', '-servername'])

AdminTask.disableEventService(['-clusterName', 'clustername'])
```

Interactive mode example usage:
- Using Jacl:
```
$AdminTask disableEventService {-interactive}
```
- Using Jython string:
```
AdminTask.disableEventService('[-interactive]')
```
- Using Jython list:
```
AdminTask.disableEventService(['-interactive'])
```

showEventServiceStatus command

Use the showEventServiceStatus command to display the event service status on your server.

Purpose

The showEventServiceStatus command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to return the status of the event service in a server or cluster. If the task is executed with no parameters, the status of all event services is displayed. To filter the list of event services to be displayed, provide nodeName, serverName, or clusterName. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName
> Use this parameter to display only the status of the event services that belong to the specified node. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
> Use this parameter to display only the status of the event services that belong to the specified server. You can use this parameter with the **nodeName** parameter to display the status of the event service belonging to the specified node and server. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
> Use this parameter to only display the status of the event services that belong to the specified cluster. You must not specify this parameter if the **nodeName** or **serverName** parameters are specified.

Sample

Batch mode example usage:
- Using Jacl:

```
$AdminTask showEventServiceStatus {-nodeName nodename
 -serverName servername}

$AdminTask showEventServiceStatus {-clusterName clustername}
```
- Using Jython string:

```
AdminTask.showEventServiceStatus('[-nodeName nodename
 -serverName servername]')

AdminTask.showEventServiceStatus('[-clusterName clustername]')
```
- Using Jython list:

```
AdminTask.showEventServiceStatus(['-nodeName', 'nodename',
 '-serverName', '-servername'])

AdminTask.showEventServiceStatus(['-clusterName', 'clustername'])
```

Interactive mode example usage:
- Using Jacl:

```
$AdminTask showEventServiceStatus {-interactive}
```
- Using Jython string:

```
AdminTask.showEventServiceStatus('[-interactive]')
```
- Using Jython list:

```
AdminTask.showEventServiceStatus(['-interactive'])
```

removeEventService command

Use the removeEventService command to remove the event service application from your server.

Purpose

The removeEventService command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the event service from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: ▆▆▆ **Vista** ▆▆▆ ▆▆ Windows 7 ▆▆ The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName
> The name of the node from which the event service should be removed. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server from which the event service should be removed. You must specify this parameter if the **nodeName** parameter is specified. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster from which the event service should be removed. You must not specify this parameter if the **nodeName** and **serverName** parameters are specified.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask removeEventService {-nodeName nodename
 -serverName servername}

$AdminTask removeEventService {-clusterName clustername}
```

- Using Jython string:

```
AdminTask.removeEventService('[-nodeName nodename
 -serverName servername]')

AdminTask.removeEventService('[-clusterName clustername]')
```

- Using Jython list:

```
AdminTask.removeEventService(['-nodeName', 'nodename',
 '-serverName', '-servername'])

AdminTask.removeEventService(['-clusterName', 'clustername'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask removeEventService {-interactive}
```

- Using Jython string:

```
AdminTask.removeEventService('[-interactive]')
```

- Using Jython list:

```
AdminTask.removeEventService(['-interactive'])
```

removeEventServiceMdb command

Use the removeEventServiceMdb command to remove the event service message driven bean from your server.

Purpose

The removeEventServiceMdb command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the event service MDB from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: Vista Windows 7 The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName

The name of the node where the event service MDB should be removed. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service MDB should be removed. If this parameter is specified, then the **serverName** parameter must be specified. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service MDB should be removed. You must not specify this parameter if the **nodeName** and **serverName** parameters are specified.

- applicationName

The name of the event service MDB application to be removed from a server or cluster.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceMdb {-applicationName appname
 -nodeName nodename -serverName servername}
$AdminTask removeEventServiceMdb {-applicationName appname
 -clusterName clustername}
```

- Using Jython string:

```
AdminTask.removeEventServiceMdb('[-applicationName appname
 -nodeName nodename -serverName servername]')
AdminTask.removeEventServiceMdb('[-applicationName appname
 -clusterName clustername]')
```

- Using Jython list:

```
AdminTask.removeEventServiceMdb (['-applicationName',
 'appname', '-nodeName', 'nodename', '-serverName', 'servername'])
AdminTask.removeEventServiceMdb (['-applicationName',
 'appname', '-clusterName', 'clustername'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceMdb {-interactive}
```

- Using Jython string:

```
AdminTask.removeEventServiceMdb('[-interactive]')
```

- Using Jython list:

```
AdminTask.removeEventServiceMdb(['-interactive'])
```

removeEventServiceDB2DB command

Use the removeEventServiceDB2DB command to remove the event service and, optionally, the associated DB2 event database.

Purpose

The removeEventServiceDB2DB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command

to remove the event service database and data sources for DB2 from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- removeDB
> The command removes the database when this parameter is set to `true` and does not remove the database when set to `false`. To remove the database, the current server must already configured to run the database commands.

- nodeName
> The name of the node that contains the server where the event service data source should be removed. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
> The name of the server where the event service data source should be removed. If this parameter is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
> The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbUser
> DB2 user ID that has privileges to create and drop the databases. This parameter must be set when the parameter **removeDB** is set to `true`. The default value is `db2inst1` if not specified.

- dbPassword
> DB2 password. This parameter must be set when the parameter **removeDB** is set to true.

- dbScriptDir
> The directory containing the database scripts generated by the event service database configuration command. If specified, the command will run the scripts in this directory to remove the event service database. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/db2.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceDB2DB {-removeDB true
  -nodeName nodename
  -serverName servername
  -dbUser db2inst1
  -dbPassword dbpassword }
```

- Using Jython string:

```
AdminTask.removeEventServiceDB2DB('[-removeDB true
 -nodeName nodename
 -serverName servername
 -dbUser db2inst1
 -dbPassword dbpassword]')
```

- Using Jython list:

```
AdminTask.removeEventServiceDB2DB(['-removeDB', 'true',
 '-nodeName', 'nodename',
 '-serverName', 'servername',
 '-dbUser', 'db2inst1',
 '-dbPassword', 'dbpassword'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceDB2DB -interactive
```

- Using Jython string:

```
AdminTask.removeEventServiceDB2DB('[-interactive]')
```

- Using Jython list:

```
AdminTask.removeEventServiceDB2DB(['-interactive'])
```

removeEventServiceDB2iSeriesDB command

Use the removeEventServiceDB2iSeriesDB command to remove the event service and, optionally, the associated DB2 for iSeries event database.

Purpose

The removeEventServiceDB2iSeriesDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the DB2 for iSeries data sources from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- nodeName
The name of the node that contains the server where the event service data source should be removed. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName
The name of the server where the event service data source should be removed. If this parameter is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName
The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

Sample

Batch mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceDB2iSeriesDB {-nodeName nodename
   -serverName servername }
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceDB2iSeriesDB('[-nodeName nodename
   -serverName servername]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceDB2iSeriesDB(['-nodeName', 'nodename',
   '-serverName', 'servername'])
  ```

Interactive mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceDB2iSeriesDB -interactive
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceDB2iSeriesDB('[-interactive]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceDB2iSeriesDB(['-interactive'])
  ```

removeEventServiceDB2ZOSDB command

Use the removeEventServiceDB2ZOSDB command to remove the event service and, optionally, the associated DB2 for z/OS event database.

Purpose

The removeEventServiceDB2ZOSDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the event service database and data sources for DB2 z/OS from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- removeDB

The command removes the database when this parameter is set to `true` and does not remove the database when set to `false`. To remove the database, the current server must already configured to run the database commands.

- nodeName

The name of the node that contains the server where the event service data source should be removed. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be removed. If this parameter is specified without the **nodeName** parameter, the

command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbName

The DB2 database name. On the DB2 client, it is the name of the catalogued database. On the native z/OS server, it is the name of the database subsystem. This parameter must be set when the parameter **removeDB** is set to `true`. The default value is `event` if not specified.

- dbUser

DB2 user ID that has privileges to create and drop the databases. This parameter must be set when the parameter **removeDB** is set to true.

- dbPassword

DB2 password. This parameter must be set when the parameter **removeDB** is set to true.

- dbScriptDir

The directory containing the database scripts generated by the event service database configuration command. If specified, the command will run the scripts in this directory to remove the event service database. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/db2zos.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceDB2ZOSDB {-removeDB true -nodeName nodename
 -serverName servername -dbUser db2user -dbPassword dbpassword
```

- Using Jython string:

```
AdminTask.removeEventServiceDB2ZOSDB('[-removeDB true -nodeName nodename
 -serverName servername -dbUser db2user -dbPassword dbpassword]')
```

- Using Jython list:

```
AdminTask.removeEventServiceDB2ZOSDB(['-removeDB', 'true', '-nodeName',
 'nodename', '-serverName', 'servername', '-dbUser', 'db2user',
 '-dbPassword', 'dbpassword'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceDB2ZOSDB -interactive
```

- Using Jython string:

```
AdminTask.removeEventServiceDB2ZOSDB('[-interactive]')
```

- Using Jython list:

```
AdminTask.removeEventServiceDB2ZOSDB(['-interactive'])
```

removeEventServiceDerbyDB command

Use the removeEventServiceDerbyDB command to remove the event service and, optionally, the associated Derby event database.

Purpose

The removeEventServiceDerbyDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the Event Service database and data source for Derby from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** Windows 7 The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- removeDB

> The command removes the database when this parameter is set to `true` and does not remove the database when set to `false`. To remove the database, the current server must already configured to run the database commands.

- nodeName

> The name of the node that contains the server where the event service data source should be removed. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

> The name of the server where the event service data source should be removed. If this parameter is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

> The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbScriptDir

> The directory containing the database scripts generated by the event service database configuration command. If specified, the command will run the scripts in this directory to remove the event service database. The default database script output directory is *profile_root*/databases/event/*node/server*/dbscripts/derby.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceDerbyDB {-removeDB true -nodeName nodename
  -serverName servername}
```

- Using Jython string:

```
AdminTask.removeEventServiceDerbyDB('[-removeDB true -nodeName nodename
  -serverName servername]')
```

- Using Jython list:

```
AdminTask.removeEventServiceDerbyDB(['-removeDB', 'true', '-nodeName',
  'nodename', '-serverName', 'servername'])
```

Interactive mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceDerbyDB -interactive
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceDerbyDB( '[-interactive]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceDerbyDB(['-interactive'])
  ```

removeEventServiceInformixDB command

Use the removeEventServiceInformixDB command to remove the event service and, optionally, the associated Informix event database.

Purpose

The removeEventServiceInformixDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the event service database and data sources for Informix from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: Vista   Windows 7   The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- removeDB

The command removes the database when this parameter is set to `true` and does not remove the database when set to `false`. To remove the database, the current server must already configured to run the database commands.

- nodeName

The name of the node that contains the server where the event service data source should be removed. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be removed. If this parameter is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbScriptDir

The directory containing the database scripts generated by the event service database configuration command. If specified, the command will run the scripts in this directory to remove the event service database. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/informix.

Sample

Batch mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceInformixDB {-removeDB true -nodeName nodename
   -serverName servername}
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceInformixDB('[-removeDB true -nodeName nodename
   -serverName servername]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceInformixDB(['-removeDB', 'true', '-nodeName',
   'nodename', '-serverName', 'servername'])
  ```

Interactive mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceInformixDB -interactive
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceInformixDB('[-interactive]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceInformixDB(['-interactive'])
  ```

removeEventServiceOracleDB command

Use the removeEventServiceOracleDB command to remove the event service and,
optionally, the associated Oracle event database.

Purpose

The removeEventServiceOracleDB command is a Common Event Infrastructure
administrative command is available for the AdminTask object. Use this command
to remove the event service tables and data sources for Oracle from a server or
cluster. The command does not remove the database. For more information about
the AdminTask object, see the WebSphere Application Server Network
Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not
support Microsoft Windows 2003, Windows Vista, or Windows 7 operating
systems.

Parameters

- removeDB

  The command removes the event service tables when this parameter is set to
  `true` and does not remove the tables when set to `false`.

- nodeName

  The name of the node that contains the server where the event service data
  source should be removed. If this parameter is specified, then the **serverName**
  parameter must be set. You must not specify this parameter if the **clusterName**
  parameter is specified.

- serverName

  The name of the server where the event service data source should be
  removed. If this parameter is specified without the **nodeName** parameter, the

command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- **clusterName**

  The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- **sysUser**

  Oracle database sys user ID. The default value is sys if not specified.

- **sysPassword**

  The password for the user specified by the sysUser parameter.

- **dbScriptDir**

  The directory containing the database scripts generated by the event service database configuration command. If specified, the command will run the scripts in this directory to remove the event service database. The default database script output directory is *profile_root*/databases/event/*node*/*server*/dbscripts/oracle.

Sample

Batch mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceOracleDB {-removeDB true -nodeName nodename
   -serverName servername -sysUser sys -sysPassword syspassword}
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceOracleDB('[-removeDB true -nodeName nodename
   -serverName servername -sysUser sys -sysPassword syspassword]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceOracleDB(['-removeDB', 'true', '-nodeName',
   'nodename', '-serverName', 'servername', '-sysUser', 'sys',
   '-sysPassword', 'syspassword'])
  ```

Interactive mode example usage:

- Using Jacl:

  ```
  $AdminTask removeEventServiceOracleDB -interactive
  ```

- Using Jython string:

  ```
  AdminTask.removeEventServiceOracleDB('[-interactive]')
  ```

- Using Jython list:

  ```
  AdminTask.removeEventServiceOracleDB(['-interactive'])
  ```

removeEventServiceSQLServerDB command

Use the removeEventServiceSQLServerDB command to remove the event service and, optionally, the associated SQL Server event database.

Purpose

The removeEventServiceSQLServerDB command is a Common Event Infrastructure administrative command is available for the AdminTask object. Use this command to remove the event service database and data sources for SQL Server from a server or cluster. For more information about the AdminTask object, see the WebSphere Application Server Network Deployment, version 6.1 documentation.

Note: **Vista** **Windows 7** The product uses a Jython version that does not support Microsoft Windows 2003, Windows Vista, or Windows 7 operating systems.

Parameters

- removeDB

The command removes the database when this parameter is set to `true` and does not remove the database when set to `false`. To remove the database, the current server must already configured to run the database commands.

- nodeName

The name of the node that contains the server where the event service data source should be removed. If this parameter is specified, then the **serverName** parameter must be set. You must not specify this parameter if the **clusterName** parameter is specified.

- serverName

The name of the server where the event service data source should be removed. If this parameter is specified without the **nodeName** parameter, the command will use the node name of the current WebSphere profile. You must not specify this parameter if the **clusterName** parameter is specified.

- clusterName

The name of the cluster where the event service data source should be removed. If this parameter is specified, then the **serverName** and **nodeName** parameters must not be set. You must not specify this parameter if the **serverName** and **nodeName** parameters are specified.

- dbServerName

The server name of the SQL Server database. This parameter must be set when the parameter **removeDB** is set to `true`.

- dbUser

SQL Server user ID that owns the event service tables. The default value is `ceiuser` if not specified.

- saUser

User ID that has privileges to drop the databases and users. The default value is `sa` if not specified.

- saPassword

The password specified by the **saUser** parameter. This parameter is required when the parameter **removeDB** is set to `true`.

- dbScriptDir

The directory containing the database scripts generated by the event service database configuration command. If specified, the command will run the scripts in this directory to remove the event service database. The default database script output directory is *profile_root*/databases/event/*node/server/* dbscripts/sqlserver.

Sample

Batch mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceSQLServerDB {-removeDB true -nodeName nodename
 -serverName servername -dbUser ceiuser -saUser sa -saPassword sapassword
 -dbServerName sqlservername}
```

- Using Jython string:

```
AdminTask.removeEventServiceSQLServerDB('[-removeDB true -nodeName nodename
  -serverName servername -dbUser ceiuser -saUser sa -saPassword sapassword
  -dbServerName sqlservername]')
```

- Using Jython list:

```
AdminTask.removeEventServiceSQLServerDB(['-removeDB', 'true', '-nodeName',
  'nodename', '-serverName', 'servername', '-dbUser', 'ceiuser', '-saUser','sa',
  '-saPassword', 'sapassword', '-dbServerName', 'sqlservername'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask removeEventServiceSQLServerDB -interactive
```

- Using Jython string:

```
AdminTask.removeEventServiceSQLServerDB('[-interactive]')
```

- Using Jython list:

```
AdminTask.removeEventServiceSQLServerDB(['-interactive'])
```

eventbucket command-line utility

The **eventbucket** command displays or changes the event database bucket
configuration.

Purpose

Displays or changes the event database bucket configuration.

eventbucket [**-status**] [**-change**]

Description

The eventbucket command displays or changes the event database bucket
configuration. Buckets are used by the rapid purge utility to purge old event data
from the event database. By running this command, you can determine the current
bucket configuration, or you can swap the active and inactive buckets.

Security: If WebSphere security is enabled, your user ID must be mapped to the
eventAdministrator role to view or change the event database bucket
configuration.

Parameters

-status
    Displays information about the current bucket configuration, including the
    active bucket setting and the bucket check interval (the frequency with which
    the data store plug-in checks to determine which bucket is active).

-change
    Swaps the buckets so the active bucket becomes inactive and the inactive
    bucket becomes active. The inactive bucket must be empty before you can use
    this option.

Examples

This example displays the current bucket configuration:
```
eventbucket -status
```

This example swaps the active and inactive buckets:

eventpurge command-line utility

The **eventpurge** command deletes events from the event database.

Purpose

Deletes events from the event database.

eventpurge [**-seconds** *seconds* | **-end** *end_time*] [**-group** *event_group*] [**-severity** *severity*] [**-extensionname** *extension_name*] [**-start** *start_time*] [**-size** *size*]

Description

The eventpurge command deletes events from the event database. You can delete all events from the event database, or you can limit the deletion to events meeting certain criteria.

Security: If WebSphere security is enabled, your user ID must be mapped to the eventAdministrator role to delete events.

Parameters

-seconds *seconds*
> The minimum age of events you want deleted. The *seconds* value must be an integer. Only events older than the specified number of seconds are deleted. This parameter is required if you do not specify the **-end** parameter.

-end *end_time*

> The end time of the group of events you want to delete. Only events generated before the specified time are deleted. The *end_time* value must be specified in the XML dateTime format (`CCYY-MM-DDThh:mm:ss`). For example, noon on 1 January 2006 in Eastern Standard Time would be `2006-01-01T12:00:00-05:00`. For more information about the dateTime data type, refer to the XML schema at www.w3.org.

> This parameter is required if you do not specify the **-seconds** parameter.

-group *eventGroup*
> The event group from which to delete events. The *event_group* value must be the name of an event group defined in the Common Event Infrastructure configuration. This parameter is optional.

-severity *severity*
> The severity of the events you want deleted. The *severity* value must be an integer; only events whose severity is equal to the value you specify are deleted. This parameter is optional.

-extensionname *extension_name*
> The extension name of the events you want included in the deletion. Use this parameter to restrict the deletion to events of a specific type. Only events whose extensionName property is equal to *extensionName* are deleted. This parameter is optional.

-start *start_time*
> The beginning time of the group of events you want to delete. Only events

generated after the specified time are deleted. The *start_time* value must be
specified in the XML dateTime format (`CCYY-MM-DDThh:mm:ss`). This parameter
is optional.

-size *size*

The number of events to delete in a single transaction. The *size* value must be
an integer. After this number of events have been deleted, the command
commits the transaction before continuing in a new transaction. This parameter
is optional.

Example

The following example deletes all events from the database whose severity is 20
(harmless) and were generated earlier than 10 minutes ago.

```
eventpurge -group "All events" -severity 20 -seconds 600
```

IBM ®

Printed in USA